Note: Before using this information and the product it supports, read the information in "Notices" at the end of this document.

First Edition (December, 2005)

This edition applies to IBM® Lotus® Domino® Designer 7 (product number L-GHUS-5RWNHM), and to all subsequent releases and modifications, until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 1994, 2005. All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Accessing the current document in the formula language.
Examples: Accessing the current document
Accessing data outside the current document and database.
Examples: Accessing data outside the current document and database.
Accessing external databases through LS:DO using @functions.
Examples: Accessing external databases through LS:DO using @functions.

Chapter 6. Formula Language
@Functions A-Z

Where does this @function work? (Part 1 A -- D) 167
Where does this @function work? (Part 1 E -- K) 170
Where does this @function work? (Part 1 L -- R) 174
Where does this @function work? (Part 1 S -- Z) 177
Where does this @function work? (Part 2 A -- D) 181
Where does this @function work? (Part 2 E -- K) 184
Where does this @function work? (Part 2 L -- R) 188
Where does this @function work? (Part 2 S -- Z) 191
@Functions with ECL security
@Abs
Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @Abs
@Abstract
Syntax
Parameters
Return value
Keywords
Rules
Files
Language cross-reference
Examples: @Abstract
@Accessed
Syntax
Return value
Usage
Usage in workflow applications
Usage in column or selection formulas
Language cross-reference
Examples: @Accessed
@ACos
Syntax
Parameters
Return value
Language cross-reference
Examples: @ACos
@AddToFolder
Syntax
Parameters
Usage
Language cross-reference
Examples: @AddToFolder
@Adjust
Syntax

Parameters
Return value
Usage
Calculating due dates
Language cross-reference
Examples: @Adjust
@AdminECLIsLocked
Syntax
Return value
Usage
@All
Syntax
Return value
Usage
Examples: @All
@AllChildren
Syntax
Return value
Usage
Examples: @AllChildren
@AllDescendants
Syntax
Return value
Usage
Examples: @AllDescendants
@Ascii
Syntax
Parameters
Return value
Usage
Examples: @Ascii
@ASin
Syntax
Parameters
Return value
Language cross-reference
Examples: @ASin
@ATan
Syntax
Parameters
Return value
Language cross-reference
Examples: @ATan
@ATan2
Syntax
Parameters
Return value
Language cross-reference
Examples: @ATan2
@AttachmentLengths
Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @AttachmentLengths
@AttachmentModifiedTimes
Syntax
Parameters
Return value
Examples: @AttachmentModifiedTimes
@AttachmentNames
Examples: @GetField .......................... 306
@GetFocusTable ................................ 306
Syntax. ......................................... 306
Parameters ..................................... 306
Return value ................................... 307
Usage ............................................ 307
Examples: @GetFocusTable ......................... 307
@GetHTTPHeader ................................ 307
Syntax. ......................................... 307
Parameters ..................................... 307
Return value ................................... 308
Usage ............................................ 308
Language cross-reference ........................ 308
Examples: @GetHTTPHeader ......................... 308
@GetIMContactListGroupNames ...................... 308
Syntax. ......................................... 308
Return value ................................... 308
Usage ............................................ 309
Examples: @GetIMContactListGroupNames .......... 308
@GetPortsList .................................. 309
Syntax. ......................................... 309
Parameters ..................................... 309
Return value ................................... 309
Usage ............................................ 309
Examples: @GetPortsList .......................... 309
@GetProfileField ................................ 309
Syntax. ......................................... 309
Parameters ..................................... 310
Return value ................................... 310
Usage ............................................ 310
Language cross-reference ........................ 310
Examples: @GetProfileField ......................... 310
@GetViewInfo ................................... 311
Syntax. ......................................... 311
Parameters ..................................... 311
Return value ................................... 311
Usage ............................................ 311
Examples: @GetViewInfo .......................... 312
@HardDeleteDocument .............................. 312
Syntax. ......................................... 312
Usage ............................................ 312
Language cross-reference ........................ 312
@HashPassword .................................. 312
Syntax. ......................................... 313
Parameters ..................................... 313
Return value ................................... 313
Usage ............................................ 313
Examples: @HashPassword .......................... 313
@Hour ........................................... 313
Syntax. ......................................... 313
Parameters ..................................... 313
Return value ................................... 313
Language cross-reference ........................ 313
Examples: @Hour ................................ 313
@if .............................................. 314
Syntax. ......................................... 314
Parameters ..................................... 314
Usage ............................................ 314
Language cross-reference ........................ 314
Examples: @If ................................... 314
@ifError .......................................... 315
Syntax. ......................................... 315
Parameters ..................................... 315
Return value ................................... 315
Usage ............................................ 316
Language cross-reference ........................ 316
Examples: @IfError ............................... 316
@Implode ........................................ 316
Syntax. ......................................... 316
Parameters ..................................... 316
Return value ................................... 317
Language cross-reference ........................ 317
Examples: @Implode ................................ 317
@InheritedDocumentUniqueID ....................... 317
Syntax. ......................................... 317
Usage ............................................ 317
Language cross-reference ........................ 317
Examples: @InheritedDocumentUniqueID .......... 318
@Integer ........................................ 318
Syntax. ......................................... 318
Parameters ..................................... 318
Return value ................................... 318
Usage ............................................ 318
Language cross-reference ........................ 318
Examples: @Integer ................................ 319
@IsAgentEnabled ................................ 319
Syntax. ......................................... 319
Parameters ..................................... 319
Return value ................................... 319
Usage ............................................ 319
Language cross-reference ........................ 319
Examples: @IsAgentEnabled ......................... 319
@IsAppInstalled ................................. 320
Syntax. ......................................... 320
Parameters ..................................... 320
Return value ................................... 320
Usage ............................................ 320
@IsAvailable ................................... 320
Syntax. ......................................... 320
Parameters ..................................... 321
Return value ................................... 321
Usage ............................................ 321
Language cross-reference ........................ 321
Examples: @IsAvailable ........................... 321
@IsCategory ..................................... 321
Syntax. ......................................... 321
Parameters ..................................... 321
Return value ................................... 321
Usage ............................................ 322
Examples: @IsCategory ........................... 322
@IsDB2 ........................................... 322
Syntax. ......................................... 322
Parameters ..................................... 322
Return value ................................... 322
Usage ............................................ 323
Examples: @IsDB2 ................................ 323
@IsDocBeingEdited ............................... 323
Syntax. ......................................... 323
Parameters ..................................... 323
Return value ................................... 323
Usage ............................................ 323
Language cross-reference ........................ 324
Examples: @IsDocBeingEdited ...................... 324
@IsDocBeingLoaded ............................... 324
Syntax. ......................................... 324
<table>
<thead>
<tr>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>@MiddleBack</td>
<td>364</td>
</tr>
<tr>
<td>Syntax</td>
<td>364</td>
</tr>
<tr>
<td>Parameters</td>
<td>364</td>
</tr>
<tr>
<td>Return value</td>
<td>365</td>
</tr>
<tr>
<td>Examples: @MiddleBack</td>
<td>365</td>
</tr>
<tr>
<td>@Min</td>
<td>365</td>
</tr>
<tr>
<td>Syntax</td>
<td>365</td>
</tr>
<tr>
<td>Parameters</td>
<td>365</td>
</tr>
<tr>
<td>Return value</td>
<td>365</td>
</tr>
<tr>
<td>Usage</td>
<td>366</td>
</tr>
<tr>
<td>Examples: @Min</td>
<td>366</td>
</tr>
<tr>
<td>@Minute</td>
<td>366</td>
</tr>
<tr>
<td>Syntax</td>
<td>366</td>
</tr>
<tr>
<td>Parameters</td>
<td>366</td>
</tr>
<tr>
<td>Return value</td>
<td>366</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>366</td>
</tr>
<tr>
<td>Examples: @Minute</td>
<td>366</td>
</tr>
<tr>
<td>@Modified</td>
<td>367</td>
</tr>
<tr>
<td>Syntax</td>
<td>367</td>
</tr>
<tr>
<td>Return value</td>
<td>367</td>
</tr>
<tr>
<td>Usage</td>
<td>367</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>367</td>
</tr>
<tr>
<td>Examples: @Modified</td>
<td>367</td>
</tr>
<tr>
<td>@Modulo</td>
<td>367</td>
</tr>
<tr>
<td>Syntax</td>
<td>367</td>
</tr>
<tr>
<td>Parameters</td>
<td>368</td>
</tr>
<tr>
<td>Return value</td>
<td>368</td>
</tr>
<tr>
<td>Usage</td>
<td>368</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>368</td>
</tr>
<tr>
<td>Examples: @Modulo</td>
<td>368</td>
</tr>
<tr>
<td>@Month</td>
<td>368</td>
</tr>
<tr>
<td>Syntax</td>
<td>368</td>
</tr>
<tr>
<td>Parameters</td>
<td>368</td>
</tr>
<tr>
<td>Return value</td>
<td>369</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>369</td>
</tr>
<tr>
<td>Examples: @Month</td>
<td>369</td>
</tr>
<tr>
<td>@Name</td>
<td>369</td>
</tr>
<tr>
<td>Syntax</td>
<td>369</td>
</tr>
<tr>
<td>Parameters</td>
<td>369</td>
</tr>
<tr>
<td>Usage</td>
<td>372</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>373</td>
</tr>
<tr>
<td>Examples: @Name</td>
<td>373</td>
</tr>
<tr>
<td>@NameLookup</td>
<td>374</td>
</tr>
<tr>
<td>Syntax</td>
<td>374</td>
</tr>
<tr>
<td>Parameters</td>
<td>374</td>
</tr>
<tr>
<td>Return value</td>
<td>375</td>
</tr>
<tr>
<td>Usage</td>
<td>375</td>
</tr>
<tr>
<td>Examples: @NameLookup</td>
<td>375</td>
</tr>
<tr>
<td>@Narrow</td>
<td>376</td>
</tr>
<tr>
<td>Syntax</td>
<td>376</td>
</tr>
<tr>
<td>Parameters</td>
<td>376</td>
</tr>
<tr>
<td>Return value</td>
<td>376</td>
</tr>
<tr>
<td>Usage</td>
<td>376</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>376</td>
</tr>
<tr>
<td>Examples: @Narrow</td>
<td>376</td>
</tr>
<tr>
<td>@NewLine</td>
<td>376</td>
</tr>
<tr>
<td>Syntax</td>
<td>376</td>
</tr>
<tr>
<td>Return value</td>
<td>377</td>
</tr>
<tr>
<td>Usage</td>
<td>377</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>377</td>
</tr>
<tr>
<td>Examples: @NewLine</td>
<td>377</td>
</tr>
<tr>
<td>@No</td>
<td>377</td>
</tr>
<tr>
<td>Syntax</td>
<td>378</td>
</tr>
<tr>
<td>Return value</td>
<td>378</td>
</tr>
<tr>
<td>Usage</td>
<td>378</td>
</tr>
<tr>
<td>Examples: @No</td>
<td>378</td>
</tr>
<tr>
<td>@NoteID</td>
<td>378</td>
</tr>
<tr>
<td>Syntax</td>
<td>378</td>
</tr>
<tr>
<td>Return value</td>
<td>378</td>
</tr>
<tr>
<td>Usage</td>
<td>378</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>378</td>
</tr>
<tr>
<td>@Nothing</td>
<td>378</td>
</tr>
<tr>
<td>Syntax</td>
<td>378</td>
</tr>
<tr>
<td>Return value</td>
<td>379</td>
</tr>
<tr>
<td>Usage</td>
<td>379</td>
</tr>
<tr>
<td>@Now</td>
<td>379</td>
</tr>
<tr>
<td>Syntax</td>
<td>379</td>
</tr>
<tr>
<td>Parameters</td>
<td>379</td>
</tr>
<tr>
<td>Return value</td>
<td>379</td>
</tr>
<tr>
<td>Usage</td>
<td>379</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>380</td>
</tr>
<tr>
<td>Examples: @Now</td>
<td>380</td>
</tr>
<tr>
<td>@OptimizeMailAddress</td>
<td>380</td>
</tr>
<tr>
<td>Syntax</td>
<td>380</td>
</tr>
<tr>
<td>Parameters</td>
<td>380</td>
</tr>
<tr>
<td>Return value</td>
<td>380</td>
</tr>
<tr>
<td>Usage</td>
<td>380</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>380</td>
</tr>
<tr>
<td>Examples: @OptimizeMailAddress</td>
<td>380</td>
</tr>
<tr>
<td>@OrgDir</td>
<td>381</td>
</tr>
<tr>
<td>Syntax</td>
<td>381</td>
</tr>
<tr>
<td>Parameters</td>
<td>381</td>
</tr>
<tr>
<td>Return value</td>
<td>381</td>
</tr>
<tr>
<td>Usage</td>
<td>381</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>381</td>
</tr>
<tr>
<td>Examples: @OrgDir</td>
<td>381</td>
</tr>
<tr>
<td>@Password</td>
<td>381</td>
</tr>
<tr>
<td>Syntax</td>
<td>381</td>
</tr>
<tr>
<td>Parameters</td>
<td>381</td>
</tr>
<tr>
<td>Return value</td>
<td>381</td>
</tr>
<tr>
<td>Usage</td>
<td>381</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>382</td>
</tr>
<tr>
<td>Examples: @Password</td>
<td>382</td>
</tr>
<tr>
<td>@PasswordQuality</td>
<td>382</td>
</tr>
<tr>
<td>Syntax</td>
<td>382</td>
</tr>
<tr>
<td>Parameters</td>
<td>382</td>
</tr>
<tr>
<td>Return value</td>
<td>382</td>
</tr>
<tr>
<td>Usage</td>
<td>382</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>382</td>
</tr>
<tr>
<td>@Pi</td>
<td>382</td>
</tr>
<tr>
<td>Syntax</td>
<td>382</td>
</tr>
<tr>
<td>Return value</td>
<td>382</td>
</tr>
<tr>
<td>Usage</td>
<td>383</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>383</td>
</tr>
<tr>
<td>Examples: @Pi</td>
<td>383</td>
</tr>
<tr>
<td>@PickList</td>
<td>383</td>
</tr>
<tr>
<td>Syntax</td>
<td>383</td>
</tr>
<tr>
<td>Parameters</td>
<td>383</td>
</tr>
<tr>
<td>Return value</td>
<td>385</td>
</tr>
<tr>
<td>Usage</td>
<td>385</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>386</td>
</tr>
<tr>
<td>Examples: @PickList</td>
<td>386</td>
</tr>
<tr>
<td>@Platform</td>
<td>386</td>
</tr>
<tr>
<td>Syntax</td>
<td>386</td>
</tr>
<tr>
<td>Parameters</td>
<td>386</td>
</tr>
<tr>
<td>Return value</td>
<td>386</td>
</tr>
<tr>
<td>Usage</td>
<td>387</td>
</tr>
<tr>
<td>Contents</td>
<td>xiii</td>
</tr>
<tr>
<td>@SetViewInfo</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Examples: @Select.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Return value</td>
</tr>
<tr>
<td>Examples: @ServerAccess</td>
<td>409</td>
</tr>
<tr>
<td>@ServerAccess</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>407</td>
</tr>
<tr>
<td>Return value</td>
<td>409</td>
</tr>
<tr>
<td>Examples: @ServerAccess</td>
<td>409</td>
</tr>
<tr>
<td>@ServerName</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Return value</td>
<td>409</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>410</td>
</tr>
<tr>
<td>Examples: @ServerName</td>
<td>410</td>
</tr>
<tr>
<td>@Set</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>410</td>
</tr>
<tr>
<td>Usage</td>
<td>410</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>410</td>
</tr>
<tr>
<td>Examples: @Set</td>
<td>410</td>
</tr>
<tr>
<td>@SetDocField</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>411</td>
</tr>
<tr>
<td>Usage</td>
<td>411</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>411</td>
</tr>
<tr>
<td>Examples: @SetDocField</td>
<td>411</td>
</tr>
<tr>
<td>@SetEnvironment</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>412</td>
</tr>
<tr>
<td>Usage</td>
<td>412</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>413</td>
</tr>
<tr>
<td>Examples: @SetEnvironment</td>
<td>413</td>
</tr>
<tr>
<td>@SetField</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>413</td>
</tr>
<tr>
<td>Usage</td>
<td>413</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>413</td>
</tr>
<tr>
<td>Examples: @SetField</td>
<td>413</td>
</tr>
<tr>
<td>@SetHTTPHeader</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>414</td>
</tr>
<tr>
<td>Return value</td>
<td>414</td>
</tr>
<tr>
<td>Usage</td>
<td>414</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>414</td>
</tr>
<tr>
<td>Examples: @SetHTTPHeader</td>
<td>414</td>
</tr>
<tr>
<td>@SetProfileField</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>415</td>
</tr>
<tr>
<td>Return value</td>
<td>415</td>
</tr>
<tr>
<td>Usage</td>
<td>415</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>415</td>
</tr>
<tr>
<td>Examples: @SetProfileField</td>
<td>415</td>
</tr>
<tr>
<td>@SetTargetFrame</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>416</td>
</tr>
<tr>
<td>Usage</td>
<td>416</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>416</td>
</tr>
<tr>
<td>Examples: @SetTargetFrame</td>
<td>416</td>
</tr>
<tr>
<td>@SetViewInfo</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>417</td>
</tr>
<tr>
<td>Usage</td>
<td>417</td>
</tr>
<tr>
<td>Examples: @SetViewInfo.</td>
<td>418</td>
</tr>
<tr>
<td>@Sign</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>418</td>
</tr>
<tr>
<td>Return value</td>
<td>418</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>418</td>
</tr>
<tr>
<td>Examples: @Sign</td>
<td>419</td>
</tr>
<tr>
<td>@Sin</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>419</td>
</tr>
<tr>
<td>Return value</td>
<td>419</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>419</td>
</tr>
<tr>
<td>Examples: @Sin</td>
<td>419</td>
</tr>
<tr>
<td>@Sort</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>420</td>
</tr>
<tr>
<td>Return value</td>
<td>420</td>
</tr>
<tr>
<td>Usage</td>
<td>420</td>
</tr>
<tr>
<td>Examples: @Sort</td>
<td>421</td>
</tr>
<tr>
<td>@Soundex</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>422</td>
</tr>
<tr>
<td>Return value</td>
<td>422</td>
</tr>
<tr>
<td>Usage</td>
<td>422</td>
</tr>
<tr>
<td>Examples: @Soundex</td>
<td>422</td>
</tr>
<tr>
<td>@Sqrt</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>422</td>
</tr>
<tr>
<td>Return value</td>
<td>422</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>422</td>
</tr>
<tr>
<td>Examples: @Sqrt</td>
<td>422</td>
</tr>
<tr>
<td>@StatusBar</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>423</td>
</tr>
<tr>
<td>Return value</td>
<td>423</td>
</tr>
<tr>
<td>Usage</td>
<td>423</td>
</tr>
<tr>
<td>Examples: @StatusBar</td>
<td>423</td>
</tr>
<tr>
<td>@Subset</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>423</td>
</tr>
<tr>
<td>Return value</td>
<td>423</td>
</tr>
<tr>
<td>Usage</td>
<td>423</td>
</tr>
<tr>
<td>Examples: @Subset</td>
<td>423</td>
</tr>
<tr>
<td>@Success</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>424</td>
</tr>
<tr>
<td>Return value</td>
<td>424</td>
</tr>
<tr>
<td>Usage</td>
<td>424</td>
</tr>
<tr>
<td>Examples: @Success</td>
<td>424</td>
</tr>
<tr>
<td>@Sum</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>424</td>
</tr>
<tr>
<td>Return value</td>
<td>424</td>
</tr>
<tr>
<td>Usage</td>
<td>424</td>
</tr>
<tr>
<td>Examples: @Sum</td>
<td>425</td>
</tr>
<tr>
<td>@Tan</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>425</td>
</tr>
<tr>
<td>Return value</td>
<td>425</td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>425</td>
</tr>
<tr>
<td>Examples: @Tan</td>
<td>425</td>
</tr>
<tr>
<td>@TemplateVersion</td>
<td>Syntax.</td>
</tr>
<tr>
<td>Parameters</td>
<td>426</td>
</tr>
<tr>
<td>Return value</td>
<td>426</td>
</tr>
<tr>
<td>Usage</td>
<td>426</td>
</tr>
</tbody>
</table>
@UpperCase
Parameters
Return value
Usage
Language cross-reference
Examples: @UpperCase

@LowerCase
Parameters
Return value
Usage
Language cross-reference
Examples: @LowerCase

Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @Parameter

Examples:

Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @UserNamesList

@UserPrivileges
Syntax
Return value
Usage
Language cross-reference
Examples: @UserPrivileges

@UserRoles
Syntax
Return value
Usage
Language cross-reference
Examples: @UserRoles

@V3UserName
Syntax
Return value
Usage
Language cross-reference
Examples: @V3UserName

@V4UserAccess
Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @V4UserAccess

@ValidateInternetAddress
Syntax
Parameters
Return value
Usage
Possible error messages
Language cross-reference
Examples: @ValidateInternetAddress

@VerifyPassword
Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @VerifyPassword

@Version
Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @VerifyPassword

@ViewShowThisUnread
Syntax
Parameters
Return value
Usage
Language cross-reference
Examples: @ViewShowThisUnread

@ViewTitle
Parameters
Return value
Usage
Language cross-reference
Examples: @ViewTitle

Contents  xvii
### Chapter 7. Formula Language

@Commands A-Z. 475

Using @Commands 475
  Syntax 475
  Parameters 475
  Return value 475
@Commands with ECL security 475
AddBookmark @Command 477
  Syntax 477
  Parameters 477
  Usage 478
  Examples: AddBookmark 478
AddDatabase @Command 478
  Syntax 478
  Parameters 478
  Usage 479
  Examples: AddDatabase 479
AddDatabaseRepID @Command 479
  Syntax 479
  Parameters 479
  Usage 479
AddToIMContactList @Command 479
  Syntax 479
  Parameters 480
AdminCertify @Command 480
  Syntax 480
  Usage 480
  Examples: AdminCertify 480
AdminCreateGroup @Command 480
  Syntax 480
  Parameters 480
  Usage 480
  Examples: AdminCreateGroup 480
AdminCrossCertifyIDFile @Command 480
  Syntax 480
  Usage 480
  Language cross-reference 480
AdminCrossCertifyKey @Command 481
  Syntax 481
  Usage 481
  Examples: AdminCrossCertifyKey 481
AdminDatabaseAnalysis @Command 481
  Syntax 481
  Usage 481
  Examples: AdminDatabaseAnalysis 481
AdminDatabaseQuotas @Command 481
  Syntax 481
  Usage 481
  Language cross-reference 482
AdminDFileClearPassword @Command 482
  Syntax 482
  Usage 482
AdminDFileExamine @Command 482
  Syntax 482
  Usage 482
AdminDFileSetPassword @Command 482
  Syntax 482
  Usage 482
Administration @Command 482
  Syntax 482
  Usage 482

Language cross-reference 473
Examples: @Zone 474

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Return value</th>
<th>Usage</th>
<th>Language cross-reference</th>
<th>Examples: @ViewTitle</th>
</tr>
</thead>
<tbody>
<tr>
<td>@WebDbName</td>
<td>467</td>
<td>467</td>
<td>467</td>
<td>467</td>
</tr>
<tr>
<td>Syntax</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
<td>Examples: @WebDbName</td>
</tr>
<tr>
<td>@Weekday</td>
<td>467</td>
<td>467</td>
<td>467</td>
<td>467</td>
</tr>
<tr>
<td>Syntax</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
<td>Examples: @Weekday</td>
</tr>
<tr>
<td>@While</td>
<td>468</td>
<td>468</td>
<td>468</td>
<td>468</td>
</tr>
<tr>
<td>Syntax</td>
<td>Parameters</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
</tr>
<tr>
<td>@Word</td>
<td>469</td>
<td>469</td>
<td>469</td>
<td>469</td>
</tr>
<tr>
<td>Syntax</td>
<td>Parameters</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
</tr>
<tr>
<td>@Year</td>
<td>470</td>
<td>470</td>
<td>470</td>
<td>470</td>
</tr>
<tr>
<td>Syntax</td>
<td>Parameters</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
</tr>
<tr>
<td>@Yes</td>
<td>471</td>
<td>471</td>
<td>471</td>
<td>471</td>
</tr>
<tr>
<td>Syntax</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
<td>Examples: @Yes</td>
</tr>
<tr>
<td>@Yesterday</td>
<td>472</td>
<td>472</td>
<td>472</td>
<td>472</td>
</tr>
<tr>
<td>Syntax</td>
<td>Return value</td>
<td>Usage</td>
<td>Language cross-reference</td>
<td>Examples: @Yesterday</td>
</tr>
<tr>
<td>@Zone</td>
<td>472</td>
<td>472</td>
<td>472</td>
<td>472</td>
</tr>
<tr>
<td>Syntax</td>
<td>Parameters</td>
<td>Return value</td>
<td>Usage</td>
<td>Time zones that are not full-hour increments from GMT</td>
</tr>
<tr>
<td>Syntax</td>
<td>Page</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailForwardAsAttachment @Command</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailOpen @Command</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>564</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailRequestCrossCert @Command</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailRequestNewName @Command</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailRequestNewPublicKey @Command</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailScanUnread @Command</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailSend @Command</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailSendCertificateRequest @Command</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>566</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailSendEncryptionKey @Command</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailSendPublicKey @Command</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MoveToTrash @Command</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples: MoveToTrash</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigateNext @Command</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples: NavigateNext</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigateNextHighlight @Command</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigateNextMain @Command</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigateNextSelected @Command</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigateNextUnread @Command</td>
<td>570</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>570</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>570</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigatePrev @Command</td>
<td>570</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>570</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>570</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigatorToBacklink @Command</td>
<td>572</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>572</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>572</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigatorProperties @Command</td>
<td>572</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>572</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavigatorTest @Command</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavNext @Command</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavNextMain @Command</td>
<td>574</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>574</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>574</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>574</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavNextUnread @Command</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavPrev @Command</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavPrevMain @Command</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavPrevSelected @Command</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NavPrevUnread @Command</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ObjectDisplayAs @Command</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewChange @Command</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples: ViewChange</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewCollapse @Command</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewCollapseAll @Command</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples: @Command[(ViewCollapseAll)]</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewExpand @Command</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewExpandAll @Command</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>623</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewExpandWithChildren @Command</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewHorizScrollBar @Command</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewMoveName @Command</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewNavigatorsFolders @Command</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewNavigatorsNone @Command</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewRefreshFields @Command</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewRefreshUnread @Command</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewRenamePerson @Command</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewShowFieldHelp @Command</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewShowObject @Command</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewShowOnlyCategories @Command</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewShowOnlySearchResults @Command</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax.</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage.</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language cross-reference</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViewShowOnlySelected @Command</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Syntax .......................... 628
Usage .......................... 628
ViewShowOnlyUnread @Command .. 628
Syntax .......................... 628
Usage .......................... 628
ViewShowPageBreaks @Command ... 628
Syntax .......................... 628
Usage .......................... 628
ViewShowServerNames @Command .. 629
Syntax .......................... 629
Usage .......................... 629
ViewShowRuler @Command .......... 629
Syntax .......................... 629
Usage .......................... 629
ViewShowSearchBar @Command ...  629
Parameters ...................... 629
Usage .......................... 629
ViewShowServerNames @Command .. 629
Syntax .......................... 629
Usage .......................... 629
ViewShowUnread @Command ........ 629
Syntax .......................... 629
Usage .......................... 629
ViewSwitchForm @Command ....... 629
Syntax .......................... 629
Parameters ...................... 629
Usage .......................... 629
WindowCascade @Command .......  630
Syntax .......................... 630
Usage .......................... 630
WindowMaximize @Command ....... 630
Syntax .......................... 630
Usage .......................... 630
WindowMaximizeAll @Command ... 630
Syntax .......................... 630
Usage .......................... 630
WindowMinimize @Command ........ 630
Syntax .......................... 630
Usage .......................... 630
WindowMinimizeAll @Command .... 631
Syntax .......................... 631
Usage .......................... 631
WindowNext @Command ........... 631
Syntax .......................... 631
Usage .......................... 631
WindowRestore @Command .......... 631
Syntax .......................... 631
Usage .......................... 631
WindowTile @Command ............ 631
Syntax .......................... 631
Usage .......................... 631
WindowWorkspace @Command ....... 632
Syntax .......................... 632
Usage .......................... 632
WorkspaceProperties @Command ... 632
Syntax .......................... 632
Usage .......................... 632
WorkspaceStackReplicaIcons @Command .. 632
Syntax .......................... 632
Usage .......................... 632
ZoomPreview @Command ........... 632
Syntax .......................... 632
Parameters ...................... 632
Usage .......................... 633

Index .......................... 635
Chapter 1. Programming Overview

This documentation describes how to attach Java(TM), JavaScript(TM), LotusScript(R), and formula code to Lotus(R) Domino(TM) design elements. Here are some overview topics:

- Programming in IBM Lotus Domino Designer(R)
- Where to use scripts and formulas
- Table of programmable design elements
- Event descriptions
- Event sequencing

Programming in IBM Lotus Domino Designer

Formula, LotusScript, Java, and JavaScript code provide an integral programming interface to Lotus Domino Designer. You attach code to various design elements depending on need. For example, if you create a computed field on a form, you would attach a formula to compute the value of the field. Or you could attach JavaScript code to the onFocus event of a field; this code would execute whenever a user places focus on the field. Or you might decide to create a formula, LotusScript, or Java agent to automatically update all the documents in a database at scheduled times.

Lotus Domino Designer provides a programming interface to development environments that support COM and OLE. Lotus Domino Designer provides a programming interface for Java applications and applets. Java applications and applets can operate locally by accessing installed Domino software or remotely by connecting to a Domino server using CORBA with IIOP protocols.

The Overview - applications and databases and database management sections of the Application Development with Domino Designer book provide guidelines, templates, and examples for the common cases where you use code. For these cases, you do not have to learn how to program. Beyond these cases, refer to this manual for complete guidelines and reference material for programming in Lotus Domino Designer.

Where to use scripts and formulas

Before you write code, make sure that a simple action won’t do the task. You can design some elements with simple actions in forms or views that do not require programming.

Where you have a choice among programming interfaces, consider these guidelines:

- Formulas are expressions that have program-like attributes. For example, you can assign values to variables and use a limited control logic. The formula language interface to Lotus Domino Designer is through calls to @functions and @commands.

  In general, formulas are best used for working within the object that the user is currently processing, for example, to return a default value to a field or determine selection criteria for a view. Additionally, formulas provide better performance in some situations and may be more convenient for simple applications. You can also execute looping logic using functions such as @For, @While, and @DoWhile.

  Note: Looping is new with Release 6.

- JavaScript is a cross-platform, object-oriented scripting language. Header scripts may be written in the Programmer’s pane by selecting JS Header from the Objects tab and typing your script in the Script
Area. Scripts may also be attached to specific events such as onClick, or to objects such as buttons. JavaScript may not be written in an agent. Lotus Domino Designer oversees the compilation and loading of user scripts, but does not store JavaScript in compiled form. In LotusScript Web agents, you can use Print statements to output HTML to the browser, including script tags. For example, Print”<script>function changeLocation()...</script>.”

JavaScript is best used for Web applications, or when a single application will be used in both the Lotus Notes(R) and Web environments. Version 1.4 of the JavaScript language is supported.

- LotusScript is a full object-oriented programming language. Its interface to Lotus Domino is through predefined classes. Lotus Domino oversees the compilation and loading of user code, and automatically includes the Domino class definitions.

LotusScript is best used where the programming logic is not simple. It excels in accessing stored databases (back-end) and provides some capabilities that formulas do not, such as the ability to manipulate a database access control list (ACL). LotusScript is limited in its UI (front-end) capabilities.

- Java is a full object-oriented programming language. Its interface to Lotus Domino is through predefined classes. It is comparable to LotusScript in agents but cannot be attached to events in the Domino UI. Lotus Domino oversees the compilation and loading of user code for agents; the code can be written natively or imported.

Java can be used in agents. Java applications and applets, written and compiled outside of Lotus Domino, can access Lotus Domino through the class interface.

### Table of programmable design elements

The following table outlines the programmable design elements in Domino. The table specifies the scope of the design element and whether it supports simple actions, formulas, LotusScript, Java, or JavaScript.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Design element</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Replication formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Agent</td>
<td>Formula Simple action LotusScript Java</td>
</tr>
<tr>
<td></td>
<td>Event</td>
<td>LotusScript Formula</td>
</tr>
<tr>
<td>Document (Edit mode) Rich text field</td>
<td>Section title formula</td>
<td>Formula Text</td>
</tr>
<tr>
<td></td>
<td>Hidden paragraph formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Hotspot (button or action)</td>
<td>Formula Simple action LotusScript JavaScript</td>
</tr>
<tr>
<td></td>
<td>Hotspot (link or formula pop-up)</td>
<td>Formula</td>
</tr>
<tr>
<td>Embedded outline design</td>
<td>Named element formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Background image formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Twisties image formula</td>
<td>Formula</td>
</tr>
<tr>
<td>Field design on forms</td>
<td>Default value formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Input translation formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Input validation formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Input enabled formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>HTML attributes formula</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Value formula for a computed field</td>
<td>Formula</td>
</tr>
<tr>
<td></td>
<td>Keyword field formula</td>
<td>Formula</td>
</tr>
<tr>
<td>Form or page design</td>
<td>Event</td>
<td>LotusScript JavaScript</td>
</tr>
<tr>
<td></td>
<td>Window title formula</td>
<td>Formula</td>
</tr>
</tbody>
</table>
## Toolbars

Toolbars provide a quick way to trigger commonly performed tasks, such as printing a document or refreshing a view. A Toolbar button works by executing a formula when you select it from a toolbar. You can customize the toolbars provided with Domino Designer or create your own.

For more details on editing existing toolbars, see Toolbars: smart shortcuts in Lotus Notes Help.

**Note:** Toolbar buttons were known as SmartIcon buttons prior to Release 6.
To create a new toolbar button

1. Choose File - Preferences - Toolbar Preferences from the menu bar. The Toolbar Preferences dialog box displays.

    Tip: You can also right-click a toolbar and select Toolbar Preferences from the pop-up menu.

2. Select the Toolbars tab.

3. To either:
   • Edit an existing toolbar, select it from the Available Toolbars list.
   • Create a new toolbar, click New Toolbar, enter a name in the text box and press ENTER.

4. Select the Customize tab.

   The name of the toolbar you are editing or creating displays in the Toolbar to modify field.

5. Add predefined Notes buttons to the toolbar by either:
   • Dragging and dropping buttons from the Available Buttons list to the Toolbar Contents pane.
   • Selecting a button (or buttons using the CNTRL key while selecting) and clicking Add to toolbar.

    Tip: You can change the display of the Available Buttons list to list them alphabetically by clicking Sort Buttons and selecting by Description.

6. To create and add a new button, click New and select Button from the drop-down list. The Edit Toolbar Button dialog box displays.

7. Enter a name for the button in the Button Text field and enter a description of the task the button performs in the Pop Up Help Text field.

   This text appears when a user mouses over the button.

8. Add the formula that this button triggers in the Formula box.

   Click the Fields & Functions button to see a list of the fields, @functions, and @commands available for you to reference in the formula.

   Click the Formula Window button if you want a larger window in which to write and edit your formula.

9. Click the Change Icon button and select an image from the Insert Image Resource dialog box.

    Tip: Supply an icon so that the button is visible to users. The default display style, which you can change on the Basics tab of the Toolbar Preferences dialog box, is Icon only. If this is selected, only icons display on the toolbar; button text does not.

10. Click Save Toolbar, then click OK to close the Toolbar Preferences dialog box.

Toolbars are associated with each user’s workspace; they are not associated with databases and are not shared among users. Toolbars are also specific to each client. A toolbar you customize in Designer does not display in the Notes client.

A Toolbar button’s formula runs on the user’s workstation.

Examples: Toolbars

1. This toolbar button formula opens the names.nsf database on the doc server to the People view:

   @Command([FileOpenDatabase]; "DOC": "NAME.NSF"; "People")

2. This toolbar button formula presents the user with the list of databases in a database catalog, and opens the database that the user selects. The first @DbColumn puts a list of the values in column 4 of the Databases by _Replica ID view in the temporary variable titles. The second @DbColumn puts a list of the values in column 2 of the Databases by _Replica ID view in the temporary variable servers. The third @DbColumn puts a list of the values in column 3 of the Databases by _Replica ID view in the temporary variable databases. The temporary variable list combines titles, servers, and databases.
for presentation to the user in @Prompt. The formula then parses the return value from @Prompt into a server name and database name for inclusion in the FileOpenDatabase @command.

titles := @DbColumn("": "doc":"CATALOG.NSF"; "Databases by _Replica ID"; 4);
servers := @DbColumn("": "doc":"CATALOG.NSF"; "Databases by _Replica ID"; 2);
databases := @DbColumn("": "doc":"CATALOG.NSF"; "Databases by _Replica ID"; 3);
list := titles + " *:* " + servers + " *:* " + databases;
member := @Prompt([OKCANCELLIST]; "Open Database"; "Select a database"; ";"; list);
server := @Left(@Right(member; " *:* "); " *:* ");
database := @Right(member; " *:* ");
@Command([FileOpenDatabase]; server;database)

Replication formulas

A replication formula selects the documents that are pulled into the current database during replication.

A replication formula must end with a SELECT statement. If the last statement in the formula is a logical expression, Domino turns it into a SELECT statement by inserting the reserved word SELECT.

@Commands are not allowed.

A replication formula runs on the server or workstation containing the formula’s database.

Examples: Replication formulas

1. The default replication formula for a database replicates all documents.
   SELECT @All

2. This formula replicates only documents containing East in the Region field.
   SELECT Region = "East"

3. This formula, which would be useful in a replica of a mail database, does not replicate documents whose From field is Arnold Runion or Mary Chen.
   SELECT !(From="Arnold Runion" | From="Mary Chen")

Agents

An agent is a user procedure that you can trigger through a number of mechanisms. An agent runs on:

• The user’s workstation if the agent’s trigger is “Action menu selection,” "Agent list selection,” or "When documents are pasted."

• The server or workstation containing the agent if the agent’s trigger is “Before new mail arrives,” "After new mail has arrived,” or "After documents are created or modified,” or any of the "On schedule" options.

You can code agents in the formula language, LotusScript, and Java, and you can use Domino-supplied agents. Before writing your own agent, see if a Domino-supplied agent will do the job.

Formula-based agents run iteratively on the documents in the database, unless you select None in the agent’s target option. You can also apply search criteria through the agent interface to specify which documents in the database are to be processed. A SELECT statement in the formula further limits the search. If you do not include a SELECT statement in the formula, Domino Designer appends a "SELECT @All" statement. Except for "SELECT @All," a SELECT statement must be the first statement in the formula to be effective.

@Commands are only allowed in agents that specify None in the target option.
LotusScript and Java agents run once. You supply the search criteria and the iteration through the language constructs. Search criteria applied through the agent interface are effective only through UnprocessedDocuments in NotesDatabase (LotusScript) or UnprocessedDocuments in AgentContext (Java).

**Examples: Agents**

1. This LotusScript code writes a value to the Category field based on the value of the TotalSales field of each document in the database. Compare with Examples 2 and 3, which use Java and formula solutions. The script example requires more lines of code than the formula solution but includes the algorithm for finding the documents being processed.

   ```lws
   Sub Initialize
     Dim session As New NotesSession
     Dim db As NotesDatabase
     Dim dc As NotesDocumentCollection
     Dim doc As NotesDocument
     Set db = session.CurrentDatabase
     Set dc = db.AllDocuments
     Set doc = dc.GetFirstDocument
     While Not(doc Is Nothing)
       category = doc.Category
       totalSales = doc.TotalSales
       Select Case totalSales(0)
         Case Is >= 200000: category(0) = "Above Quota"
         Case Is >= 100000: category(0) = "OK"
         Case Else: category(0) = "Below Quota"
       End Select
       doc.Category = category
       Call doc.Save(True, False)
       Set doc = dc.GetNextDocument(doc)
     Wend
   End Sub
   ```

2. This Java agent writes a value to the Category field based on the value of the TotalSales field of each document in the database. Compare with Examples 1 and 3, which use LotusScript and formula solutions. As with LotusScript, the Java code includes the algorithm for finding the documents being processed.

   ```java
   import lotus.domino.*;
   public class JavaAgent extends AgentBase {
     public void NotesMain() {
       try {
         Session session = getSession();
         AgentContext agentContext = session.getAgentContext();
         // (Your code goes here)
         Database db = agentContext.getCurrentDatabase();
         DocumentCollection dc = db.getAllDocuments();
         Document doc = dc.getFirstDocument();
         while (doc != null) {
           double totalSales = doc.getItemValueDouble("TotalSales");
           if (totalSales >= 200000)
             doc.replaceItemValue("Category", "Above quota");
           else if (totalSales >= 100000)
             doc.replaceItemValue("Category", "OK");
           else
             doc.replaceItemValue("Category", "Below quota");
           doc.save(true, false);
           doc = dc.getNextDocument();
         }
       } catch(Exception e) {
         e.printStackTrace();
       }
     }
   }
   ```
3. This formula writes a value to the Category field based on the value of the TotalSales field of each document in the database, assuming that all documents are selected for processing. Compare with Examples 1 and 2, which use LotusScript and Java solutions. The formula executes once on each document selected by combining outside criteria with the SELECT statement. Data declarations are implicit and formula syntax is cryptic making for compact source code.

```
FIELD Category := @If(TotalSales >= 200000; "Above Quota"; TotalSales >= 100000; "OK"; "Below Quota");
SELECT @All
```

4. This formula writes a value to the Category of selected documents based on the value of the TotalSales field. The SELECT statement, if it is not SELECT @All, must precede the statements in the formula that it applies to.

```
SELECT TotalSales >= 200000;
FIELD Category := "Above Quota"
```

5. This LotusScript code finds the sum of all OrderTotal fields in a database for one day, and writes a new record to the database containing the daily total. Each record in the database has OrderNumber, Date, and OrderTotal fields. The script finds all the documents in the database, then uses a loop and a comparison of dates to limit processing to today’s documents. For each document, the script adds the OrderTotal to a dailyTotal variable. The script places the words “DAILY TOTAL” in the OrderNumber field for the document that it writes, and places the dailyTotal value in the OrderTotal field.

```
Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim dc As NotesDocumentCollection
    Dim doc As NotesDocument
    Dim dateLimit As New NotesDateTime("Today")
    While Not(db = session.CurrentDatabase)
        Dim doc As NotesDocument
        Dim dateLimit As New NotesDateTime("Today")
        Set doc = session.GetFirstDocument(db)
        Dim docNew As New NotesDocument(db)
        Set itm = docNew.AppendItemValue("OrderNumber", "DAILY TOTAL")
        Set itm = docNew.AppendItemValue("OrderTotal", dailyTotal)
        Set itm = docNew.AppendItemValue("Date", Date$)
        Call docNew.Save(True, False)
    Wend
End Sub
```

### Actions

An action is a formula, LotusScript, JavaScript, or simple (no programming) procedure that you can associate with a view, form, subform, or page. You can create a standard action, an action with subactions, a shared action, or insert system actions that are supplied with Notes.

If you create a standard action, when you open the view or page, or open a document based on the form, the action becomes available as a menu command under Actions and/or as a button on an action bar. If you create an action with subactions, the parent action is not programmable but appears as a button with...
a drop-down arrow on the action bar. Each programmable subaction displays in a drop-down list when
the parent action is clicked. If you select Actions from the menu, the parent action appears as a menu
command and when given focus, displays a list of the available subactions.

Note: Subactions are new with Release 6.

You can insert the standard system actions, which include action buttons that perform basic functions
such as forwarding, editing, or sending a document. Additionally, you can create and insert a shared
action that you can reuse in several different design elements. This enables you to make any updates or
enhancements to the action’s code in one place and it is automatically updated in all the elements that
use it.

You can conditionally suppress availability of an action on the menu or action bar with a "Hide action if
formula is true" formula. @Commands are not allowed in hide formulas.

Actions run on the user's workstation.

Examples: Actions

1. This LotusScript action prints the name of each Domino database in the Domino data directory on the
computer running the script. The FirstDatabase and NextDatabase methods of NotesDbDirectory walk
through all the databases for the specified server where null defaults to the current computer.

Sub Click(Source As Button)
    Dim directory As New NotesDbDirectory(""
    Dim db As NotesDatabase
    Set db = directory.GetFirstDatabase(DATABASE)
    While Not(db Is Nothing)
        MessageBox db.Title
        Set db = directory.GetNextDatabase()
    Wend
    MessageBox "The End"
End Sub

2. This formula lists the names in the Address Book on the CORP1 server, lets the user select any
number of names, combines the selected names into a string using a comma and a space to separate
names, and inserts the string into the current field. This action works best when the user is in the
SendTo field of a mail database.

    last := @Left(@DbColumn(""; "CORP1" : "NAMES.NSF"; "People"; 1); ",
    first := @RightBack(@DbColumn(""; "CORP1" : "NAMES.NSF"; "People"; 2); ",
    list := first + "," + last;
    name := @Prompt([OKCANCEL|LISTMULT]; "Send To"; "Who are you sending this memo to?"; ","; list);
    @Command([EditInsertText]; @implode(name; ","))

3. This formula is a 'Hide action if formula is true" formula. The action becomes available on the form
menu or action bar only if the field OrderTotal is 100 or less. (If the user just entered a value for
Order_Total in the current document, a document refresh must occur before the new value is
effective.)

    OrderTotal > 100

4. These scripts collectively force the user to use an action to place an existing document in Edit mode.
The action places the current document in Edit mode. The Postopen and Querychangemode
event scripts prevent the user from changing to Edit mode through other means such as Actions -
Edit Document (CTRL+E).

    '(Globals) object, (Declarations) event
    Dim allowEdit As Integer
    '(Form) object, Postopen event
    Sub Postopen(Source As Notesuidocument)
        '(Let document pass if new or not in EditMode
        'Otherwise if existing document is in EditMode
        'Set allowEdit so Querymodechange doesn't reprocess
        'TurnEditMode off so document opens in Read mode

8 Programming Guide, Volume 1: Overview and Formula Language
' Tell the user to use the action
If source.EditorMode And Not source.IsNewDoc Then
    allowEdit = True
    source.EditorMode = False
    MessageBox_ "Use Edit mode action to edit document"
Else
    allowEdit = False
End If
End Sub

'(Form) object, Querymodechange event
Sub Querymodechange(Source As NotesUIDocument, Continue As Integer)
    'Allow user to proceed, and turn off allowEdit if
    'user clicked the action (allowEdit on)
    'already processed by Postopen (allowEdit on)
    'trying to get out of Edit mode
    '(allowEdit off but EditMode on)
    'Tell user to click action if changing existing document
    'to Edit mode and not already processed by Postopen
    '(allowEdit and EditMode off)
    If allowEdit Or (source.EditorMode And Not allowEdit) Then
        allowEdit = False
    Else
        MessageBox_ "Use Edit mode action to edit document"
        continue = False
    End If
End Sub

'(Action) object, Click event
Sub Click(Source As Button)
    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Set uidoc = workspace.CurrentDocument
    'Turn on allowEdit so Querymodechange will let it pass
    'Turn on EditMode
    allowEdit = True
    uidoc.EditorMode = True
End Sub

Hotspots

In form design, subform design, page design, navigator design, layout region design, and rich text fields, you can create hotspots. A hotspot activates when the user selects it and can be any of several types:

- A link hotspot jumps to another object. As the designer, you supply the link.
- A text pop-up displays text. As the designer, you supply literal text.
- A button performs an action. As the designer, you supply a simple action, or formula, LotusScript, or JavaScript code.
- A formula pop-up displays text based on the result of a formula. As the designer, you supply a formula.
- An action hotspot performs an action. As the designer, you supply a simple action, or formula, LotusScript, or JavaScript code.

Buttons and action hotspots are the same except that a button appears as a button and an action hotspot appears as highlighted text.

@Commands are not allowed in formula pop-up hotspots.

A hotspot runs on the user’s workstation.
Examples: Hotspots

1. This JavaScript code shows an alert when activated by the onClick event of a button or action hotspot.
   ```javascript
   alert("This is the alert box.")
   ```

2. This JavaScript code prompts the user for his or her name when activated by the onClick event of a button or action hotspot.
   ```javascript
   var message
   var defaultanswer
   message = "Please enter your name:"
   defaultanswer = myForm.from.value;
   result = prompt(message, defaultanswer);
   if(result) {
      myForm.results.value = result;
   }
   ```

3. This JavaScript example could be attached to the onClick event for a button or action hotspot. It determines how a mail message is delivered. For "bugs," the mail is directed to "SprManager." Other requests are directed to "WishListManager." The final line in this script launches mail.
   ```javascript
   var isBug = confirm("Are you reporting a bug?");
   var recipient = isBug ? "SprManager@xyz.com" : "WishListManager@xyz.com"
   document.location = "mailto:" + recipient;
   ```

4. This LotusScript button or action hotspot is useful in a mail message to a group of persons. When the recipient clicks the hotspot, the script sends a mail message indicating "Yes" or "No" for the "RSVP." For testing, the name of the current user is used -- in practice, you would use your name. Compare with the next example, which uses a formula solution.
   ```lotusscript
   'declarations
   Const MB_YESNO = 4
   Const MB_ICONQUESTION = 32
   Const IDYES = 6
   '
   'click event
   Sub Click(Source As Button)
      Dim session As New NotesSession
      Dim db As NotesDatabase
      Dim doc As NotesDocument
      Set db = session.CurrentDatabase
      Set doc = New NotesDocument(db)
      yesno = MessageBox("Do you want to attend?", MB_YESNO+MB_ICONQUESTION)
      If yesno = IDYES Then rsvp = "Yes" Else rsvp = "No"
      Call doc.AppendItemValue("Subject", "RSVP")
      Call doc.AppendItemValue("Body", rsvp)
      Call doc.Send(True, session.UserName)
   End Sub
   ```

5. This formula button or action hotspot is useful in a mail message to a group of persons. When the recipient clicks the hotspot, the formula sends a mail message indicating "Yes" or "No" for the "RSVP." For testing, the name of the current user is used -- in practice, you can use your name. Compare with the previous example, which uses a LotusScript solution.
   ```lotusscript
   yesno := @Prompt([YESNO]; "RSVP"; "Do you want to attend?");
   rsvp := @If(yesno; "Yes"; "No");
   @MailSend(@UserName; ""; MB_MAILSEND; "RSVP"; ""; rsvp; "")
   ```

6. This LotusScript button or action hotspot displays the sum of all OrderTotal fields in a database for one day. Each record in the database has OrderNumber, Date, and OrderTotal fields. The script finds all the documents in the database, then uses a loop and a comparison of dates to limit processing to today's documents. For each document, the script adds the OrderTotal to a dailyTotal variable.
   ```lotusscript
   Sub Click(Source As Button)
      Dim session As New NotesSession
      Dim db As NotesDatabase
      Dim dc As NotesDocumentCollection
      Set db = session.CurrentDatabase
      Set dc = db.AllDocuments
   ```
dailyTotal = 0
Set doc = dc.getFirstDocument
While Not(doc Is Nothing)
   odate = doc.Date
   dn = Datenumber(Year(odate(0)), Month(odate(0)), Day(odate(0)))
   orderNumber = doc.OrderNumber
   If dn = Today And orderNumber(0) <> "DAILY TOTAL" Then
      orderTotal = doc.OrderTotal
      dailyTotal = dailyTotal + orderTotal(0)
   End If
   Set doc = dc.getNextDocument(doc)
Wend
MessageBox Format(dailyTotal, "Currency")
End Sub

7. This action hotspot is part of a navigator. The formula prompts the user and opens a view selected by the user.

   view := @Prompt([OKCANCELLIST]; "Which view do you want to open?"; ""; ""; "By first name"; "By last name");
   @Command([OpenView]; view)

8. This formula shows "pop-up text from a formula." You can highlight as a hotspot the following text on a form: "Please note your user name and the current date and time." The formula associated with the hotspot would evaluate to a text string containing the necessary information.

   "Your user name is " + @UserName + @NewLine + "The date and time are " + @Text(@Now)

**Form, selection, and column formulas**

As part of the view design process, you specify **form**, **selection** and **column formulas**

@Commands are not allowed.

These formulas run on the user's workstation.

**Form formulas**

A form formula determines which forms are used for composing and displaying documents under different conditions.

A form formula is optional. Lotus Domino Designer selects a form in the following preference order:

1. Form stored in the document.
2. Form specified in the form formula.
3. Form with which the document was created.
4. Default form for the database.

A form formula must evaluate to the name of a form. To enter a form formula:

1. Open the view in Lotus Domino Designer.
2. Select the Form Formula object on the Objects tab.
3. Enter a formula in the Script area.

**Form formulas and creating new documents in a view**

A form formula will override the form called by a view action which uses a formula or LotusScript to create a document.

For example, a view has the following form formula:

@If(@IsResponseDoc; "Response"; "MainTopic")
This form formula states that a document in the view will be displayed using the Response form if it is a response document, otherwise the MainTopic form will be used. However, if a user attempts to compose a new document using a different form such as "Phone Number" by selecting Create - Phone Number from the Notes menu, the user will see the MainTopic form instead of the Phone Number form. The same is true of view actions that use formulas or LotusScript to compose a new document.

To avoid this problem, add the following line to the form formula.
@If(@IsNewDoc; @Return(Form); "")

Examples: Form formulas
1. This formula creates new documents using the "Open New Discussion" form and accesses existing documents using the "Main Topic" form. The first line of the formula enables a user to create a document using a view action when that action calls for a form other than "Main Topic."
@If(@IsNewDoc; @Return(form); ");
@If(@IsNewDoc; "Open New Discussion"; "Main Topic")

2. This formula instructs Lotus Domino to open all documents selected from a view on the Web using a Web form and all documents selected from a view in Lotus Notes using a Notes form.
@If(@ClientType="Notes";"notesForm";"webForm")

Selection formulas
A selection formula selects the documents that appear in a view.
A selection formula must end with a SELECT statement. If the last statement in the formula is a logical expression, Lotus Domino Designer turns it into a SELECT statement by prepending the reserved word SELECT.

Examples: Selection formulas
1. The default selection formula for a database is:
   SELECT @ALL
2. This formula selects only documents based on the "Main" or "First response" form:
   SELECT Form="Main" | Form="First response"
3. These examples are from the mail template mail.ntf. The first example is the selection formula for the Drafts By Category view; the documents selected for this view are those where both the DeliveredDate and PostedDate are not filled in (are null). The second example is the selection formula for the Sent By Category; the documents selected for this view are those where the PostedDate is filled in, but the DeliveredDate is not. The third example is the selection formula for the Received By Category view; in this case, either the DeliveredDate or the Date field is filled in.
   SELECT DeliveredDate = "" & PostedDate = ""
   SELECT DeliveredDate = "" & PostedDate != ""
   SELECT DeliveredDate != "" | Date != ""

Column formulas
A column formula determines what is displayed in a column of a view. It must evaluate to a text string.

Examples: Column formulas
1. This formula specifies the column contents as the Subject field.
   Subject
2. This formula specifies the column contents as the Subject field followed by a blank (or nothing if the Subject field is empty), followed by the From field in parentheses. By default, the From field contains the name of the author of the document.
   DEFAULT From := @Author;
   @If(Subject != ""; Subject + " "; ") + "(" + From " ")"
3. This formula specifies the column contents as the number 1 for Weekdays forms and 2 for all other forms. If you make this column the first column, hidden, and sorted in ascending sequence, you force the Weekdays document (or documents) to the top of the view.

   @If(Form = "Weekdays"; 1; 2)

4. This formula reformats the contents of Person_Name to put the last name first followed by a comma, a space, and the first name.

   @If(@Contains(Person_Name; " "); @Right(Person_Name; " ") + ", " + @Left(Person_Name; " "); Person_Name)

5. This formula puts the name of the current weekday in the column, except that Saturdays and Sundays are treated as Fridays.

   T := @weekday(@Now);
   @If(T = 2; "Monday"; T = 3; "Tuesday"; T = 4; "Wednesday"; T = 5; "Thursday"; "Friday")

6. For a database that contains an "acl" Dialog list field that is set to Use Access Control List for choices and that has the following view and column properties settings:

   In the View Properties box:
   • Change the Lines per row to the number of carriage returns you want to include in the row (maximum available is 9).
   • Select Shrink rows to content.

   In the Column Properties box:
   • Choose New Line as the Multi-value separator.
   • Deselect the Show multiple values as separate entries check box.

   This formula populates the column with a maximum of nine entries from the access control list for the current database:

   acl

---

Window title, section access, and insert subform formulas

In form and page design, you specify window title, section access (form only), and insert subform (form only) formulas.

@Commands are not allowed.

These formulas run on the user’s workstation.

Window title formulas

A window title formula generates the text that appears in the title bar of documents using the form. The formula must evaluate to a text or numeric value, except that a single field of any type is permissible.

Rich text fields cannot be used in window title formulas.

Examples: Window title formulas

1. This formula displays the static text "Focus Group Discussion" in the title bar.

   "Focus Group Discussion"

2. This formula displays the static text "New Topic" for a new document. For an existing document, the formula displays the contents of the Subject field followed by text indicating the number of response documents associated with this document.

   @If(@IsNewDoc; "New Topic"; Subject + @DocDescendents(" (No Responses)"; "; "; 1; "; ( No Responses)"; "(1 Response)"; "( % Responses)")

3. This formula displays a window title suited to a Response or a Response to Response document.

   @If(@IsNewDoc; "New Response to " + Subject;
   "Response " + @DocNumber("") + " of " + @DocSiblings + " to " + Subject)

4. This formula, which evaluates to a time-date value, is legal.
5. In this formula, the time-date field must be explicitly converted to text because it is part of an expression.
"Response created on " + @Text(@Created)

Section access formulas
A section access formula specifies the names or ACL roles of users who can edit the fields in the section. Other users only have reader access to the section. However, this specification does not override the access control list for the database.

A section access formula must evaluate to a name, role, or a list of names or roles.

Examples: Section access formulas
1. This formula restricts edit access to a section to Mary Chen and Donna Hill.
"Mary Chen" ; "Donna Hill"
2. This formula restricts edit access to the section to those users listed in Column 1 of the By Person view of the current database.
@DbColumn("" ; "" ; By Person" ; 1)
3. This formula restricts edit access to the section to those users assigned the [EditStaff] role in the ACL.
"[EditStaff]"
4. This formula defines who can edit the section based on the content of the Status field on the form. If Status contains "Submitted," users with [Editors] or [Admins] user roles in the ACL can edit it. If Status contains "Draft," only the user whose name is in the OriginalAuthor field can edit it. Otherwise, anyone not in "-nobody-" displays in the "Who can edit this section" dialog box if a user attempts to edit it.
@if(Status = "Submitted"; "[Editors]";"[Admins]"; Status = "Draft"; OriginalAuthor; "-nobody-")

Insert subform formulas
An insert subform formula specifies the name of a subform for insertion into the form. The formula must evaluate to a text value that is the name of a subform. An invalid name results in a run-time error.

Examples: Insert subform formulas
This example uses the Business Address subform if the Categories field is "Business" and uses the Home Address subform otherwise.
@if(Categories = "Business"; "Business Address"; "Home Address")

Section title and hidden paragraph formulas
In form and page design and in rich text fields, you can use formulas to generate section titles and hide paragraphs.

@Commands are not allowed.

These formulas run on the user's workstation.

Section title formulas
The title for a collapsible section on a form, page, or rich text field can be generated from a formula. A section title formula must evaluate to a text or numeric value, except that a single field of any type is permissible.

Examples: Section title formulas
If Name has the value "Mary Chen," this formula prints "Mary Chen's personal information" as the section title.
Hidden paragraph formulas
A paragraph in an embedded outline, form, page, or rich text field can be hidden (not displayed) if an associated formula is true.

Examples: Hidden paragraph formulas
This formula hides a paragraph if the value of the Department field is not Sales.
Department != "Sales"

Hidden column and row custom color formulas
In view design, you can use formulas to hide columns and to customize row colors.

Hidden column formulas
A column in a view can be hidden (not displayed) if an associated formula is true.

Note: Hidden column formulas are new with Release 6.

For details, see the topic, "Using a hide-when formula in a column" in the Application Development with Domino Designer book.

Examples: Hidden column formulas
This formula allows the column to display only if the current user is a manager in the current database.
!@Contains(0DbManager; @UserName)

Row custom color formulas
Color can be applied to the text and background of a row in a view based on a formula. The formula is specified as the value for a "Use value as color" column in the view. The formula must evaluate to a list of three or six numbers:
• A 3-number value specifies the RGB value for the text of the row. The background color is unaffected.
• A 6-number value specifies the RGB value for the background (first three numbers) and text (second three numbers) of the row.

Note: Row custom color formulas are new with Release 6.

Examples: Row custom color formulas
This formula applies one set of colors (red on pink) to rows where the value of the Region field is "North" and another set of colors (black on white) to all other rows.
@1f(Region = "North"; 255:193:253:255:0:0;

Named element formulas
For frames, link hotspots, and outline entries, formulas can be used to generate the kind of named element (for example, Form, Page, or View), the name of the database containing the element, and the name of the element.

To enter a named element formula:
1. Select Named Element from the Content Type drop-down list in the Hotspot Link, Frame, or Outline Entry Properties box.
2. Click the Formula(@) button.
   The Named Element Properties box displays.
3. In the "Formula for kind of named element" box, enter either a formula that returns a named element type, or the type of named element you want to access (Form, View, etc.) in quotation marks. "Page" is the default named element type.

4. In the "Formula for database" box, enter either a formula that returns a database name, or the name of the database you want to access in quotation marks. Include the NSF extension.

5. In the "Formula for name of named element" box, enter either a formula that returns the name of a named element, or the name of a named element in quotation marks.

Each of these formulas must evaluate to a text value. @Commands are not allowed.

These formulas run on the user's workstation.

Note: Named element formulas are new with Release 6.

Examples: Named element formulas

1. The following formulas are for a named element representing the content of a frame. The frame contains a form named "Form A" if the current user is a manager in the current database and a page named "Page X" otherwise. In both cases, the named element is in the database "Test New Features.nsf."

   The "Formula for kind of named element" box contains:
   @If(@Contains(@DbManager; @UserName); "Form"; "Page")

   The "Formula for database" box contains:
   "Test New Features.nsf"

   The "Formula for name of named element" contains:
   @If(@Contains(@DbManager; @UserName); "Form A"; "Page X")

2. This example, when triggered from a hotspot button, either links to a page called "Web Customer Info," if the current user is a Web user, or to a form called "Customer Info" otherwise.

   In the "Formula for kind of named element" box, enter:
   @If(@Contains(@UserName; "Anonymous"); "Page"; "Form")

   In the "Formula for database" box, enter "Customer.nsf."

   In the "Formula for name of named element" box, enter:
   @If(@Contains(@UserName; "Anonymous"); "Web Customer Info"; "Customer Info")

---

Image formulas

Formulas can be used to specify the name of an outline image, the background image in a view, the background image in an embedded outline, the twisties image in a column, and the twisties image in an embedded outline.

@Commands are not allowed.

These formulas run on the user's workstation.

Note: Twistie images are new with Release 6.

Examples: Image formulas

This background image formula uses the image resource Backgrnd.gif if the current user is a manager in the current database and Cloud.gif otherwise.

@If(@Contains(@DbManager; @UserName); "Backgrnd.gif"; "Cloud.gif")
This twistie image formula uses the image resource folder.gif if the current user is not a manager in the current database.

@If(@Contains(@DbManager; @UserName); "folder.gif"; "")

**Events**

You can write scripts and formulas to handle events that occur during processing as shown in the descriptions topic. For information on the sequence of events based on certain tasks, see the table *sequencing*.

In the Programmer’s pane, you access objects and events through the Objects tab in the Info List.

Database, view, form, button, navigator, and field events run on the user’s workstation.

**Examples: Events**

1. This LotusScript event executes when a database opens. It opens a particular view depending on the value of the OrgUnit1 part of the user’s name.

   ```lotus
   Sub Postopen(Source As NotesUidatabase)
   Dim session As New NotesSession
   Dim userName As NotesName
   Set userName = session.CreateName(session.User)
   Select Case userName.OrgUnit1
   Case "Marketing"
     Call Source.OpenView("Marketing")
   Case "Engineering"
     Call Source.OpenView("Engineering")
   Case Else
     Call Source.OpenView("General")
   End Select
   End Sub
   ```

2. This LotusScript event executes before the Engineering view opens. If the OrgUnit1 part of the user’s name is not "Engineering," the script displays a message and sets Continue to False so that the view does not open.

   ```lotus
   Sub Queryopen(Source As NotesUiview, Continue As Variant)
   Dim session As New NotesSession
   Dim userName As NotesName
   Set userName = session.CreateName("session.User")
   If userName.OrgUnit1 <> "Engineering" Then
     Continue = False
     Messagebox _
     "This view is reserved for engineering",, , _
     "No admittance"
   End If
   End Sub
   ```

3. This LotusScript event executes when the user opens a document. The document goes into Edit mode even if the user doesn’t open it that way, the ruler opens, and all sections expand.

   ```lotus
   Sub Postopen(Source As NotesUidocument)
   source.EditMode = True
   source.Ruler = True
   Call source.ExpandAllSections
   End Sub
   ```

4. This JavaScript button script alters the value of one of the fields on the associated form. In this example, a document can be assigned a priority rating from 1 to 10, with 1 being the highest priority. The document’s priority rating is held in a field called “Priority” on the form. The button "Increase Priority" allows users to increase the priority without the need to edit the field directly. When the button is clicked, the Priority field’s value is decreased by 1, assigning it a higher priority rating. The script will not allow a priority of less than 1 to be assigned.
5. This LotusScript event executes when the user enters the FullName field. The script fills in the
Fullname field by concatenating the FirstName field, a space, and the LastName field.

Sub Entering(Source As Field)
    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Set uidoc = workspace.CurrentDocument
    firstName = uidoc.FieldGetText("FirstName")
    lastName = uidoc.FieldGetText("LastName")
    fullName = firstName & " " & lastName
    Call uidoc.FieldSetText("FullName", fullName)
End Sub

6. This JavaScript event executes when the focus moves away from the field. The script trims the white
space from the start and end of the field and then tests whether the value is a valid zip code. If it is
invalid, a message is displayed in the status bar and the focus is set back onto the field itself. Note that "trim" and "isValidZipCode" are user-defined functions that must be available from within the
onBlur handler.

var zipCode = trim(this.value);
if(!isValidZipCode(zipCode)){
    window.status = "Illegal zip code."
    this.focus();
}

7. This LotusScript event executes when the user exits from the Age field. The script forces the user to
enter a numeric value.

Sub Exiting(Source As Field)
    Dim workspace As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Set uidoc = workspace.CurrentDocument
    age = uidoc.FieldGetText("Age")
    If age = "" Or Not Isnumeric(age) Then
        While age = ""
            age = Inputbox _
                ("Whoa! you must enter an age")
        Wend
        While Not Isnumeric(age)
            age = Inputbox("Age must be numeric")
        Wend
        Call uidoc.FieldSetText("Age", age)
    End If
End Sub

8. This LotusScript event executes when the user exits from a field. The script sends a mail message to
Mary Chen.

Sub Exiting(Source As Field)
    Dim session As New NotesSessionFactory
    Dim db As NotesDatabase
    Set db = session.CurrentDatabase
    Dim doc As New NotesDocument(db)
    doc.Form = "Form"
    doc.Subject = "Sales Updated"
    doc.Body = "Sales updated by " & session.UserName
    Call doc.Send(True, "Mary Chen")
End Sub
Field design formulas

As part of the field design process, you can write the following formulas to run on the user’s workstation:

- Default value formula
- Input translation formula
- Input validation formula
- Input enabled formula
- HTML attributes formula
- Value formula for a computed field
- Keyword field formula

@Commands are not allowed.

Default value formulas

A default value formula provides an initial value for a field. This formula executes when a document is created, and for rich text fields that are not editable when the document is saved. The user can change the value in the field except for rich text fields that are not editable. You must specify a default value formula for an rich text field.

A default value formula must evaluate to a value suitable for storage in the current field.

Examples: Default value formulas
This default value formula makes the initial value for a field the user surname in uppercase letters. The @Left function returns the part of the name to the left of the first slash, if there is a slash, which extracts the name proper from the hierarchical name. The @RightBack function returns the part of the name to the right of the last space.

@UpperCase(@RightBack(@Left(@UserName; "/"; " "))

Input translation formulas

An input translation formula converts the data entered in the field to adjust the field value or make the field conform to a format. This formula executes when the document containing the field is saved.

An input translation formula should return a value of the same data type as the current field. For example, if the field is a Text field, the input translation formula should return a text string value. If the field is a Date/Time field, the input translation formula should return a date/time value.

Examples: Input translation formulas
This input translation formula makes the Subject field proper case (initial capitals) with no extraneous spaces.

@Trim(@ProperCase(Subject))

Input validation formulas

An input validation formula checks the data entered in the field against criteria that you specify. This formula executes after the input translation formula.

An input validation formula should end with a call to @Success or @Failure.

Examples: Input validation formulas
1. This input validation formula doesn’t let the Cost field exceed 100.

@If(Cost<100; @Success; @Failure("Cost must be less than $100"))
2. This formula forces the user to enter a ten-digit number. The formula removes up to two hyphens if they are present. The formula converts the result to a number and returns a failure if an error occurs on the conversion. The formula converts the number back to text and checks its length, returning a failure if the length is incorrect. This last step is necessary because @TextToNumber successfully converts, for example, 123A to 123.

\[
\begin{align*}
N1 := & \text{@if}(@\text{contains}(\text{Home\_Phone}; "-")) \text{; } @\text{left}(\text{Home\_Phone}; "-") \text{+ } @\text{right}(\text{Home\_Phone}; "-")) \text{; } \text{Home\_Phone}) \\
N2 := & \text{if}(@\text{contains}(\text{N1}; "-")) \text{; } @\text{left}(\text{N1}; "-") \text{+ } @\text{right}(\text{N1}; "-")) \text{; } \text{N1}) \\
N := & \text{@texttonumber}(\text{N2}) \text{; } @\text{if}(\text{@iserror}(\text{N}); \text{@return}\text{(@failure("Home\_phone\ must\ be\ zip-xxx-xxxx."))); } @\text{success}) \\
& @\text{if}(\text{@@length}(\text{@text}(\text{N})) \text{!= } 10; \text{@failure("Home\_phone\ must\ be\ zip-xxx-xxxx."))); } @\text{success})
\end{align*}
\]

**Input enabled formulas**

An input enabled formula enables or disables editing of a "Native OS style" field.

An input enabled formula should evaluate to @True or @False.

**Examples: Input enabled formulas**

1. This input enabled formula prevents a "Native OS style" field from being edited.
   
   @False

2. This formula prevents a "Native OS style" field from being edited if "Closed" is the value of the Status field.
   
   Status != "Closed"

**HTML attributes formulas**

An HTML attributes formula attaches HTML attributes to a field for use in Web applications.

An HTML attributes formula should be a text constant.

**Examples: HTML attributes formulas**

This HTML attributes formula specifies the size of a text field in a Web application.

"SIZE=40 MAXLENGTH=50"

**Value formulas for computed fields**

If the field type is Computed, Computed for display, or Computed when composed, you must write a formula to specify the field contents. This formula executes: for Computed, when the document is created and when it is saved; for Computed for display, whenever the document is opened; for Computed when composed, only when the document is created. Its value cannot be changed.

A computed field formula must evaluate to a value suitable for storage in the current field.

**Examples: Value formulas for a computed fields**

This computed field formula makes the Date field contain the time and date that the document is created.

@Created

**Keyword field formulas**

A keyword field presents the user with a list of keywords for entry into the field. The list can be the result of a formula or a list of constants.

A keyword field formula must evaluate to a value or list of values suitable for storage in the current field.

**Examples: Keyword field formulas**

1. This formula uses as a keyword list the values in column 1 of the Names view in the current database.
This formula reformats the user name as last name, comma, first name, taking care to handle hierarchical names. The formula looks up the reformatted user name in the Phone Numbers view of the current database and returns a list consisting of the values found in columns 2 and 3.

\[
\begin{align*}
n & := \text{Left}(@V3UserName; \\
& \text{Right}(n; " ") + \\
& \text{Left}(n; " "); \\
\text{DbLookup}++; \\
\text{DbLookup}++; \\
\text{DbLookup}++; \\
\end{align*}
\]

**Event descriptions**

Simple action, Formula, LotusScript, JavaScript, and Java code executes in response to the occurrence of events in the following objects: actions, action hotspots, agents, hotspot buttons, databases, fields, folders, forms, formula pop-ups, outlines, pages, script libraries, subforms, and views.

You code the same event twice; once for the Notes client platform and once for the Web platform.

**Note:** The separate handling of Notes client and Web browser events is new for Release 6. See [Compatibility issues](#) below. The one exception to the ‘coding same event twice’ rule is with actions and shared actions. If you are designing in Release 6 and running in Release 5, you must code two separate actions, one for each platform. Use the hide-when formulas to hide an action from Notes or Web platforms.

Web events can be coded only in JavaScript. You can specify the same JavaScript code for both a Web event and its corresponding Notes client event.

The following table lists the events that can be handled using the formula language, LotusScript, JavaScript and Java code in the Notes client and Web browser environments. Non-programming handlers such as simple actions are also listed. For information on sequencing of events, see the table [Event sequencing](#). For information on the user interface, see [Exploring the Programmer’s pane](#) and [Using the Objects tab](#).

<table>
<thead>
<tr>
<th>Event</th>
<th>Run</th>
<th>Language</th>
<th>Object</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Client</td>
<td>Formula</td>
<td>Agent</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Java</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imported Java</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simple action(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click</td>
<td>Client</td>
<td>Formula</td>
<td>Action</td>
<td>Object is selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simple action(s)</td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Navigator object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture Hotspot</td>
<td></td>
</tr>
<tr>
<td>Click</td>
<td>Client</td>
<td>Formula</td>
<td>Formula Popup</td>
<td>Object is selected</td>
</tr>
<tr>
<td>Column Value</td>
<td>Client</td>
<td>Field</td>
<td>Folder column</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td>Formula</td>
<td>View column</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simple Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Declarations</td>
<td>Client</td>
<td>LotusScript</td>
<td>Action</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Agent</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Database Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Globals</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Navigator object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LotusScript Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Library</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Default Value</td>
<td>Client</td>
<td>Formula</td>
<td>Field</td>
<td>Document is created</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Selection</td>
<td>Client</td>
<td>Simple Search</td>
<td>Agent</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entering <strong>Note:</strong></td>
<td>Client</td>
<td>LotusScript</td>
<td>Field</td>
<td>Object is selected</td>
</tr>
<tr>
<td>Exiting <strong>Note:</strong></td>
<td>Client</td>
<td>LotusScript</td>
<td>Field</td>
<td>Object is deselected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Form Formula</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Frame - Client</td>
<td>Client</td>
<td>Formula</td>
<td>Outline Entry</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HelpRequest</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Help is selected</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Hide When</td>
<td>Client</td>
<td>Formula</td>
<td>Outline Entry</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTML Attributes</td>
<td>Web</td>
<td>Formula</td>
<td>Field</td>
<td>Form is loaded</td>
</tr>
<tr>
<td>HTML Body Attributes</td>
<td>Web</td>
<td>Formula</td>
<td>Form</td>
<td>Form is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
<td>------------</td>
<td>-----------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>HTML Head Content</td>
<td>Web</td>
<td>Formula</td>
<td>Form</td>
<td>Form is loaded</td>
</tr>
<tr>
<td>Image</td>
<td>Client</td>
<td>Formula</td>
<td>Outline Entry</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Initialize</td>
<td>Client</td>
<td>LotusScript</td>
<td>Action</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Agent</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Database Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Globals</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Navigator object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LotusScript Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Library</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Input Enabled</td>
<td>Client</td>
<td>Formula</td>
<td>Field</td>
<td>Form is loaded</td>
</tr>
<tr>
<td>Input Translation</td>
<td>Client</td>
<td>Formula</td>
<td>Field</td>
<td>Document is saved</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Validation</td>
<td>Client</td>
<td>Formula</td>
<td>Field</td>
<td>After Input Translation</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InViewEdit Note: New with Release 6</td>
<td>Client</td>
<td>LotusScript</td>
<td>Folder</td>
<td>View entry is edited: at query, validation, and save</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>JavaAgent</td>
<td>Client</td>
<td>Java</td>
<td>Agent</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Javascript Library</td>
<td>Client</td>
<td>JavaScript</td>
<td>Script Library</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Java class name</td>
<td>Client</td>
<td>Java</td>
<td>Script Library</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>JS Header</td>
<td>Client</td>
<td>JavaScript</td>
<td>Form</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Client</td>
<td>Formula</td>
<td>Outline Entry</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>----------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ObjectExecute</td>
<td>Client</td>
<td>LotusScript</td>
<td>Action</td>
<td>Object is activated by an OLE2 server that is FX/NotesFlow™ enabled</td>
</tr>
<tr>
<td>onBlur</td>
<td>Client</td>
<td>JavaScript</td>
<td>Field</td>
<td>Object is deselected</td>
</tr>
<tr>
<td>onBlur - Web</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Object is deselected</td>
</tr>
<tr>
<td>onBlur - Web</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Object is deselected</td>
</tr>
<tr>
<td>onChange</td>
<td>Client</td>
<td>JavaScript</td>
<td>Field</td>
<td>Contents of object change</td>
</tr>
<tr>
<td>onChange</td>
<td>Web</td>
<td>JavaScript</td>
<td>Field</td>
<td>Contents of object change</td>
</tr>
<tr>
<td>onClick</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Object is selected</td>
</tr>
<tr>
<td>onDblClick</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Double-click occurs on object</td>
</tr>
<tr>
<td>onFocus</td>
<td>Client</td>
<td>LotusScript</td>
<td>Field</td>
<td>Object is selected</td>
</tr>
<tr>
<td>onFocus</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Object is selected</td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>onHelp</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Help is selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>JavaScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>onHelp</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Help is selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture</td>
<td></td>
</tr>
<tr>
<td>onKeyDown</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Any key is pressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture</td>
<td></td>
</tr>
<tr>
<td>onKeyPress</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>An alphanumeric key is pressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture</td>
<td></td>
</tr>
<tr>
<td>onKeyUp</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Any key is released</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** New for Formula and LotusScript with Release 6
<table>
<thead>
<tr>
<th>Event</th>
<th>Run</th>
<th>Language</th>
<th>Object</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>onLoad</td>
<td>Web</td>
<td>JavaScript</td>
<td>Form Page</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>onMouseDown</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Object is selected with either the right or left mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot Button Field Form Page Picture</td>
<td></td>
</tr>
<tr>
<td>onMouseMove</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Mouse is moved over object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot Button Field Form Page Picture</td>
<td></td>
</tr>
<tr>
<td>onMouseOut</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Mouse is moved out of object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot Button Field Form Page Picture</td>
<td></td>
</tr>
<tr>
<td>onMouseOver</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Mouse is moved into object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot Button Field Form Page Picture</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
<td>----------</td>
<td>----------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>onMouseUp</td>
<td>Web</td>
<td>JavaScript</td>
<td>Action</td>
<td>Mouse button is released over object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture</td>
<td></td>
</tr>
<tr>
<td>onReset</td>
<td>Web</td>
<td>JavaScript</td>
<td>Form</td>
<td>Object is reset</td>
</tr>
<tr>
<td>onSelect</td>
<td>Web</td>
<td>JavaScript</td>
<td>Field</td>
<td>Contents of object are selected</td>
</tr>
<tr>
<td>OnSubmit</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is saved</td>
</tr>
<tr>
<td>Note: New for</td>
<td></td>
<td>JavaScript</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>Formula and</td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LotusScript</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnSubmit</td>
<td>Web</td>
<td>JavaScript</td>
<td>Form</td>
<td>Before object is saved</td>
</tr>
<tr>
<td>OnUnload</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is unloaded</td>
</tr>
<tr>
<td>Note: New for</td>
<td></td>
<td>LotusScript</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>Formula and</td>
<td></td>
<td>JavaScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LotusScript</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onUnLoad</td>
<td>Web</td>
<td>JavaScript</td>
<td>Form</td>
<td>Before object is unloaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td>Client</td>
<td>LotusScript</td>
<td>Action</td>
<td>Object is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Agent</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Database Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Globals</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Navigator object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LotusScript Script Library</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>----------</td>
<td>-----------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>PostDocumentDelete</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>After a document is deleted (the document is still available)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PostDragDrop</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>After a drag and drop operation in object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>PostEntryResize</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>After a resize operation in a calendar folder or view</td>
</tr>
<tr>
<td>Note: New with Release 6</td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>PostModeChange</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>After object is changed to Read or Edit mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td>PostOpen</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>After object is opened</td>
</tr>
<tr>
<td>Note: Preferred is onLoad for Form and Page</td>
<td></td>
<td>LotusScript</td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>PostPaste</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>After a paste operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>PostRecalc</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>After object is refreshed (and values are recalculated)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JavaScript</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td>PostSave</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>After object is saved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JavaScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PostSend</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>After object is sent</td>
</tr>
<tr>
<td>Note: New with Release 6</td>
<td></td>
<td>JavaScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QueryAddToFiler</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Before a document is added to a folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------</td>
<td>----------</td>
<td>-------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QueryClose</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>Object is being closed</td>
</tr>
<tr>
<td>Note: Preferred is</td>
<td></td>
<td>LotusScript</td>
<td>Folder</td>
<td>onUnload for Form and Page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryDocumentDelete</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>Before a document is marked for deletion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td>QueryDocumentUndel</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>Before a document is unmarked for deletion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryDragDrop</td>
<td>Client</td>
<td>Formula</td>
<td>Database Script</td>
<td>Before a drag and drop operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryEntryResize</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Before a resize operation in a calendar folder or view</td>
</tr>
<tr>
<td>Note: New with Release 6</td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryModeChange</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before a document is changed to Read or Edit mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JavaScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td>QueryOpen</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is opened</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JavaScript</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td>QueryOpen</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Before object is opened</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryOpenDocument</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Before a document is loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryPaste</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Before a paste operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QueryRecalc</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is refreshed (and values are recalculated)</td>
</tr>
<tr>
<td>Note: New with Release 6</td>
<td></td>
<td>JavaScript</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td>QueryRecalc</td>
<td>Client</td>
<td>Formula</td>
<td>Folder</td>
<td>Before object is refreshed (and values are recalculated)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>QuerySave</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is saved</td>
</tr>
<tr>
<td>Note: Preferred is onSubmit for Form</td>
<td></td>
<td>LotusScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Run</td>
<td>Language</td>
<td>Object</td>
<td>Trigger</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------</td>
<td>------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>QuerySend</td>
<td>Client</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is sent</td>
</tr>
<tr>
<td><strong>Note:</strong> New with Release 6</td>
<td></td>
<td>JavaScript</td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegionDoubleClick</td>
<td>Client</td>
<td>Formula</td>
<td>Folder View</td>
<td>Region in a calendar view or folder is double-clicked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LotusScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Client</td>
<td>Formula</td>
<td>Outline Entry</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Target Frame</td>
<td>Client</td>
<td>Formula</td>
<td>Form Page</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target Frame (single click)</td>
<td>Client</td>
<td>Formula</td>
<td>Folder View</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target Frame (double click)</td>
<td>Client</td>
<td>Formula</td>
<td>Folder View</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminate</td>
<td>Client</td>
<td>LotusScript</td>
<td>Action</td>
<td>Object is unloaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Agent</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Database Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Folder</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Form</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Globals</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Navigator object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Picture Hotspot</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LotusScript Script</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Library</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>Client</td>
<td>Formula</td>
<td>Field</td>
<td>Computed field is calculated</td>
</tr>
<tr>
<td>Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>View Selection</td>
<td>Client</td>
<td>Formula</td>
<td>Folder View</td>
<td>Object is loaded</td>
</tr>
<tr>
<td>Simple Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebQueryOpen</td>
<td>Web</td>
<td>Formula</td>
<td>Form</td>
<td>Before object displays**</td>
</tr>
<tr>
<td>WebQuerySave</td>
<td>Web</td>
<td>Formula</td>
<td>Form</td>
<td>Before object is saved**</td>
</tr>
</tbody>
</table>
**WebQueryOpen and WebQuerySave must be a formula with either of the following @commands:
@Command([RunAgent];"agentname")
@Command([ToolsRunMacro];"agentname")

LotusScript subroutines and functions
You can add LotusScript subroutines and functions to an object. Your scripts are added to, and can be selected from, the list of events belonging to the object.

LotusScript Declarations and Options areas
Each object has a Declarations area where you can write non-executable LotusScript statements that apply to all LotusScript events in the object, and an Options area for statements such as Option, Use, UseLSX, and Const. Each form, folder, page, subform, and view has a Globals area where you can write non-executable LotusScript statements that apply to all LotusScript events in the object.

Current document in onLoad and PostOpen
Changes made to the current document in an onLoad or PostOpen event are treated as default values. The document is not marked as changed. If the user closes the document at this point, the onLoad or PostOpen changes are lost. You must explicitly save the changes, for example, with the Save method of NotesUIDocument, if you want to be sure they are applied.

Compatibility issues
In Notes Release 5 certain JavaScript events occur on the Notes client as well as the Web browser. In addition, Formula/LotusScript events that respond to the same user actions also occur on the Notes client. For example, both the onBlur and Exiting events occur when the Notes client user exits a field.

Release 6 distinguishes between the application of events to the Notes client and Web browser. You code one JavaScript or LotusScript event for the Notes client and a separate JavaScript event for the Web browser. Those JavaScript events that occur on the Notes client in Release 5 allow LotusScript and in some cases Formula in Release 6. The corresponding LotusScript-only events still occur, but their continued use in Release 6 applications is discouraged.

The following table lists the affected events:

<table>
<thead>
<tr>
<th>User action</th>
<th>Event</th>
<th>Release</th>
<th>Occurs on</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action, Action Hotspot, Button</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select object</td>
<td>onClick</td>
<td>R5 &amp; Release 6</td>
<td>Web only</td>
<td>JavaScript</td>
</tr>
<tr>
<td>Field</td>
<td>Click</td>
<td>R5 &amp; Release 6</td>
<td>Client only</td>
<td>LotusScript</td>
</tr>
<tr>
<td>Enter object</td>
<td>Entering</td>
<td>R5 &amp; Release 6</td>
<td>Client only</td>
<td>LotusScript</td>
</tr>
<tr>
<td></td>
<td>onFocus</td>
<td>R5 &amp; Release 6</td>
<td>Client &amp; Web</td>
<td>JavaScript</td>
</tr>
<tr>
<td>Exit object</td>
<td>Exiting</td>
<td>R5 &amp; Release 6</td>
<td>Client only</td>
<td>LotusScript</td>
</tr>
<tr>
<td></td>
<td>onBlur</td>
<td>R5 &amp; Release 6</td>
<td>Client &amp; Web</td>
<td>JavaScript</td>
</tr>
</tbody>
</table>
In Notes Release 5, for example, if you code the onBlur event for FieldOne as follows:

```javascript
with (window.document.forms[0]) {
  if (FieldOne.value == "foo") {
    FieldTwo.value = "bar"
  }
}
```

FieldTwo is set to "bar" if FieldOne is "foo" when the user exits the field in either a Web browser or the Notes client.

To duplicate this functionality in Release 6, you specify the code twice, once for the onBlur - Web event and again for the onBlur - Client event, or specify the code under "Common JavaScript."

If, in addition to coding onBlur, you code the Exiting event for FieldOne as follows:

```ltscript
Sub Exiting(Source As Field)
  Dim w As New NotesUIWorkspace
  Dim uidoc As NotesUIDocument
  Set uidoc = w.CurrentDocument
  If uidoc.FieldGetText("FieldOne") = "foo" Then
    Call uidoc.FieldSetText("FieldThree", "bar")
  End If
End Sub
```

FieldThree is also set to "bar" if FieldOne is "foo" when the user exits the field in the Notes client, but not in a Web browser.

In Release 6, if backwards compatibility is not an issue, you should combine the code in the onFocus - Client event and remove the Exiting event. You can code in JavaScript:

```javascript
with (window.document.forms[0]) {
  if (FieldOne.value == "foo") {
    FieldTwo.value = "bar"
    FieldThree.value = "bar"
  }
}
```

Or LotusScript:

```ltscript
Sub OnBlur(Source As Field)
  Dim w As New NotesUIWorkspace
  Dim uidoc As NotesUIDocument
  Set uidoc = w.CurrentDocument
  If uidoc.FieldGetText("FieldOne") = "foo" Then
```
Call uidoc.FieldSetText("FieldTwo", "bar")
Call uidoc.FieldSetText("FieldThree", "bar")
End If
End Sub

Forward compatibility

Notes Release 5 applications running in the Release 6 Notes client behave the same. Those JavaScript events that worked in the Notes client continue to work on both the Web and the client. The LotusScript PostOpen, QueryClose, QuerySave, Entering, Exiting, and Click events continue to work.

When a Release 5 application is saved in Release 6 Domino Designer, JavaScript events are moved into their corresponding Web and (where applicable) Client events. The LotusScript events remain. For example, the onBlur event is moved into the onBlur - Web and onBlur - Client events, and the Exiting event remains.

However, if backwards compatibility is not an issue, you are urged to revise the code in the following cases:
- For the JavaScript events that run on both the Notes client and the Web, where you use conditional code to modify the behavior for one environment or the other: instead use two routines, one for the client and one for the Web.
- For the LotusScript events: move the code to the corresponding "on" client event.

Backwards compatibility

Release 6 applications running in the Release 5 Notes client behave the same as in the Release 6 client except that events new to Release 6 do not work in R5. For example, onLoad - Client for LotusScript does not work in Release 5.

If you recompile an Release 6 application in Release 5, JavaScript client events are lost and JavaScript Web events are reinstated for both client and Web. For example, onLoad - Client for JavaScript disappears if compiled in Release 5. JavaScript - Web does not disappear if compiled in Release 5 but now works for both client and Web.

Therefore a caveat exists to the guidelines for forward compatibility. If you continue to design in both Release 5 and Release 6, you should keep the LotusScript PostOpen, QueryClose, QuerySave, Entering, and Exiting events, not use the LotusScript "on" events, and code the JavaScript events for both client and Web (use "Common JavaScript").

onHelp and HelpRequest

In Release 5 the HelpRequest event (Formula) performs your action and suppresses standard help when the user presses F1 in the Notes client. The onHelp event (JavaScript) performs your action and (in addition) calls standard help when the user presses F1 in a Web browser.

In Release 6 the onHelp - Web event is the same as the Release 5 onHelp event. The onHelp - Client event, which supports Formula, LotusScript, and JavaScript, has the behavior of HelpRequest in Release 5. HelpRequest is gone from form design.

Event sequencing

The following table shows the sequencing of events during common Notes tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Sequence of events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing modes (edit/read) in a document</td>
<td>QueryModeChange (Form)</td>
</tr>
<tr>
<td></td>
<td>PostModeChange</td>
</tr>
<tr>
<td>Task</td>
<td>Sequence of events</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Closing a database</td>
<td>QueryClose&lt;br&gt;Terminate&lt;br&gt;[optional] Script Library Terminate</td>
</tr>
<tr>
<td>Closing a database from a view</td>
<td>QueryClose (View)&lt;br&gt;Terminate (View)&lt;br&gt;Globals Terminate (View)&lt;br&gt;[optional] Script Library Terminate (View)&lt;br&gt;QueryClose (Database)&lt;br&gt;Terminate (Database)&lt;br&gt;[optional] Script Library Terminate (Database)</td>
</tr>
<tr>
<td>Closing a document</td>
<td>QueryClose (Form)&lt;br&gt;onUnload&lt;br&gt;Terminate (Fields)&lt;br&gt;Terminate (Form)&lt;br&gt;Globals Terminate&lt;br&gt;[optional] Script Library Terminate</td>
</tr>
<tr>
<td>Composing a new document</td>
<td>[optional] Script Library Initialize&lt;br&gt;Globals Initialize&lt;br&gt;Initialize&lt;br&gt;JS Header&lt;br&gt;QueryOpen&lt;br&gt;Initialize (Fields)&lt;br&gt;PostOpen&lt;br&gt;onLoad</td>
</tr>
<tr>
<td>Deleting a document in a view</td>
<td>QueryDocumentDelete (Database Script event)&lt;br&gt;PostDocumentDelete (Database Script event)</td>
</tr>
<tr>
<td>Task</td>
<td>Sequence of events</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Editing an existing document</td>
<td>QueryOpenDocument (View)</td>
</tr>
<tr>
<td></td>
<td>[optional] Script Library Initialize (Form)</td>
</tr>
<tr>
<td></td>
<td>Globals Initialize</td>
</tr>
<tr>
<td></td>
<td>Initialize (Form)</td>
</tr>
<tr>
<td></td>
<td>JS Header</td>
</tr>
<tr>
<td></td>
<td>QueryOpen</td>
</tr>
<tr>
<td></td>
<td>Initialize (Fields)</td>
</tr>
<tr>
<td></td>
<td>PostOpen</td>
</tr>
<tr>
<td></td>
<td>onLoad</td>
</tr>
<tr>
<td>Entering a field</td>
<td>Entering</td>
</tr>
<tr>
<td></td>
<td>onFocus</td>
</tr>
<tr>
<td>Exiting a field</td>
<td>onBlur</td>
</tr>
<tr>
<td></td>
<td>Exiting</td>
</tr>
<tr>
<td></td>
<td>onChange</td>
</tr>
<tr>
<td>Leaving a view</td>
<td>QueryClose</td>
</tr>
<tr>
<td></td>
<td>Terminate</td>
</tr>
<tr>
<td></td>
<td>Globals Terminate</td>
</tr>
<tr>
<td></td>
<td>[optional] Script Library Terminate</td>
</tr>
<tr>
<td>Opening a database to a view</td>
<td>[optional] Script Library Initialize (View)</td>
</tr>
<tr>
<td></td>
<td>Globals Initialize (View)</td>
</tr>
<tr>
<td></td>
<td>Initialize (View)</td>
</tr>
<tr>
<td></td>
<td>QueryOpen (View)</td>
</tr>
<tr>
<td></td>
<td>PostOpen (View)</td>
</tr>
<tr>
<td></td>
<td>[optional] Script Library Initialize (Database)</td>
</tr>
<tr>
<td></td>
<td>Initialize (Database)</td>
</tr>
<tr>
<td></td>
<td>PostOpen (Database)</td>
</tr>
<tr>
<td>Opening a database</td>
<td>[optional] Script Library Initialize</td>
</tr>
<tr>
<td></td>
<td>Initialize</td>
</tr>
<tr>
<td></td>
<td>PostOpen</td>
</tr>
<tr>
<td>Opening a view</td>
<td>[optional] Script Library Initialize</td>
</tr>
<tr>
<td></td>
<td>Globals Initialize</td>
</tr>
<tr>
<td></td>
<td>Initialize</td>
</tr>
<tr>
<td></td>
<td>QueryOpen</td>
</tr>
<tr>
<td></td>
<td>PostOpen</td>
</tr>
<tr>
<td>Task</td>
<td>Sequence of events</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Refreshing a document</td>
<td>Postrecalc (Form)</td>
</tr>
<tr>
<td>Refreshing a view</td>
<td>QueryRecalc</td>
</tr>
<tr>
<td>Running an agent</td>
<td>[optional] Script Library Initialize</td>
</tr>
<tr>
<td></td>
<td>Initialize</td>
</tr>
<tr>
<td></td>
<td>Terminate</td>
</tr>
<tr>
<td></td>
<td>[optional] Script Library Terminate</td>
</tr>
<tr>
<td>Saving a document</td>
<td>QuerySave (Form)</td>
</tr>
<tr>
<td></td>
<td>onSubmit</td>
</tr>
<tr>
<td></td>
<td>PostSave</td>
</tr>
<tr>
<td>Undeleting a document in a view</td>
<td>QueryDocumentUndelete (Database Script event)</td>
</tr>
</tbody>
</table>
Chapter 2. User Interface

This section provides an introduction to the Programmer’s pane user interface in Lotus Domino Designer. The Programmer’s pane allows you to add functionality to applications with Java, JavaScript, LotusScript, Formula language, and Simple actions.

Accessing the Programmer’s pane

1. Launch Lotus Domino Designer. You can launch Lotus Domino Designer from the operating system or the Notes client. From the Notes client, click the Designer icon or choose View - Design from a database menu.
2. Open an existing design element or resource, or create a new one.

For more information see the Introduction to Domino Designer topics "Exploring Lotus Domino Designer" and "Starting Lotus Domino Designer" in Application Development with Domino Designer.

Exploring the Programmer’s pane

The Programmer’s pane is located in Lotus Domino Designer in the lower half of the Work pane, or by itself in some design elements and resources such as agents and shared fields. The Programmer’s pane is context-sensitive and may change slightly depending on the programming language you select.

For information about developing design elements, see Application Development with Domino Designer.

The Programmer’s pane consists of the following main components.
<table>
<thead>
<tr>
<th>Interface element</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors box</td>
<td>Displays compile-time errors for LotusScript and Java.</td>
</tr>
<tr>
<td>Help button</td>
<td>Launches Lotus Domino Designer 6 Help from the Programmer’s pane. Located on the Reference tab. Same as F1 or Help - Context Help.</td>
</tr>
<tr>
<td>Info list</td>
<td>Contains the Objects and Reference tabs.</td>
</tr>
<tr>
<td>Objects tab</td>
<td>Lists all of the objects and events available for programming in the current context.</td>
</tr>
<tr>
<td>Paste button</td>
<td>Pastes selected information into the Script area. CTRL+V is a shortcut.</td>
</tr>
<tr>
<td>Paste full text check box</td>
<td>If selected, pastes the syntax of an event, method, or property from the Reference tab into the Script area when you click the Paste button.</td>
</tr>
<tr>
<td>Reference list</td>
<td>Allows you to see and select reference information.</td>
</tr>
<tr>
<td>Reference tab</td>
<td>Lists reference information and syntax.</td>
</tr>
<tr>
<td>Run menu</td>
<td>Enables you to select the desired programming environment (Client or Web) and language (Formula, Simple action(s), LotusScript, JavaScript, Imported Java, and Java). It displays only the environments and languages available for the current design state. You can choose to use the same JavaScript for both environments by selecting the option Common JavaScript. <strong>Note:</strong> Using the Run menu to select a programming environment is new with Release 6.</td>
</tr>
<tr>
<td>Script area</td>
<td>Provides a space to write and compile all recognized programming languages.</td>
</tr>
<tr>
<td>Title bar</td>
<td>Displays the title of the design element that the programming language is attached to. Double-click here to maximize or minimize the Programmer’s pane (does not change for Agents).</td>
</tr>
</tbody>
</table>
Exploring the Java interface in the Programmer’s pane

Java programming can be done in the Programmer’s pane of an agent or a script library.

<table>
<thead>
<tr>
<th>Interface element</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes tab</td>
<td>Lists all classes being used in the agent.</td>
</tr>
<tr>
<td>Compile button</td>
<td>Compiles the current class, or the entire agent.</td>
</tr>
<tr>
<td>Edit Project button</td>
<td>Allows you to add resource, class, source or archive files to an agent project. It can also be used to include a Script Library on the class path.</td>
</tr>
<tr>
<td>Export button</td>
<td>Exports the code from an agent to a Java source file.</td>
</tr>
<tr>
<td>Java Debug Console</td>
<td>Displays the output generated by your Java programming in a separate task. Choose File - Tools - Show Java Debug Console.</td>
</tr>
<tr>
<td>New Class button</td>
<td>Creates a space for a new class in the Script area.</td>
</tr>
<tr>
<td>Objects tab</td>
<td>Lists all of the objects and events available for programming Java agents.</td>
</tr>
</tbody>
</table>

**Note:** The Objects tab is new with Release 6.

When [Imported Java](#) is selected from the Run menu, the Programmer’s pane does not include the Edit Project, New Class, Export, and Compile buttons. The Programmer’s pane includes the following features for Imported Java.
### Interface

<table>
<thead>
<tr>
<th>Interface element</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base class box</td>
<td>Allows you to enter the base class of the current agent. If you import class files, the base class is displayed here.</td>
</tr>
<tr>
<td>Import Class Files button</td>
<td>Imports Java class files into an agent.</td>
</tr>
</tbody>
</table>

### Using the Info List

The Info List appears on the left side of the Programmer’s pane and contains the following context-sensitive tabs:

- **Objects**
- **Reference**
- **Classes** (Java only)

#### Adjusting the size of the Info List

You can adjust the width of the Info List by dragging the vertical bar on the right side of the box to the desired size. For some design elements you can adjust the height of the Info List, by dragging the horizontal bar at the top of the box to the desired size. Adjusting the height of the Info List also affects the height of the entire Programmer’s pane.

### Using the Objects tab

The Objects tab is located in the Info List. It contains a context-sensitive list of objects and events available for the selected programming language. The Objects tab allows you to:

- Program any object or event in the design element.
- Locate or view programmed objects.
- Determine the programming language used for a particular object or event at a glance.

Press CTRL+0 (zero) to access the Objects tab from the Script area using the keyboard. In the Java programming environment, CTRL+0 accesses the Classes tab.
Programming an object’s properties and events

1. Select the desired run-time platform and language, if available, from the Run menu.
2. Select the object’s property or event from the Objects tab.
3. Type or paste code into the Script area.

Locating code within an object

To locate code within an object, click on the object in the Work pane. The pointer on the Objects tab jumps to the appropriate object. The code can be viewed by clicking the appropriate event. For example, to find code attached to a button, click the button in the Work pane, the Objects tab jumps to the button object place holder and displays the code attached to the click event. To find code attached to the onBlur event of a field, click the field in the Work pane, the Objects tab jumps to the field object place holder and displays the first coded event in the list. If the onBlur event is not the first coded event for the field object, you must click the onBlur event to display its code in the Script area.

Note: When you click from one object to another of the same type, the event selected for the first object is also selected for the second object. For example, if you click a field in the Work pane and select the Default Value event, then click a different field, the event selected for the second field is Default Value. Clicking objects of different types in the Work pane moves the Objects tab pointer to the first coded event for each object.

Color-coding event icons

All events that contain code are color-coded for the appropriate run-time platforms. An event coded to run in the Notes client displays a green icon. An event coded to run on the Web displays as a yellow icon. An event coded for both platforms displays half green and half yellow with a red dividing line down the center.

Determining the programming language at a glance

Each programming language recognized in the Programmer’s pane has an icon related to it. These icons appear to the left of the object or event on the Objects tab. The following table shows the relationship of each icon to its respective programming language.

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula language</td>
<td>◆</td>
</tr>
<tr>
<td>JavaScript</td>
<td>◆</td>
</tr>
<tr>
<td>LotusScript</td>
<td>❌</td>
</tr>
<tr>
<td>Simple Action(s)</td>
<td>◆</td>
</tr>
<tr>
<td>System Command(s)</td>
<td>◆</td>
</tr>
</tbody>
</table>

Using the Reference tab

The Reference tab provides you with context-sensitive information for each programming language recognized in the Programmer’s pane. You can view this information by clicking the Reference tab and selecting the desired topic from the Reference list.

The Reference tab allows you to:
• See the syntax for an event, method, or property.
Paste information from the Reference tab into the Script area.

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Reference tab components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula Language</td>
<td>Database Fields, Formula @Commands, Formula @Functions</td>
</tr>
<tr>
<td>Imported Java</td>
<td>No Reference tab available.</td>
</tr>
<tr>
<td>Java</td>
<td>Core Java, Notes Java, Third-party Java</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Web D.O.M., Notes D.O.M.</td>
</tr>
<tr>
<td>LotusScript</td>
<td>LotusScript Language, Domino: Classes, Domino: Constants,</td>
</tr>
<tr>
<td></td>
<td>Domino: Subs and Functions, Domino: Variables, OLE Classes</td>
</tr>
<tr>
<td>Simple Action(s)</td>
<td>No Reference tab available.</td>
</tr>
</tbody>
</table>

Pasting information from the Reference tab into the Script area

1. Place the cursor where you want the item to appear in the Script area.
2. Click the item you want to paste, then click Paste.

Tip: You can also double-click the item to paste it.

Using shortcut keys to access the Reference tab

The table below lists key combinations you can use to access the Reference tab from the Script area.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Selects</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL+1</td>
<td>The first reference component</td>
</tr>
<tr>
<td>CTRL+2</td>
<td>The second reference component</td>
</tr>
<tr>
<td>CTRL+3</td>
<td>The third reference component</td>
</tr>
<tr>
<td>CTRL+4</td>
<td>The fourth reference component</td>
</tr>
<tr>
<td>CTRL+5</td>
<td>The fifth reference component</td>
</tr>
<tr>
<td>CTRL+6</td>
<td>The sixth reference component</td>
</tr>
</tbody>
</table>

Using the Errors box

The Errors box is present for Java and LotusScript. Use the Errors box to view error messages that occur during script entry and compilation. The Errors box displays only one error at a time. To view other errors, click on the box and select the error from the list. When you select an error from the list, the line containing the error is highlighted in the Script area.

Using the Script area

The Script area formats LotusScript automatically by indenting and adjusting the case of keywords as needed. There is no automatic formatting for other programming languages in Lotus Domino Designer.

Use the Script area to:

- Set Script area properties
- Move the insertion point while editing code
- Select text
- Edit text with menu commands or with key combinations
- Save and delete code
- Rename an object, subprogram, or class
Automatically complete statements

**Setting Script area properties**
The Programmer’s Pane Properties box allows you to:
- Set text properties for the desired programming language
- Set automatic code formatting properties
- Control Auto Complete properties

To open the Properties box, click inside the Script area and then choose Edit - Properties.

**Setting text properties**
These selections, located on the Font tab, apply to all text in the Programmer’s pane for all design elements.

To set the text properties for a specific programming language, if available, click Script/Java, Formulas, or Simple actions under "Text formatting for all." Select the desired text font, size, and color from the lists provided.

Select different colors as desired to represent identifiers, keywords, comments, errors, and constants using the lists on the right side of the Properties box.

**Setting format properties**
The Format tab selections apply to specific languages.

LotusScript code is automatically indented. To disable this property check the Auto-Indent LotusScript box.

To specify whether line wrapping is enabled or disabled when writing formula language, check the Auto-Wrap Formulas box. The default setting is enabled.

Use the Automatically add "Option Declare" box to add the Option Declare statement to new LotusScript objects. The default setting is disabled.

**Moving the insertion point while editing text in the Script area**
The table below lists keys that move the insertion point in the Programmer’s pane.

<table>
<thead>
<tr>
<th>Key</th>
<th>Moves the insertion point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+End</td>
<td>To the end of the last line.</td>
</tr>
<tr>
<td>Ctrl+Home</td>
<td>To the beginning of the first line.</td>
</tr>
<tr>
<td>Ctrl+Left Arrow</td>
<td>To the first character in the previous word.</td>
</tr>
<tr>
<td>Ctrl+Right Arrow</td>
<td>To the first character in the next word.</td>
</tr>
<tr>
<td>Down Arrow</td>
<td>One line down.</td>
</tr>
<tr>
<td>End</td>
<td>To the end of the line.</td>
</tr>
<tr>
<td>Home</td>
<td>To the beginning of the line.</td>
</tr>
<tr>
<td>Left Arrow</td>
<td>One character to the left.</td>
</tr>
<tr>
<td>Right Arrow</td>
<td>One character to the right.</td>
</tr>
<tr>
<td>Up Arrow</td>
<td>One line up.</td>
</tr>
</tbody>
</table>
Tip: Use PgUp and PgDn to scroll one screen up or down, overlapping one line from the previous screen.

Selecting text
You can select text with key combinations and the mouse.

Table of key combinations

<table>
<thead>
<tr>
<th>Key combination</th>
<th>Selects</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL+A</td>
<td>All text in the Script area.</td>
</tr>
<tr>
<td>SHIFT+CTRL+LEFT ARROW</td>
<td>The previous word or selects the first part of the word if the insertion point is in the word.</td>
</tr>
<tr>
<td>SHIFT+CTRL+RIGHT ARROW</td>
<td>The next word or selects the remainder of the word if the insertion point is in the word.</td>
</tr>
<tr>
<td>SHIFT+DOWN ARROW</td>
<td>The text starting at the right of the insertion point and ending at the character below and to the left of the insertion point.</td>
</tr>
<tr>
<td>SHIFT+END</td>
<td>The text starting with the character at the insertion point and ending with the last character on the line.</td>
</tr>
<tr>
<td>SHIFT+HOME</td>
<td>The text starting with the character to the left of the insertion point and ending with the first character on the line.</td>
</tr>
<tr>
<td>SHIFT+LEFT ARROW</td>
<td>The character to the left of the insertion point.</td>
</tr>
<tr>
<td>SHIFT+RIGHT ARROW</td>
<td>The character to the right of the insertion point.</td>
</tr>
<tr>
<td>SHIFT+UP ARROW</td>
<td>The text starting at the left of the insertion point and ending at the character above and to the right of the insertion point.</td>
</tr>
</tbody>
</table>

Table of mouse operations

<table>
<thead>
<tr>
<th>Mouse operation</th>
<th>Selects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-click (on the word)</td>
<td>The entire word and the following space.</td>
</tr>
<tr>
<td>Drag</td>
<td>The text from the starting position of the mouse to the ending position.</td>
</tr>
<tr>
<td>Right-click</td>
<td>The Edit menu.</td>
</tr>
<tr>
<td>SHIFT+Click (left mouse click)</td>
<td>The text from the insertion point to the position of the mouse.</td>
</tr>
</tbody>
</table>

Editing text with key combinations

<table>
<thead>
<tr>
<th>Key combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL+F</td>
<td>Finds and replaces text.</td>
</tr>
<tr>
<td>CTRL+Y</td>
<td>Redo.</td>
</tr>
<tr>
<td>CTRL+Z</td>
<td>Undo.</td>
</tr>
</tbody>
</table>

Editing text with menu commands

<table>
<thead>
<tr>
<th>Menu command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit - Copy</td>
<td>Copies selected text to the Clipboard.</td>
</tr>
<tr>
<td>Edit - Cut</td>
<td>Moves selected text to the Clipboard.</td>
</tr>
<tr>
<td>Edit - Delete</td>
<td>Deletes selected text (doesn’t go to the Clipboard).</td>
</tr>
</tbody>
</table>
### Saving and deleting text in the Script area

You can use the menu bar to save or delete chunks of code.

**To save text in the Script area**

Choose File - Save to save a design element or resource and all of its associated programming. You can also save a design element or resource when you exit from it.

**To delete text in the Script area**

1. Select the object or event from the Objects tab.
2. Choose Edit - Select All, or manually select all the text in the Script area.
3. Choose Edit - Cut or Edit - Delete.

### Renaming an object, subprogram, or class

For Formula, JavaScript, LotusScript, and Simple action(s) you can rename an object by changing its name in its Properties box. The programming attached to that object will automatically be associated with its new name. An agent may be renamed without affecting the Java in it. You must change any references to the object; for example, calls to methods that reference the name of the object.

To rename a subroutine or function, edit the script that defines it. If you change the name of a Domino-defined subprogram such as Initialize or Click, a new user-defined subprogram with the revised name is created and automatically added to the Info List. The script from the original event is moved to the newly-created subprogram, and the original script becomes empty.

### Automatically completing statements

Use auto completion to look up and paste syntax elements right into the Programmer’s pane as you enter code. Auto completion uses type-ahead functionality to let you quickly find and select the options you want, and also provides parameter prompting to show you how to complete a statement.

**Note:** Auto completion is new with Release 6.

Use auto completion when programming the following:

- **Formula language @functions**
- **Formula language @commands**
- **LotusScript classes**

**Using auto completion for formula language @functions**

In the Programmer’s pane, follow these conventions to use auto completion:

- Enter the at (@) symbol to trigger auto complete. A pop-up list of available @functions appears.
From the pop-up list, you can either type ahead to select the function you need, or scroll through the list and select it. Press ENTER to paste the function into the Script area and close the list.

- If the function has a parameter list, type an opening parenthesis ( ) to display a pop-up box with the parameter prompt. The syntax of the first parameter appears in bold.
- If the function has keyword options, up and down arrows display surrounding two colon-delimited numbers. The first number identifies the number of the first keyword option, which is displayed in bold; the second number identifies the total number of keyword options to choose from. For example, @Prompt contains ten keyword options of which [OK] is the first. @Prompt displays as follows in the auto completion pop-up:
The up and down arrows indicate that you can click the up and down keyboard keys to display each option in the list of keyword options.

```
+--------------------------------------------+
| 1:10                                      |
+--------------------------------------------+
```

@Prompt( [OK]; title:prompt)
If you click the down arrow key on the keyboard, the display changes to display the second keyword option:
```
2:10 @Prompt([YesNo]; title:prompt)
```

- Type a semicolon (;) after each parameter. The syntax of the next parameter in the string appears in bold.
- Type a closing parenthesis ( ) ) or press ESC to close the pop-up box.

**Using auto completion for formula language @commands**

In the Programmer’s pane, follow these conventions to use auto completion:

- Enter an at sign (@) to see a pop-up list of functions. Select ”command” from the list and then enter an opening parenthesis ( ). A pop-up list of command names appears.
  From the pop-up list, you can either type ahead to select the command name you need, or scroll through the list and select it. Press ENTER to paste the command into the Script area and close the list. Bracket are automatically placed around the command name like this:
  @Command([EditFind])

- If the command has a parameter list, type a semicolon (;) to display a pop-up box with the parameter list. The syntax of the first parameter appears in bold.
  Type a semicolon (;) after each parameter. The syntax of the next parameter in the string appears bold.
  Type a closing parenthesis ( ) ) or press ESC to close the pop-up box.

**Using auto completion for LotusScript class statements**

In the Programmer’s pane, follow these conventions to use auto completion:

- When declaring an object using the Dim statement, enter the word As followed by a SPACE to trigger the pop-up list of available classes, as in this example:
  Dim doc As[SPACE]
  From the pop-up list, you can either type ahead to select the class you need, or scroll through the list and select it. Press ENTER to paste the class into the Script area and close the list.

- To see a list of available properties and methods for a class, enter a period ( . ) after an object name.
  From the pop-up list, you can either type ahead to select the element you need, or scroll through the list and select it. Press ENTER to paste the element into the Script area and close the list.

- If a method has a parameter list, type an open parenthesis ( ) to display a pop-up box with the parameter list. The syntax of the first parameter appears in bold.
  Type a comma (, ) after each parameter. The syntax of the next parameter in the string appears bold.
To enable/disable auto completion
Auto completion is enabled by default. Use the Auto Complete tab on the Programmer's Pane Properties box to change auto completion settings.

Note that even if the options are disabled you can still use the menu commands or accelerator keys to display pop-up lists or boxes.

To redisplay a list of options, you can do any of the following:
• Choose Edit - List Members from the menu
• Press CTRL+ALT+T
• Type an at sign (@) when writing formula language code, if auto completion is enabled

To redisplay a parameter box, if one is applicable in this situation, you can do either of the following:
• Choose Edit - Parameter Info from the menu
• Press CTRL+SHIFT+SPACE

Options on the Auto Complete tab
The options on the Auto Complete tab apply to all languages. Select or deselect the options as follows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto List Member box</td>
<td>A pop-up list of available options automatically displays when you begin typing code.</td>
<td>The default is enabled. To disable auto lists, deselect this box.</td>
</tr>
<tr>
<td>Auto Parameter Popup box</td>
<td>If an option has a parameter list, a pop-up box with a parameter prompt appears when you type a language-specific trigger, such as an open parenthesis ( ( ) for @functions.</td>
<td>The default is enabled. To disable parameter prompting, deselect this box.</td>
</tr>
<tr>
<td>Delay box</td>
<td>The value is the amount of time to wait between a trigger keystroke and another keystroke before a pop-up list displays. If you type a second keystroke before the time has elapsed, the pop-up list does not appear. The value is in milliseconds.</td>
<td>The default is 200 milliseconds.</td>
</tr>
</tbody>
</table>

Writing Java in an agent
1. Select Agents from the Design pane.
2. Click New Agent or open an existing agent. The Agent Properties box appears.
3. Specify name, options, run time options, and security as needed.
4. Select Java from the Run menu.
5. Write your Java after the remark line, “Your code goes here,” or click Edit Project to add source, resource, class, or archive files to the current agent.

Note: Any changes made to the Java files in the current agent exist only in the agent. If you want to update the original files, you must export the agent to the directory containing the files you want to update.

6. (Optional) Click New Class to add a new class. A horizontal bar separates each class. The name of a class appears on the bar above the class to which it refers.
7. Click on Compile below the Script area. You can choose to compile the current class or the entire agent.
8. (Optional) Click Export to export Java to a project directory.
For information about creating an agent, see the Adding Automation to Applications topic "Creating an agent” in Application Development with Domino Designer.

Note: The Agent Properties box is new with Release 6.

Viewing output from a Java agent
You can review output from a Java agent that is running locally in the Java Debug Console. To activate the console, choose File - Tools - Show Java Debug Console. To clear output from the window, click Clear.

If a Java agent is running on a server, the Java Debug Console output is redirected to a server log file.

Adding resource, class, or archive files to an agent
1. Click Edit Project.
2. From the Browse menu, choose the filing system for the resources you want to import.
3. If using the Local File System, enter the desired path in the Base directory box.
4. (Optional) Select the appropriate check box for Show file types. The default file type is All.
5. Select the files you want to include as a resource and click Add/Replace File(s). To select all of the files in the base directory and all of its subdirectories, click Add/Replace All. To remove a file, or group of files from the Current Agent Files column, select the file(s) and click Delete.
6. Put the files in the proper order in the Current Agent Files column by clicking Reorder Up or Reorder Down to move files up or down, or by dragging files into place. You can move files one at a time, or in groups.

Including a script library on the class path
1. Click Edit Project.
2. Choose Shared Java Libraries from the Browse menu.
3. Choose the desired database from the options in the Database box.
4. Select the script library to add to the class path.
5. Click Add/Replace File(s).

Note: The Database box is new with Release 6.

Compiling Java
There are three ways to compile Java:
• A complete compilation occurs when you choose File - Save. All compile-time errors are reported at this time.
• You can compile each class individually by clicking the class in the Script area, then clicking Compile and selecting "Compile Class Name."
• Compile all of the classes at once by clicking the Compile button and selecting "Compile All."

Errors are reported in the Errors box. Click the Errors box to see which line in the Script area contains the error.

Errors not detectable during compilation are reported at run time.

Exporting Java
You can export the Java code you wrote in a Java agent or Java script library to the file system.
1. Make sure the code compiles and runs as expected.
2. With the Java agent or script library open in the programmer’s pane, click Export.

The Choose Base Directory dialog box displays.
3. Select the directory you want to save the Java code to.
4. Click Ok.
   The resulting file is called "JavaAgent.java."
   For details on exporting Java applets, see the "Including Java applets in applications" chapter in the Application Development with Domino Designer book.

Importing Java

Select Imported Java from the Run menu.
1. Click Import Class Files.
2. Under the Available Java Files column, select the filing system and base directory for the files you want to import. If you are importing from a named package, the base directory must be above the directory containing the class files.
3. (Optional) Select the appropriate check box for Show file types. The default file type is All.
4. Select the files you want to import from the file list and click Add/Replace File(s). To import all of the files in the base directory and all of its subdirectories, click Add/Replace All. To remove a file, or group of files from the Current Agent Files column, select the file(s) and click Delete.
5. Put the files in the proper order in the Current Agent Files column by clicking Reorder Up or Reorder Down to move files up or down, or by dragging files into place. You may move files one at a time, or in groups.
6. Enter the base class in the Base class box or select it from the list.
7. (Optional) Click Refresh, or Refresh All to reload files in the Current Agent Files column.
8. Click OK to import files.
9. (Optional) Click Reimport Class Files to import additional files, or reimport an existing file.

Writing JavaScript in the Programmer’s pane

JavaScript allows you to write scripts that function on the Web as well as in the Domino client. The Objects tab provides a list of JavaScript events supported in Lotus Domino Designer. JavaScript is not available for agents. For LotusScript Web agents, you can use Print statements to output HTML to the browser, including script statements. For example, Print "<script>function changeLocation() {...}</script>.

The Programmer’s pane allows you to:
- Program object events with JavaScript
- Use [S Header] to enter JavaScript directly into the page header
- Compile your JavaScript before leaving the Programmer’s pane
- Import and export JavaScript files

To program an object event in JavaScript
1. Click the placeholder for the object event in the Objects tab
2. Select the client you want the code to run in from the first Run drop down list. 
   Client indicates the Notes client and Web indicates a Web browser.
3. Select JavaScript from the second Run drop down list.
   If you select Common JavaScript, the code you enter executes in both the Notes client and a Web browser.
4. Type your JavaScript in the Script area.
Writing JavaScript in a page header
The Programmer’s pane provides a placeholder called JS Header on the Objects tab that allows you to write JavaScript in the header of a page, form, or subform. JavaScript written in the header in this way does not require a formula to return a JavaScript function, or text to be marked as passthru HTML.
1. Click JS Header on the Objects tab.
2. Select Web or Client from the first Run drop down list to indicate whether this code executes when accessed by a Web browser or the Notes client.
3. Select JavaScript or Common JavaScript from the second drop down list.
   If you select Common JavaScript, the code you enter executes in both the Notes client and a Web browser.
4. Write your JavaScript in the Script area.

Compiling JavaScript
JavaScript is compiled by choosing File - Save, or clicking the green check mark,

✅

but it is not stored in its compiled form. All JavaScript is recompiled each time it is run.

Errors detected when the script is saved are displayed in the status bar at the bottom of the Script area. Only one error is displayed at a time. Once an error is detected, you must correct the errors before you can exit the design element. If you want to save a script with errors, make the incorrect line a remark by preceding it with two forward slashes. For example:

// This is a remark.

or

/**<author>
this is statement one.
this is statement two.
</author>*/

Importing and exporting JavaScript
You can copy text from any file into the current JavaScript object. Click in the script area where you want to insert the text, then use the File - Import menu command to choose a file. Files with a JS extension are displayed by default. To see other file types, enter *.* in the File name box. Select the file and click Import.

You can also copy JavaScript to a file by using the File - Export menu command. Select an existing file or type a new file name with the JS extension and click Export.

Note: Import and Export for JavaScript is new with Release 6.

Writing LotusScript in the Programmer’s pane
The Programmer’s pane allows you to:
• Use the placeholder [Globals] to define subprograms and declare variables that are available to all objects in the current document
• Create an additional script in LotusScript
• Complete LotusScript class statements automatically
• Complete a LotusScript block statement automatically
• Complete a LotusScript %directive automatically
- Compile scripts
- Import and export LotusScript to files

**Defining global variables and subprograms**

The Programmer’s pane contains a placeholder on the Objects tab called (Globals) that allows you to define global variables and subprograms using LotusScript. To define a global variable or subprogram perform these steps:

1. Click (Globals) on the Objects tab.
2. Select (Options), (Declarations), Initialize, or Terminate from the Objects tab.
3. Enter your LotusScript variable or subroutine in the Script area.
4. Refer to the subprogram or variable in any script in the application.

**Creating an additional script in LotusScript**

A number of scripts, such as Initialize, Terminate, and Click, are automatically defined as events for the current Domino object. You select them from the Info List. You can create additional scripts and other block structures as shown in the following table. When a new script is added, it appears on the Objects tab.

<table>
<thead>
<tr>
<th>What you type</th>
<th>What LotusScript does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class name</td>
<td>Adds an empty Class/End Class block called <em>name</em> to the end of the (Declarations) script for the current object. Positions the insertion point in the Class statement.</td>
</tr>
<tr>
<td>DefType letter_range</td>
<td>Adds a DefType statement to the end of the (Options) script for the current object. Positions the insertion point at the DefType statement.</td>
</tr>
<tr>
<td>Dim name</td>
<td>If the Dim statement is typed outside of a subprogram, adds a Dim statement for <em>name</em> to the end of the (Declarations) script for the current object. Positions the insertion point at the Dim statement.</td>
</tr>
<tr>
<td>Function name</td>
<td>Creates an empty Function/End Function block called <em>name</em> for the current object and adds it to the list of events. Positions the insertion point in the function.</td>
</tr>
<tr>
<td>Option keyword</td>
<td>Adds an Option statement to the end of the (Options) script for the current object. Positions the insertion point at the Option statement.</td>
</tr>
<tr>
<td>Property Get name</td>
<td>Creates an empty Property Get/End Property block called <em>name</em> for the current object and adds an item called <em>name</em> Get to the list of events. Positions the insertion point in the subprogram.</td>
</tr>
<tr>
<td>Property Set name</td>
<td>Creates an empty Property Set/End Property block called <em>name</em> for the current object and adds an item called <em>name</em> Set to the list of events. Positions the insertion point in the subprogram.</td>
</tr>
<tr>
<td>Sub name</td>
<td>Creates an empty Sub/End Sub block called <em>name</em> for the current object and adds it to the list of events. Positions the insertion point in the subroutine.</td>
</tr>
<tr>
<td>Type name</td>
<td>Creates an empty Type/End Type block called <em>name</em> for the current object and adds it to the end of the (Declarations) script for the object. Positions the insertion point at the beginning of the Type block.</td>
</tr>
</tbody>
</table>
Completing a LotusScript block statement automatically

For LotusScript the following block structures are automatically terminated and the insertion point is placed in a new, indented line within the block structure.

<table>
<thead>
<tr>
<th>This statement</th>
<th>Automatically terminates this statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Forall</td>
<td>Forall</td>
</tr>
<tr>
<td>End If</td>
<td>If...Then</td>
</tr>
<tr>
<td>End Select</td>
<td>Select Case</td>
</tr>
<tr>
<td>Loop</td>
<td>Do, Do While, or Do Until</td>
</tr>
<tr>
<td>Next</td>
<td>For</td>
</tr>
<tr>
<td>Wend</td>
<td>While</td>
</tr>
</tbody>
</table>

Completing a LotusScript %directive automatically

If a script contains a %Rem directive without a matching %End rem directive, LotusScript inserts a new line containing %End rem immediately below the %Rem directive when the Script area is closed or saved, or when a different object or script is selected.

Compiling LotusScript

Compilation takes place in two phases:
1. A partial compilation occurs as you enter the script. Syntax and other per-line errors are reported at this time.
2. A complete compilation occurs when you save the script. All remaining compile-time errors are reported at this time.

If you attempt to save a script with compilation errors, you are given a choice of editing the script again or exiting without saving your changes. If you exit, all changes are lost.

If you want to save a script with errors, make the incorrect line a remark by preceding it with an apostrophe or a REM statement, or setting it off with a %REM ... %END REM directive.

Errors not detectable during compilation are reported at run time.

If you change (Globals) on a form, all scripts associated with the (Globals) script, with the form, and with all objects on the form are recompiled. Scripts in shared fields and subforms are not recompiled since they aren’t on the form.

If you change the script for an object on a form, all events associated with the object, but nothing else, are recompiled.

For more information on recompiling LotusScript, see "Recompiling all LotusScript."

Importing and exporting LotusScript

You can copy text from any file into the current LotusScript event or object. Click anywhere in the script area then use the File - Import menu command to choose a file. Files with an LSS extension are displayed by default. To see other file types, enter *.* in the File name box. Click Import. If the file you want to import has a LotusScript object with the same name as the current object, the LotusScript Import Options dialog box is displayed. Choose Yes to replace the existing script with the imported script.

You can also copy LotusScript to a file by choosing File - Export. Select an existing file or type a new file name with the LSS extension and click Export. From the LotusScript Export Options dialog box select "Current section only" to export the current script, view, form, or agent object. Select "Current object
only” to export all scripts in the current object, for example, just this button or action. Select “All objects” to export all scripts in the current database. Click OK to copy the scripts to the file.

Exploring the LotusScript Debugger

The debugger allows you to:
- Start and stop the debugger
- Select a subprogram
- Step through a script
- Debug with breakpoints
- Use the debugger utilities
- Examine and set the values of the script variables

The debugger has the following components.

<table>
<thead>
<tr>
<th>Interface element</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakpoints tab</td>
<td>Displays a list of breakpoints set in the script.</td>
</tr>
<tr>
<td>Calls tab</td>
<td>Displays subprograms currently on the execution stack. <strong>Note:</strong> Starting in Release 6, this tab replaces the Calls box.</td>
</tr>
<tr>
<td>Close Debugger button</td>
<td>Stops the script currently being debugged and terminates the debugging session. <strong>Note:</strong> This button is new with Release 6.</td>
</tr>
<tr>
<td>Continue button</td>
<td>Runs the current script until an error is found.</td>
</tr>
<tr>
<td>Debug pane</td>
<td>Displays the script currently being debugged -- an Object box and an Event box.</td>
</tr>
<tr>
<td>Event box</td>
<td>Displays the event containing the script.</td>
</tr>
<tr>
<td>New Value box</td>
<td>Allows you to enter a new value for any variable listed on the Variables tab.</td>
</tr>
<tr>
<td>Object box</td>
<td>Displays the name of the object containing the script.</td>
</tr>
<tr>
<td>Output tab</td>
<td>Displays the output of the script.</td>
</tr>
<tr>
<td>Interface element</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Script pane</td>
<td>Displays the script currently being debugged.</td>
</tr>
<tr>
<td>Step Exit button</td>
<td>Exits the current subprogram.</td>
</tr>
<tr>
<td>Step Into button</td>
<td>Steps through a script one line at a time.</td>
</tr>
<tr>
<td>Step Over button</td>
<td>Executes the current statement and steps to the next statement, or steps over a subprogram called by the current statement.</td>
</tr>
<tr>
<td>Stop button</td>
<td>Stops all scripts that are at a breakpoint and closes the debugger.</td>
</tr>
<tr>
<td>Utilities pane</td>
<td>Contains the Breakpoints, Calls, Output, and Variables tabs.</td>
</tr>
<tr>
<td>Variables tab</td>
<td>Displays the variables in the current script.</td>
</tr>
</tbody>
</table>

### Using the LotusScript Debugger

When you run a script in Debug mode, the script is in one of three states:

- When a script is **interrupted** at a breakpoint, the debugger has control.
- When a script is **stepping**, control passes to the script and then back to the debugger after a single statement in the script is executed.
- When a script is **continuing**, it runs uninterrupted until a breakpoint is reached. If you do not set any breakpoints in a script, it runs as if the debugger were not present.

**Note:** The debugger does not run on the Terminate event.

### To use the LotusScript Debugger

1. To enable the debugger, do one of the following:
   - Choose File - Tools - Debug LotusScript
   - Click the "Debug LotusScript" icon on the toolbar
   The message "LotusScript debugging started" appears on the status line.

2. Perform the action that starts the script, like clicking a button or choosing an action. When you are in Debug mode and the script runs, execution pauses at the first line of the script and the debugger opens.

3. To adjust the windows, drag the horizontal bar separating the panes as desired.

4. To disable the debugger, do one of the following:
   - Click the Close Debugger box
   - Choose File - Tools - Debug LotusScript again
   - Click the "Debug LotusScript" icon on the toolbar again
   The message "LotusScript debugging terminated" appears on the status line.

### Stopping script execution

To stop script execution while the debugger is open, click Stop. All scripts that are at a breakpoint are stopped as if the end of the scripts were reached, and the debugger closes.

### Selecting a subprogram

As you execute a script the current subprogram appears in the debugger window.

The Calls tab contains a list of the subprograms currently in the execution stack in order of execution, with the currently executing subprogram at the top of the list. Subprograms are listed as `object: event`, for example, `Calculate_totals: CLICK`.  

---

54  Programing Guide, Volume 1: Overview and Formula Language
If you select a subprogram from the list, its script appears in the debugger window. If you select the subprogram that is currently executing, the current pointer points to the statement about to be executed. If you select another subprogram, the current pointer points to the statement that calls the next subprogram in the stack.

**Stepping through a script**

The debugger provides the following facilities for stepping through LotusScript:

- **Step into a subprogram**
  To execute the current statement and step to the next statement, or step into the subprogram if the current statement calls a subprogram, click Step Into, choose Debug - Step Into, or press F8.

  Step Into proceeds to the next statement in the program. If the current statement calls a subprogram, the debugger displays the script for the subprogram and sets the current line to the first executable statement in the subprogram. If no source script is available for the subprogram (because it is an external file), Step Into behaves the same as Step Over.

- **Step over a subprogram**
  To execute the current statement and step to the next statement, or step over the subprogram if the current statement calls a subprogram, click Step Over, choose Debug - Step Over, or press SHIFT+F8.

  Step Over proceeds to the next statement in the current program unit. If the statement calls a subprogram, the debugger executes the entire subprogram as if it were a single statement and sets the current line to the next statement in the calling program unit.

- **Exit from a subprogram**
  To execute the remaining statements in a subprogram and step to the next statement in the calling program unit, click Step Exit, choose Debug - Step Exit, or press CTRL+F8.

  Step Exit continues executing the current subprogram and stops in the subprogram that called it at the line following the call. If the subprogram was not called by another, execution continues to the next breakpoint or to completion.

**Debugging with breakpoints**

A breakpoint interrupts script execution just before the statement at which the breakpoint is set. While script execution is interrupted, you can examine and modify the values of variables and use other debugger commands. Breakpoints cannot be set for the Terminate event.

After you set a breakpoint, you can permanently clear it, temporarily disable it, or enable it again. Breakpoints are displayed as red stop signs

- 😋 when enabled and red stop signs with yellow slashes

- 😋 when disabled.

For a statement continued over multiple lines, the last line is highlighted during stepping and stopping on a breakpoint. To set, disable, enable, or clear a multi-line statement, you must select the last line.

A breakpoint remains in a script until you explicitly clear it. When an executing script first displays in the debugger, any breakpoints that were not cleared from the last debugging session reappear.

**Note:** Persistent breakpoints are new with Release 6.

The debugger provides the following breakpoint facilities:

- **Set a breakpoint**
Select a statement at which no breakpoint is currently set. Double-click the statement, choose Debug - Set/Clear Breakpoint, or press F9.

- **Clear a breakpoint**
  Select a statement at which a breakpoint is currently set. Double-click the statement once if the stop sign has a yellow slash or twice if the stop sign is solid red, choose Debug - Set/Clear Breakpoint, or press F9. To clear all breakpoints from all scripts in the active document, choose Debug - Clear All Breakpoints.

- **Disable a breakpoint**
  Select a statement at which an enabled (solid red stop sign) breakpoint is set. Double-click the statement or choose Debug - Disable Breakpoint. To disable all breakpoints from all scripts in the active document, choose Debug - Disable All Breakpoints or press SHIFT+F9.

- **Enable a breakpoint**
  Select a statement at which a disabled (red stop sign with yellow slash) breakpoint is set and choose Debug - Enable Breakpoint. To enable all breakpoints from all scripts in the active document, choose Debug - Enable All Breakpoints.

- **Continue script execution**
  To start executing the current script or to resume execution after the script is interrupted at a breakpoint, click Continue, choose Debug - Continue, or press F5.

### Using the debugger utilities

The debugger utilities appear on the tabbed panels in the Utilities pane of the debugger:

- Breakpoints tab
- Variables tab
- Output tab
- Calls tab

#### Using the breakpoints panel

Click the Breakpoints tab or choose Debug - Breakpoints to access the breakpoints panel.

The breakpoints panel displays the breakpoints in the following format:

```
object: event: line
```

If the breakpoint is disabled, (Disabled) is appended to the display.

#### Using the variables panel

Click the Variables tab or choose Debug - Variables to access the variables panel.

The variables defined for the procedure appear in a three-column display, showing the name, value, and data type of each variable. To view array or type members, click the arrow to the left of the variable name.

#### To change the value of a variable

1. Select the variable.
2. Enter the value in the New Value box.
3. Click the green check mark.

#### Using the output panel

Click the Output tab or choose Debug - Output to access the output panel.

The output panel displays script output, for example, the content of a Print statement. You can:

- View the output.
• Clear the output panel by clicking Clear All.
• Click Copy to move selected output to the clipboard. Choose Edit - Select All to select all of the text in the output panel.

Using the calls panel
Click the Calls tab to access the calls panel. For more information about the calls panel, see the topic "Selecting a subprogram” in this chapter.

Using the Remote Debugger
You can use the remote debugger to step through and debug LotusScript agents running on the server. The agent you want to debug must be running at the time that you start the remote debugging tool.

More than one user can attach to and debug the same agent. However, only one user has control to debug it at any given time. A second user may get control by attaching to the same agent. This capability is useful, for instance, if you want help from a coworker in debugging an agent. You can ask the coworker to attach to and debug the server agent and that coworker can do so without having to be in the same physical location as you. Note that if you attempt to continue debugging the server agent after your coworker has attached to and begun to debug it, you will get the error, "Another user has attached to the agent and taken control. Please try to reconnect for further debugging."

Complete these steps before using the remote debugger:
1. Enable remote debugging on the server
2. Enable remote debugging in the agent
3. Start the remote debugger

"To debug a scheduled LotusScript agent remotely” defines how to use the remote debugger once the server and agent are set up to allow for remote debugging.

Note: Remote debugging is new with Release 6.

To enable remote debugging on the server
1. Before starting the server, in the server’s notes.ini file, add the following value to the ServerTasks= list:
   Rdebug
   If the server is already running, to begin the remote debugging task enter the following at the server console:
   load rdebug
2. From the Server Tasks tab of the server document, select the far right tab entitled Remote Debug Manager. Set Allow remote debugging on this server to "Enabled."

   Tip: You must have administrative access to the server to edit this field.
3. Enter a time limit for running the rdebug task on the server in the Turnoff Server Debug after setting.
   Tip: If you set this value to -1, the task can run forever.
4. Enter a time interval in the Agent Wait at Start Time field if you want to force the agent to pause before running, giving you time to attach the remote debugger to the target agent. (Recommended.)
5. Check the Ports - Internet Ports - Remote Debug Manager tab, to ensure that the TCP/IP port status property is set to "Enabled."
6. Make sure you are listed on the server’s Security tab as having permission to run agents.
7. Save and Close the server document.
To enable remote debugging in the agent

1. On the Basics tab of the Agent Properties box, select the On schedule Trigger radio button. Click Schedule.
   The Agent Schedule dialog box displays.
2. Select the server from the "Run on" list box.
3. Add a Stop statement to the beginning of the Initialize event code in the agent. The Stop statement works as follows:
   - If the remote debugger task is not running on the server or if it is not enabled, this statement does nothing. Performance of the agent is not affected.
   - If the remote debugger is running and enabled, it pauses the execution of the agent code for the time specified in the "Agent Wait at Start Time" field if you specified one in the Server Tasks tab - Remote Debug Manager tab of the server document. This gives you time to attach to the agent in the Remote Debugger window before it begins executing.
   - If the Remote Debugger window is open and you have attached to the agent, the Stop statement acts like a break point and stops execution.
   
   **Tip:** You can also add the following statement to the start of the agent code to broadcast to the console that the agent is preparing to run.
   ``` |
   Print "AgentName is about to start running***********"
   ```

To start the Remote Debugger

1. To run a specific agent you want to debug, either:
   - Increase the run frequency of the agent by clicking the Schedule button on the Basics tab of the Agent Properties box. In the Agent Schedule dialog box, change the hour value of "Run agent every" to zero and the minutes value to 5.
   - Enter the following command on the server console:
     ``` |
     tell amgr run databaseName.nsf 'agentName'
     ```
2. While the agent you want to debug is running on the server, select File - Tools - Remote Debugger from the menu.
   
   **Tip:** You know an agent is running if you added print and stop statements at the beginning of its Initialize event. The print statement cause the agent to print to the console when it is starting and the stop statement delays its execution.
   The Domino Debugger 6 welcome page opens.
3. Select File - Select Debug Target from the menu.
   The Select Debug Target dialog box displays.
4. Choose the server that is running the agent from the Server list box and click Open.
   A list of the databases on the server displays.
5. Select the database that contains the agent and click Open.
   If the agent you want to debug is currently running, it appears in the Debug Target box.
6. Select the agent you want to debug from the Debug Target list and click Open.
   The Script Debugger window opens, displaying the (Options) event.

To debug a scheduled LotusScript agent remotely

1. To display the agent code you want to debug in the Script pane, select the agent or script library name from the Object list box and the name of the event that triggers it or the subroutine from the Event list box on the Debug pane.
The code displays in the Script pane. If the script has already begun executing, a yellow arrow points to the line of code that is the next to be executed.

2. Select the mode with which you want to debug the code using one of the following action buttons:

   **Note**: To set a breakpoint, highlight the line you want to break before executing and choose File - Set/Clear Breakpoint or F9.

<table>
<thead>
<tr>
<th>Action Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break</td>
<td>Breaks execution of the agent code on whichever line is currently executing, including lines within loops. Does not close the debugger. You can click Continue to resume execution.</td>
</tr>
<tr>
<td>Continue</td>
<td>Runs the code in the Script pane and stops only when it encounters a breakpoint. Otherwise, runs until the agent code is complete.</td>
</tr>
<tr>
<td>Step Exit</td>
<td>If the debugger is executing code within a subprogram, exits the subprogram and returns to the statement that called it. If no statement in the current agent code called the subprogram, continues executing until it reaches the next breakpoint or the end of the agent code.</td>
</tr>
<tr>
<td>Step Into</td>
<td>Steps through the code one line at a time and drops into subroutines or functions. <strong>Note</strong>: Use this button repeatedly to step through each line of the script one line at a time.</td>
</tr>
<tr>
<td>Step Over</td>
<td>Steps through the code one line at a time and does not drop into subroutines or functions, though the sub or function code is executed.</td>
</tr>
<tr>
<td>Stop</td>
<td>Disconnects the remote debugging server task; the agent code stops executing.</td>
</tr>
</tbody>
</table>

3. As the code executes, the debugging results display in the Utility pane. The Utility pane contains the following tabbed sections:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakpoints</td>
<td>Lists the locations of breakpoints you have added to the code.</td>
</tr>
<tr>
<td>Calls</td>
<td>Identifies what is currently being executed in the agent and displays all subprograms currently on the execution stack. For example, <code>folderDocs:INITIALIZE 5</code> indicates that the fifth line of the Initialize event in the folderDocs agent is the next line to execute.</td>
</tr>
<tr>
<td>Output</td>
<td>Displays the server console output. Any print statements in the code or error messages appear here.</td>
</tr>
<tr>
<td>Variables</td>
<td>Lists the variables created and their values as values are applied to them in the code. You can use the edit box at the bottom of the pane to change the value of the selected variable.</td>
</tr>
</tbody>
</table>

---

**Debugging Java code remotely**

You can debug Java code running under control of a Notes client JVM (Java Virtual Machine) with a debugger that supports the JPDA (Java Platform Debugger Architecture), such as Eclipse. See java.sun.com/products/jpda for information on the JPDA. See www.eclipse.org for information on Eclipse and the Eclipse debugger. You cannot debug Java code running on a Domino server.

Complete these steps to debug Java code:

1. **Enable Java debugging on a Notes client**
2. **Enable Java debugging in an agent, Web service, or script library**
3. **Connect a debugger to the JVM**
Note: Debugging Java agents is new with Release 7.

CAUTION:
Java debugging is insecure and degrades performance. Disable Java debugging when it is not needed.

To enable Java debugging on a Notes client
The Notes client supports Java debugging in the following contexts. Each context has its own JVM. Only one user can debug at a time in each context.
- Foreground -- Java code that runs in the Notes client interactively, for example, an agent triggered from the Actions menu.
- Background -- Java code that runs in the Notes client under control of the task loader, for example, a locally scheduled agent.
- Web preview -- Java code being previewed in a browser through Domino Designer, for example, an applet on a form.

Java code from a script library runs in the context of the calling code.

To enable and disable Java debugging on the Notes client:
1. Choose File - Tools - Java Debugging Preferences. This brings up the “Java Debugging Preferences” dialog.
2. To enable foreground debugging, select "Client Agents/Applets" and specify a port number to connect the Notes and debugger computers. Deselect to disable.
3. To enable background debugging, select "Locally Scheduled Agents" and specify a port number to connect the Notes and debugger computers. Deselect to disable.
4. To enable Web preview debugging, select "Http Preview" and specify a port number to connect the Notes and debugger computers. Deselect to disable.

Pick a port number that you think is free. This may require trial and error.

Java debugging is disabled by default.

If changes are made to the foreground or background preference, Notes must be restarted. If changes are made to the Web preview preference, the preview must be restarted.

To enable Java debugging in an agent, Web service, or script library
To enable Java debugging, go to the Security tab of the Agent or Web Service properties box, or to the Script Library properties box, and check:

Compile Java code with debugging information

If you are compiling the Java to be debugged outside Domino Designer, compile it with the -g option.

After modifying an agent or Web service, export the source code to a file that is accessible to your debugger.

To connect a debugger to the JVM
The following instructions assume the Eclipse debugger. You have to interpolate for other debuggers.
1. Start the Notes client containing the agent or Web service to be debugged with Java debugging enabled.
2. Start the debugger.
3. Create a Java project and switch to the Java perspective. For example: choose File - New - Project; select "Java" in the Project window; enter a name for the project; select "Finish"; switch to the Java perspective. Or open the project if it already exists.
4. Import the source Java files. For example: right-click on the project folder in the left-hand pane; select "Import"; browse to and select the source files; import them.
   Source code that is not imported is not available to the debugger. You can see the execution thread and can access variables, but you cannot see the source code and set breakpoints.

5. In Notes, start the agent or Web service to be debugged.

6. In Eclipse, attach the debugger to the Notes JVM. For example, choose Run - Debug; enter a name for the debug configuration; select the "Connect" tab; select the project, enter the host name or address of the Notes computer, and enter the Java debug port number on the Notes client; select "Apply" (only necessary the first time or if you change the configuration); select "Debug." The host address is 127.0.0.1 if the Notes client and Eclipse are on the same machine.

After you attach the debugger, you will see the execution threads for the JVM. An agent’s thread looks something like this:

Thread(AgentThread: JavaAgent)(Running)

You can suspend the agent’s thread to gain control over it. At this point, the usual debugger features become available (setting breakpoints, stepping, continuing, viewing variable values, and so on).

The Java agent must run long enough for you to attach the debugger to the JVM. You might delay execution at the beginning of the procedure, for example, by inserting a loop with sleep statements.

If Notes and the debugger are on separate computers, and you get the error "Failed to connect to remote VM," the designated port is probably taken by another process. Try specifying another port number and restarting Notes.

**Security considerations**

Java debugging over a network is insecure. You should restart Notes or the Web preview without Java debugging enabled when you’re not debugging.

**Using Script Libraries**

Script libraries can contain LotusScript, Java, or JavaScript.

The scope of a script library is the current database. All scripts in a database can avail themselves of the LotusScript, JavaScript, or Java in a library in that database. However, the library is lost to scripts outside the database. For example, if a button in a document uses a script library and you mail the document to or paste it into another database that does not have the same script library, the script fails. If a script attempts to use a library not in the current database, the error message, "Error loading USE or USELSX module" displays.

**Note:** The use of JavaScript in script libraries is new with Release 6.

**To create a script library**

1. Choose Create - Design - Script Library
2. Choose one of the following:
   - LotusScript Library
   - Javascript Library
   - Java Library
   The Programmer’s pane displays the empty script in the Work pane.
3. To name the library, choose Edit - Properties.
Each library in a database must have a unique name. This is true for multilingual databases as well; you cannot differentiate between multiple script libraries of the same name using the $Language field of the design note.

4. Enter the script in the Script pane.
   For LotusScript, first select an existing script; it can be an (Options), (Declarations), Initialize, Terminate, or user script and either:
   • Enter new or change existing code for the script.
   • To write your own LotusScript script, replace the title of the script following the Sub or Function statements. For example, to write a script called Once, change Sub Initialize to Sub Once.
     The editor automatically creates a new script using this new title.

5. Choose File - Save, then File - Close to save and close the library.

See the following topics for instructions on incorporating a library:

• Incorporating a LotusScript script library
• Incorporating a Java script library
• Incorporating a JavaScript script library

To access an existing script library

1. From the Design pane, expand Shared Code and select Script Libraries.
   A list of script libraries displays in the Design list.

2. Select the script library you want to view.
   The script library appears in the Programmer’s pane.

Incorporating a LotusScript script library

When you use a LotusScript script library, the script in the (Options), (Declarations), Initialize, and Terminate events of the library become available as though they were in the current object’s corresponding scripts. User scripts in the library become available as though they were in the current object.

To incorporate a LotusScript script library

• Enter a LotusScript Use statement in the (Options) script for the object or for the (Globals) object.
  For example, to make the script library named Market2 available to a form’s scripts, enter either of the following statements in the (Declarations) script for the form:
  – Use “Market2”
  – Const m2 = “Market2” Use m2
  The name is case insensitive and should not contain spaces.

You cannot change a declaration in a script library while a script or agent using that declaration is open. You must first close the script or agent using the declaration, make the change in the library script, and then reopen the script using the declaration. If you do make a change while a script is open, you must comment out or delete the script that uses the declaration, close and reopen the script, and then reinsert or remove the comments from the script.

See “Recompiling all LotusScript” for details on how you can find and update scripts that incorporate libraries you have made changes to.

Incorporating a Java script library

Using a Java script library you can define common classes that you can then access from any Java agent.
To use a Java script library in a Java agent
1. From the Java agent programmer’s pane, click Edit Project.
   The Organize Java Agent Files dialog box displays.
2. From the Browse box, select Shared Java Libraries.
3. Select the library you want to include.
4. Click Add/Replace File(s).
   The library displays in the Current Agent Files list.
5. Click Ok.

Incorporating a Javascript script library
Using a Javascript script library enables you can share Javascript functions across multiple design
elements, such as forms and pages.

You can incorporate the following types of JavaScript script libraries:

<table>
<thead>
<tr>
<th>Type:</th>
<th>In the resulting HTML:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline</td>
<td>Inserts a &lt;SCRIPT&gt; tag that is executed when the document is loaded.</td>
</tr>
<tr>
<td>Associated with JS Header object</td>
<td>Displays within the &lt;HEAD&gt; tag where the JS Header object is executed.</td>
</tr>
</tbody>
</table>

To incorporate a Javascript script library
1. Open the design element you want to add the library to in the Work pane.
2. Place the cursor where you want to incorporate the library. For:
   - Inline script libraries, click the design element’s background
   - A script that should be associated with the JS Header object, select JS Header from the Objects list
     in the Programmer’s pane
3. Choose Create - Resource - Insert Resource from the menu.
   The Insert Resource dialog box displays.
5. Select the library from the Available resources list and click Ok.
   If you created an inline script library, a script icon displays on the design element.
   If you created a script library associated with the JS Header object, a button displays in the
   Programmer’s pane labeled with the script library’s name.

Recompiling all LotusScript
The Recompile all LotusScript feature enables you to recompile every piece of LotusScript code contained
in a database.

Being able to recompile all the LotusScript code in a database is especially useful if, for instance, you
want to make a change to a LotusScript script library that you have incorporated into several design
elements across a database. By recompiling all LotusScript, you can retrieve a list of all the design
elements that incorporate the now incorrect code and use your time to fix the code rather than to try to
find coding errors and inconsistencies.

Note: This feature is new with Release 6.

A recompile takes two passes at a database. The first pass identifies all the elements that contain
LotusScript code and need to be recompiled. The second pass recompiles all the elements identified in the
first pass and saves them again. It then displays a list of those elements in which it found scripting
errors. You can open each element from the list to find and fix its errors.
To recompile all the LotusScript code in a database

1. From the Designer menu, select Tools - Recompile All LotusScript.

   Tip: The Recompile option of the Tools menu is not enabled if you do not have a specific database or a design element from a specific database selected.

   The Compile all LotusScript in dbname dialog box displays and begins recompiling the LotusScript code in the database. (dbname is the name of the current database)

   Tip: To stop recompilation, press CNTRL+BREAK, then Cancel. Canceling the operation may leave some LotusScript objects in an inconsistent state.

2. If none of the LotusScript code in the database has any errors, the Design Elements list displays all the elements that were compiled and "All code successfully compiled" displays below it. Click Ok to close the dialog box.

3. If any errors are found, the Design Elements list displays the elements that contain errors and "Not all elements compiled successfully" displays below it. Either:
   - To close the dialog box without fixing the incorrect code, click Cancel.
   - To edit an element, select it from the list and click Ok.

   The selected element displays in the programmer's pane.

4. Click each script object to display its associated code and find the error.

   The text of the line of code that contains the error displays in red and the error is defined in the Errors box. You cannot save and close the object until all errors present are corrected.

   Tip: If you want to save a script with errors, make the incorrect line a remark by preceding it with an apostrophe or a REM statement, or setting it off with a %REM ... %END REM directive.

5. Once you have fixed and closed the element, rerun the tool to display any elements that contain remaining errors.

6. Repeat the process until no more errors are reported.

Writing formulas in the Programmer’s pane

Formulas may be written in the Programmer’s pane, or in the Formula window located in the properties boxes of some objects. Formula windows are also located in other parts of the Domino interface such as Replication Settings.

To enter a formula in the Programmer’s pane

1. Select the correct object on the Objects tab. The object can be an action, a button, a hotspot, a field, or the form itself.

2. Choose Formula from the Run menu.

3. For fields, click the correct event on the Objects tab. The event can be Default Value, Input Translation, Input Validation, Input Enabled, or HTML Attributes.

4. Enter the formula in the Script area. If auto completion is enabled a pop-up list appears. You can select a formula from the list and press ENTER to paste it in the Script area, or continue to type the formula manually.

   Note: You can also click the Reference tab to see a list of @functions, @commands, and fields.

5. Click the check mark that appears on the right just above the Script area to accept your new formula or the edits to an existing formula. You can click the X to cancel your formula or edits.

6. If you receive syntax error messages, edit the formula until the errors are corrected.

The column formula has the appearance of a Programmer’s pane but lacks an Object box. Your formula applies to the column that is currently selected.
Using the formula window
The Formula Window appears anywhere you can enter a formula outside of the Programmer’s pane. When entering a formula in the Formula window, you can:
- Enter it directly into the small white area on the Hide When tab of a properties box.
- Click Formula Window or Zoom to open a larger Formula Window.

To enter a formula in the small Formula window
1. Open a screen containing a Formula Window.
2. Enter your formula in the Formula window. A check mark
   [✓]
   and an X
   [✗]
   appear next to the Formula window when you begin typing. Click the check mark to accept any changes you have made. Click the X to undo any changes made prior to clicking the check mark.
3. If you receive syntax error messages, edit the formula until the errors are corrected.

To enter a formula in the large Formula window
1. Open the properties box of a design element.
2. Select the Hide When tab.
3. Select Formula Window to display a larger Formula window.
4. To select @functions and fields from a list, select Formula Window, then select "Fields and Functions."
5. Type your formula.
6. Click Done.
7. If you receive syntax error messages, edit the formula until the errors are corrected.

To delete a formula, highlight the formula and press DEL.

Using the Programmer’s pane for Simple action(s)
To create a Simple action in the Programmer’s pane:
1. Select Simple action(s) from the Run menu.
2. Click Add Action.
3. Select an action from the list.
4. Enter any additional information. This varies depending on the action selected.

Note: The Reference tab is not available in Simple action(s) mode.
Chapter 3. Programming Domino for Web Applications

You should be aware of certain procedures, restrictions, and enhancements available to you when using Lotus Domino Designer to program Web applications. This section describes the following areas:

- Formula language
- Web agents
- JavaScript
- Web services

For additional information, see "URL commands for Web applications" in Application Development with Domino Designer.

Formula language

The formula language works, with restrictions, for Web applications. The formula language is particularly useful for implementing Notes client menu commands as buttons, hotspots, and actions in the browser environment where the Notes client menu is not available. This section describes:

- Where formulas work on the Web
- @Functions on the Web
- @Commands on the Web

Where formulas work on the Web

The following table outlines where formulas work on the Web.

<table>
<thead>
<tr>
<th>Type of formula</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Agent</td>
<td>From a browser, you can start an agent with the @commands [ToolsRunMacro] or [RunAgent] or with the URL command. OpenAgent. The agent runs on the Domino server, not in the browser.</td>
</tr>
<tr>
<td>Column</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Computed field value</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Computed text</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Default value</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Event</td>
<td>The only formula events that work in a browser are WebQueryOpen and WebQuerySave in forms, and these events are restricted to executing the @commands [ToolsRunMacro] or [RunAgent] (which run an agent on the Domino server).</td>
</tr>
<tr>
<td>Form</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Hidden paragraph</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Hide action</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Hide column</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Hotspot</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Input translation</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Input validation</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Insert subform</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Type of formula</td>
<td>Application</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Keyword field</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Named element</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Replication</td>
<td>Applies to the back-end database.</td>
</tr>
<tr>
<td>Section access</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Section title</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Selection</td>
<td>Works in a browser.</td>
</tr>
<tr>
<td>Toolbar button</td>
<td>Applies only to the Notes client.</td>
</tr>
<tr>
<td>Window title</td>
<td>Works in a browser.</td>
</tr>
</tbody>
</table>

@Functions on the Web

Several @functions are particularly useful in Web applications. However, some @functions are restricted.

- Getting client information (@ClientType and @BrowserInfo)
- Opening a URL (@URLOpen)
- Getting the database name (@WebDbName)
- Getting and setting request-header fields (@GetHTTPHeader and @SetHTTPHeader)
- Getting the URL command (@UrlQueryString)
- Manipulating the URL (@URLDecode and @URLEncode)
- Field validation (@Failure and @Success)
- Linking to the next and previous pages (@DbCommand)
- Restricted @functions

Getting client information

The @ClientType function returns "Web" from a browser, "Notes" from a Notes client, and "None" from an agent. Here's an example of a computed text formula:

```plaintext
@if(@ClientType = "Web"; "You are running from a browser"
   @ClientType = "Notes"; "You are running from a Notes client"
   "You are not running from a browser or a Notes client")
```

The @BrowserInfo function returns information about a browser depending on the parameter value. The following example gets the browser type and platform if the user is running from a browser:

```plaintext
@if(@ClientType = "Web";
   "You are running from a " + @BrowserInfo("BrowserType") +
   " browser on " + @BrowserInfo("Platform")),
@if(@ClientType = "Notes";
   "You are running from a Notes client on " + @Platform;
   "You are not running from a browser or a Notes client")
```

Opening a URL

The @URLOpen function opens the Web page specified by a URL. This example opens lotus.com:

```plaintext
@URLOpen("http://www.lotus.com")
```

In Web applications, you must specify the parameter. You cannot bring up the URL Open dialog box.

Getting the database name

The following formula gets the name of the current database and adjusts it for use in a URL:

```plaintext
@WebDbName
```

@WebDbName substitutes a forward slash for a back slash, and "%20" (hexadecimal 20) for a space. It is equivalent to:
Getting and setting request-header fields

The @GetHTTPHeader function returns the value of an HTTP request-header field. This example returns the value of the Host field:

```plaintext
@GetHTTPHeader("Host")
```

The @SetHTTPHeader function sets the value of an HTTP request-header field. This example sets the value of the Set-Cookie field:

```plaintext
@SetHTTPHeader("Set-Cookie":"COOKIE1=4646")
```

Note: These @functions are new with Release 6.

Getting the URL command

The following formula returns in a text list the URL command responsible for the current page and any parameters:

```plaintext
@UrlQueryString
```

To get only the value of a parameter, specify the parameter name, for example:

```plaintext
@UrlQueryString("Category")
```

Note: This @function is new with Release 6.

Manipulating the URL format

@URLDecode and @URLEncode format any special characters in the URL, such as spaces or punctuation, so that the URL can be displayed as readable text or passed to a computer.

The following formula would return the readable text string "

```plaintext
@URLDecode ("Domino";"http://www.acme.com/4Q%20profit%26loss")
```

To return an encoded URL that can be used as an address, do not encode an entire URL, as that will remove necessary punctuation. Instead, encode only the final information, for example:

```
"http://www.acme.com/" + @URLEncode ("Domino";"4Q profit&loss")
```

Note: This @function is new with Release 6.

Field validation

@Success and @Failure work in field input validation formulas on the Web. The @Failure path causes the message specified as the parameter to appear on a new page. In the following validation formula, if the user fails to enter a value for RequiredField, the word "Gong" in bold appears on a new page:

```plaintext
@If(RequiredField=""; @Failure("<B>Gong</B>")); @Success)
```

You can make the failure page more meaningful by using more extensive HTML in the error message:

```plaintext
msg1 := "This is a required field.<br><br>
msg2 := "<a href="/" + @WebDbName + "/Main+Document?OpenForm"">
msg3 := "Click here<a> to try again.";
msg := msg1 + msg2 + msg3;
@If(RequiredField=""; @Failure(msg); @Success)
```

Linking to the next and previous pages

In a Web view, @DbCommand with "Domino" as the first parameter pages down and up:

```plaintext
@DbCommand("Domino";"ViewNextPage")
```
@DbCommand("Domino"; "ViewPreviousPage")

**Note:** When called from an action on a page or document in a Web application, @DbCommand acts on an embedded view in the same page or document.

**Restricted @functions**
The following @functions do not work or are restricted on the Web.

<table>
<thead>
<tr>
<th>@Function</th>
<th>Web restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Certificate</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DbColumn(ODBC)</td>
<td>Only works if the remote server is on the same machine as the Domino server.</td>
</tr>
<tr>
<td>@DbCommand</td>
<td>On Web only @DbCommand(&quot;Domino&quot;) is permitted.</td>
</tr>
<tr>
<td>@DbLookup(ODBC)</td>
<td>Only works if the remote server is on the same machine as the Domino server.</td>
</tr>
<tr>
<td>@DDEExecute</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DDEInitiate</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DDEPoke</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DDETerminate</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DeleteDocument</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DialogBox</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DocChildren</td>
<td>On Web only works in column formulas.</td>
</tr>
<tr>
<td>@DocDescendants</td>
<td>On Web only works in column formulas.</td>
</tr>
<tr>
<td>@DocLevel</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DocLock</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DocMark</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@DocNumber</td>
<td>On Web only works in column formulas.</td>
</tr>
<tr>
<td>@DocParentNumber</td>
<td>On Web only works in column formulas.</td>
</tr>
<tr>
<td>@DocSiblings</td>
<td>On Web only works in column formulas.</td>
</tr>
<tr>
<td>@Domain</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@EditECL</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@EditUserECL</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@Environment</td>
<td>Does not work on Web. (Use CGI variables instead.)</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>Does not work on Web. (Use CGI variables instead.)</td>
</tr>
<tr>
<td>@FontList</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@GetFocusTable</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@GetIMContactListGroupNames</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@GetPortsList</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@HardDeleteDocument</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@IsAgentEnabled</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@IsDocBeingMailed</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@IsEmbeddedInsideWCT</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@IsModalHelp</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@LaunchApp</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@MailDbName</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@Function</td>
<td>Web restriction</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>@MailEncryptSavedPreference</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@MailEncryptSentPreference</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@MailSavePreference</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@MailSend</td>
<td>[Encrypt] and [Sign] do not work on Web.</td>
</tr>
<tr>
<td>@MailSignPreference</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@PickList</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@Platform</td>
<td>On Web returns only the platform.</td>
</tr>
<tr>
<td>@Prompt</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@RefreshECL</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@Responses</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@SetEnvironment</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@StatusBar</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@TemplateVersion</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@UpdateFormulaContext</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@URLGetHeader</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@URLHistory</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@UserPrivileges</td>
<td>Does not work on Web.</td>
</tr>
<tr>
<td>@ViewShowThisUnread</td>
<td>Does not work on Web.</td>
</tr>
</tbody>
</table>

@Commands on the Web

The @commands listed below are supported on the Web; the @commands not mentioned here do not work on the Web. These @commands may have restrictions or behave differently on the Web than in the Notes client.

- CalendarFormat
- CalendarGoto
- Clear
- CloseWindow
- Compose
- EditClear
- EditDocument
- EmptyTrash
- FileCloseWindow
- FileOpenDatabase
- FileSave
- Folder
- FolderDocuments
- MoveToTrash
- NavigateNext
- NavigateNextMain
- NavigatePrev
- NavigatePrevMain
- NavNext
CalendarFormat

The [CalendarFormat @command] changes the number of days that a calendar view displays. The Web options are:

```
@Command([CalendarFormat] ; "1")
@Command([CalendarFormat] ; "2")
@Command([CalendarFormat] ; "7")
@Command([CalendarFormat] ; "14")
@Command([CalendarFormat] ; "30")
@Command([CalendarFormat] ; "365")
```

This @command is equivalent to a URL command formatted as follows:

http://host/database/universalID?OpenView&Grid=n&Date=yyyy-mm-dd

CalendarGoto

The [CalendarGoto @command] goes to a specified date in a calendar view. The following example goes to today's date:

```
@Command([CalendarGoto]; @Today)
```

This @command is equivalent to a URL command formatted as follows:

http://host/database/universalID?OpenView&Grid=n&Date=yyyy-mm-dd

Compose

The [Compose @command] creates a new document.

To create a main document in the current database, specify only the form parameter and implement Compose in a view, page, or navigator. The following example, implemented as a view action, creates a main document based on the "Main Topic" form.

```
@Command([Compose]; "Main Topic")
```
To create a response document, specify the @command as above but implement it in an open document. For example, if the following code is implemented as an action on the “Main Topic” form, opening a document based on that form and pressing the button creates a response document:

@Command([Compose]; "Response")

To create a main document in another database, specify the database and form parameters. You must specify server as an empty string because the browser cannot access another server. The following example creates a new document in "document examples.nsf" based on the “Main Topic” form in that database.

@Command([Compose]; "" : "document examples.nsf" ; "Main Topic")

These @commands are equivalent to URL commands formatted as follows:

http://host/database/view?OpenForm
http://host/database/form?OpenForm&ParentUNID=mainunid

**EditClear and Clear**

The [EditClear] and [Clear] @commands delete the current, open document:

@Command([EditClear])

or

@Command([Clear])

For Web applications, you can implement these @commands only in a form. After the deletion, a document with the text "Deleted" replaces the current document.

In the Notes client, these @commands mark documents for deletion or delete text, depending on the design element they are used in.

These @commands are equivalent to a URL command formatted as follows:


**EditDocument**

The [EditDocument] @command toggles a document between Read and Edit modes:

@Command([EditDocument])

For Web applications, you cannot use the mode and previewpane parameters.

This @command is equivalent to a URL command formatted as follows:


**EmptyTrash**

The [EmptyTrash] @command deletes the documents marked for deletion and refreshes the current view.

@Command([EmptyTrash])

This @command works for a view on the Web if "Use applet in the browser" or "Allow selection of documents" is in effect.

**FileCloseWindow and CloseWindow**

In a browser, the [FileCloseWindow] and [CloseWindow] @commands simulate closing a window by loading another page, usually the previous page displayed.

See [FileSave] for using FileCloseWindow or CloseWindow with FileSave to submit a document.

FileCloseWindow and CloseWindow do not work on the Web unless "Use JavaScript when generating pages" is set in the database properties. This is the default.
**FileOpenDatabase**

On the Web, use the `FileOpenDatabase @command` only as shown in conjunction with `OpenDocument`.

**FileSave**

The `FileSave @command` saves the current document, which must be open for editing.

To submit a document that is open for editing, issue `FileSave` followed by `OpenView`, `FileCloseWindow`, or `CloseWindow`. This can be implemented as a form action or button.

Using `FileSave` followed by `OpenView` saves the document then opens a specified view in the browser, for example:

```plaintext
@Command([FileSave]);
@Command([OpenView]; "All Documents")
```

Using `FileSave` followed by `FileCloseWindow` or `CloseWindow` saves the document then uses the `$\$Return$` field or lack thereof to determine what to open next in the browser.

```plaintext
@Command([FileSave]);
@Command([FileCloseWindow])
```

or

```plaintext
@Command([FileSave]);
@Command([CloseWindow])
```

If the document contains no `$\$Return$` field, the browser displays a page with the message, "Form processed."

If the document contains a `$\$Return$` field, the browser displays its HTML or follows a link. Typically `$\$Return$` is a "computed for display" text field. The following example presents some text followed by a view link:

```plaintext
"<h3>Document submitted</h3><hr><font size=2><a href="/" + @WebDbName + "/All%20Documents?OpenView=All%20Documents view</a>"
```

To have the browser follow a link, specify the formula for the `$\$Return$` field as a Domino URL command in brackets. The following example opens a view:

```plaintext
"[/" + @WebDbName + "/All%20Documents?OpenView]"
```

See "Customizing 'Form processed' confirmation for the Web" in Application Development with Domino Designer for more information.

FileSave does not work on the Web unless "Use JavaScript when generating pages" is set in the database properties. This is the default.

**Folder and FolderDocuments**

The `Folder` and `FolderDocuments` @commands copy or move selected documents from a view or folder to a folder.

1. This code copies the selected documents to the Favorite Stuff folder:
   ```plaintext
   @Command([Folder]; "Favorite Stuff"; "0")
   ```
   or
   ```plaintext
   @Command([FolderDocuments]; "Favorite Stuff"; "0")
   ```

2. This code moves the selected documents to the Archive folder:
   ```plaintext
   @Command([Folder]; "Archive"; "1")
   ```
   or
   ```plaintext
   @Command([FolderDocuments]; "Archive"; "1")
   ```
These @commands work on the Web only if "Use applet in the browser" is in effect for the implementing view or folder.

**MoveToTrash**

The `MoveToTrash` command marks selected documents in a view or folder for deletion.

This @command works for a view on the Web if "Use applet in the browser" or "Allow selection of documents" is in effect.

**NavigateNext and NavNext**

The `NavigateNext` and `NavNext` commands open the next document in the current view or folder.

For Web applications, you can only use these @commands on forms; you cannot use them in views or folders.

These @commands are equivalent to a URL command formatted as follows:


**NavigateNextMain and NavNextMain**

The `NavigateNextMain` and `NavNextMain` commands open the next main document in the current view or folder.

For Web applications, you can only use these @commands on forms; you cannot use them in views or folders.

These @commands are equivalent to a URL command formatted as follows:


**NavigatePrev and NavPrev**

The `NavigatePrev` and `NavPrev` commands open the previous document in the current view or folder.

For Web applications, you can only use these @commands on forms; you cannot use them in views or folders.

These @commands are equivalent to a URL command formatted as follows:


**NavigatePrevMain and NavPrevMain**

The `NavigatePrevMain` and `NavPrevMain` commands open the previous main document in the current view or folder.
or
@Command([NavPrevMain])

For Web applications, you can only use these @commands on forms; you cannot use them in views or folders.

These @commands are equivalent to a URL command formatted as follows:

**OpenDocument**
The `OpenDocument` command used in conjunction with `OpenView`, opens an existing document in the current database. The view must be sorted and the `OpenView` command must specify key exactly except for case. For example, the following code opens the first document in "Main View" whose first sorted column contains the value "one":
@Command([OpenView]; "Main View"; "one");
@Command([OpenDocument])

This differs from the Notes client in several respects:
- In the Notes client, you can specify `OpenDocument` without a preceding `OpenView` or without specifying a key, to open the current document in a view. This fails on the Web.
- In the Notes client, you can specify a partial key, for example, "o" or "on" instead of "one." The key must be exact on the Web.

For Web applications, you can open the first document in the view with the argument "$first":
@Command([OpenView]; "Main View"; "$first");
@Command([OpenDocument])

To open the document in Edit mode, specify as "1" the `writeOrReadOnly` parameter to `OpenDocument`. For example:
@Command([OpenView]; "Main View"; "one");
@Command([OpenDocument]; "1")

For Web applications, you cannot use the `UNID` and `width:height` parameters to `OpenDocument`.

The `OpenDocument` command, used in conjunction with `FileOpenDatabase`, opens an existing document in another database. This example opens the first document in the "All Documents" view in "document examples.nsf":
@Command([FileOpenDatabase]; ""; "document examples.nsf"; "All Documents"; "$first");
@Command([OpenDocument])

These @commands are equivalent to URL commands formatted as follows:

**OpenFrameset**
The `OpenFrameset` command opens a frameset.
@Command([OpenFrameset]; "WebToDoFS")

This @command is equivalent to a URL command formatted as follows:
http://host/database/frameset?OpenFrameSet&Frame=NotesView&Src=source
OpenHelpDocument
The `OpenHelpDocument` command opens a Help database, or a specified database, to a specified document. The following example opens Designer Help to the "OpenDocument" topic in the "Search" view:

```command
@Command([OpenHelpDocument]; [DesignerHelp]; "Search"; "OpenDocument")
```

This command is equivalent to a URL command formatted as follows:

```
```

OpenNavigator
The `OpenNavigator` command opens a navigator.

```command
@Command([OpenNavigator]; "Main Navigator")
```

Do not use the `solo` parameter in a Web application.

This command is equivalent to a URL command formatted as follows:

```
http://host/database/navigator?OpenNavigator
```

OpenPage
The `OpenPage` command opens a page.

```command
@Command([OpenPage]; "Page One")
```

This command is equivalent to a URL command formatted as follows:

```
http://host/database/page?OpenPage
```

OpenView
The `OpenView` command opens a view in the current database. For example, the following code opens "Main View":

```command
@Command([OpenView]; "Main View")
```

If `key` is specified, the view must be sorted and the OpenView command must specify `key` exactly except for case. On the Web, the view opens with the row containing `key` at the top. The following code opens "Main View" at the first row that contains the value "one":

```command
@Command([OpenView]; "Main View"; "one");
@Command([OpenDocument])
```

In the Notes client, you can specify a partial key, for example, "o" or "on" instead of "one." In a Web application, the key must be exact.

In a Web application, the argument "$first" for `key` means the first row.

These commands are equivalent to URL commands formatted as follows:

```
http://host/database/view?OpenView
http://host/database/view/OpenView?StartKey=one
```

See `FileSave` for using OpenView with FileSave to submit a document.

RefreshFrame
On the Web, the `RefreshFrame` command refreshes the current frame ignoring any parameter.

```command
@Command([RefreshFrame])
```

RemoveFromFolder
The `RemoveFromFolder` command removes selected documents from a folder.

```command
@Command([RemoveFromFolder])
```
This @command works on the Web only if "Use applet in the browser" is in effect for the implementing view or folder.

**ToolsRunMacro and RunAgent**

The **ToolsRunMacro** and **RunAgent** @commands run an agent in the current database. For example, the following formula runs the agent "Status = open":

@Command([ToolsRunMacro]; "Status = open")

or

@Command([RunAgent]; "Status = open")

Agents run on the Domino server containing the database not on the browser computer. See "Web Agents".

**ViewChange and SwitchView**

The **ViewChange** and **SwitchView** @commands open a view in the current database. For example, the following formula opens "Main View":

@Command([ViewChange]; "Main View")

or

@Command([SwitchView]; "Main View")

In a Web application, you cannot omit the *view* parameter.

These @commands are equivalent to a URL command formatted as follows:

http://host/database/view/OpenView

**ViewCollapse**

The **ViewCollapse** @command collapses everything in the current category or under the current main document in a view.

@Command([ViewCollapseAll])

This @command works for a view on the Web only if "Using Java Applet" is in effect.

**ViewCollapseAll**

The **ViewCollapseAll** @command collapses a view so that only the topmost level of category names appears.

@Command([ViewCollapseAll])

This @command is equivalent to a URL command formatted as follows:

http://host/database/viewy/OpenView&Start=1&Count=30&CollapseView

**ViewExpand**

The **ViewExpand** @command expands everything in the current category or under the current main document in a view.

@Command([ViewExpandAll])

This @command works for a view on the Web only if "Using Java Applet" is in effect.

**ViewExpandAll**

The **ViewExpandAll** @command expands a view so that all levels appear.

@Command([ViewExpandAll])

This @command is equivalent to a URL command formatted as follows:

http://host/database/By+Category/OpenView&Start=1&Count=30&ExpandView
**ViewRefreshFields**

The `ViewRefreshFields @command` recalculates all computed field values in the current, open document:

```
@Command([ViewRefreshFields])
```

This @command works on the Web only if "Use applet in the browser" is in effect for the implementing view or folder.

**ViewShowSearchBar**

In a Web application, the `ViewShowSearchBar @command` opens the search view.

```
@Command([ViewSearchBar])
```

In the Notes client, this @command toggles the search bar in a view.

This @command is equivalent to a URL command formatted as follows:

```
http://host/database/universalID/$searchForm?SearchView
```

---

**Web agents**

Agents cannot run in a browser. They can be activated from a browser but run on the Domino server containing the agent.

**Setting up a Web agent**

Using the Agent Properties box, do the following:

- Check the "Shared" option when you create the agent.
- Set the agent trigger to "On event" and either "Agent list selection" or "Action menu selection."
- Set the agent target to either:
  - "None“ for agents that work on the current document such as those launched from WebQueryOpen or WebQueryClose, or a form action or hotspot that works on fields in the current document.
  - "All documents in database" for agents that work on existing documents such as those launched from the OpenAgent URL or a view action. The actual documents processed depend on the agent code, for example, the SELECT statement in a formula or the UnprocessedDocuments property in LotusScript.
- Check "Run as web user" on the Security tab to run the agent using the browser login name. Otherwise the agent runs with the rights of the agent signer.

**Activating a Web agent**

You can activate agents from a browser in two ways:

- OpenAgent URL command. Entering the name of an agent wherever URLs are allowed launches the agent.
- `RunAgent @command` or `ToolsRunMacro @command` You can use these @commands, which are equivalent, in an action, a hotspot action, a hotspot button, the WebQueryOpen event, or the WebQuerySave event.

The name of a hidden agent (for example, if the trigger is "Agent list selection") must include the parentheses when RunAgent or ToolsRunMacro launches the agent. The parentheses are optional when the OpenAgent URL command launches the agent. The OpenAgent URL requires conversion of special characters for Web use; for example, a space must be specified as + (plus sign) or %20.

Two form events work for Web processing:

- WebQueryOpen occurs before Lotus Domino converts the document being opened to HTML and sends it to the browser. You can change initial field values and do other pre-processing.
• WebQuerySave occurs before a Web document is saved. You can change final field values and do other post-processing.

To prevent the document from being saved, the document must contain a text field named SaveOptions with "0" as its value.

Note: The SaveOptions field must be an existing field, the value of which the Web agent changes to "0". If the Web agent creates this field, the document will be saved regardless of the value.

LotusScript and Java in Web agents

A Web agent can be written in LotusScript or Java, as well as formula or simple actions.

An agent is the only context in which a Web application can use LotusScript. Anything that runs in the browser, such as an action, hotspot action, hotspot button, or event, cannot contain LotusScript. LotusScript agents activated from the Web cannot use any of the classes NotesTimer, NotesUIDatabase, NotesUIDocument, NotesUIView, or NotesUIWorkspace.

You can write HTML to the browser by putting it in print statements. Domino accumulates print statements and creates a page with their contents after the agent runs.

To display a new page after an agent runs, put one print statement at the end of the code containing the page’s URL in brackets. You can start the URL with a slash and the name of the database to designate an element in the current database.

In Java, you must get a PrintWriter object with AgentBase.getOutputStream and use this object to write to the browser.

The available context depends on how the agent is started:
• If the RunAgent or ToolsRunCommand @command starts the agent, DocumentContext or getDocumentContext gets an object representing the current document.
• If the OpenAgent URL command starts an agent, there is no current database, but DocumentContext and getDocumentContext return a special NotesDocument or Document object containing the values of the CGI variables supported by Lotus Domino. In the case of RunAgent or ToolsRunCommand, CGI values are available but not automatically. The form backing up the current document must contain a field named after each desired CGI variable.

The OpenAgent URL command passes arguments at the end of the URL string delineated by ampersands. For example:
http://host/database/view/OpenView&Start=1&Count=20

In LotusScript and Java agents, you can get the arguments by parsing the Query_String item in the document returned by DocumentContext or getDocumentContext. Query_String contains the entire URL command that started the agent.

Examples: Web agents

1. This agent is named "Change Status to Closed" and sets the Status field to the value "Closed" in all documents that meet the selection criteria. The target is "All documents in database."

   FIELD Status := "Closed";
   SELECT @ALL
   To run the agent, you can enter the OpenAgent URL command wherever URLs are permitted, for example:
   http://localhost/Web+test.nsf/Change+Status+to+Closed?OpenAgent
   Or activate an action or hotspot that contains the following code:
2. A "Computed for display" field named HeadText initially has the following value:

"This document is being opened from Notes at " + @Text(@Now)

The WebQueryOpen event on the form calls the agent "ChangeHeadText." The target is "None." The code is:

@Command([ToolsRunMacro]; "ChangeHeadText")

ChangeHeadText contains the following formula. When a document based on this form is opened from a browser, WebQueryOpen causes HeadText to be changed; when a document is opened from Notes, HeadText remains as is.

FIELD HeadText :=
"This document is being opened from a browser at " + @Text(@Now); @All

3. This is a LotusScript version of "Change Status to Closed." It runs on "All documents in database" and uses UnprocessedDocuments to get the documents to be processed. The Print statement replaces the "Agent done" page with its text.

Sub Initialize
Dim s As New NotesSession
Dim db As NotesDatabase
Dim dc As NotesDocumentCollection
Dim doc As NotesDocument
Set db = s.CurrentDatabase
Set dc = db.UnprocessedDocuments
Set doc = dc.GetFirstDocument
Do While Not(doc Is Nothing)
    doc.Status = "Closed"
    Call doc.Save(False, True)
    Set doc = dc.GetNextDocument(doc)
Loop
Print "<B>All Status fields set to 'Closed'</B>"
End Sub

4. This is a LotusScript version of "Change Status to Closed" that opens the "Main View" page in a browser when the agent terminates.

Sub Initialize
Dim s As New NotesSession
Dim db As NotesDatabase
Dim dc As NotesDocumentCollection
Dim doc As NotesDocument
Set db = s.CurrentDatabase
Set dc = db.UnprocessedDocuments
Set doc = dc.GetFirstDocument
Do While Not(doc Is Nothing)
    doc.Status = "Closed"
    Call doc.Save(False, True)
    Set doc = dc.GetNextDocument(doc)
Loop
dbname$ = Evaluate("@WebDbName")
Print "[/" + dbname$ + "+Main+View?OpenView]"
End Sub

5. This is a Java version of "Change Status to Closed."

```java
import lotus.domino.*;
import java.io.PrintWriter;
import java.util.Vector;

public class JavaAgent extends AgentBase {

    public void NotesMain() {
        try {
            Session session = getSession();
            AgentContext agentContext = session.getAgentContext();
```
// (Your code goes here)
DocumentCollection dc = 
    agentContext.getUnprocessedDocuments();
Document doc = dc.getFirstDocument();
while (doc != null) {
    doc.replaceItemValue("Status", "Closed");
    doc.save(false, true);
    doc = dc.getNextDocument(doc);
} 
PrintWriter pw = getAgentOutput();
Vector v = session.evaluate("@WebDbName");
pw.println("[/" + v.firstElement() + "/Main+View?OpenView]"); 
} catch(Exception e) {
    e.printStackTrace();
} 
}

6. This agent parses Query_String to extract one argument, which must be "Open" or "Closed." It must be run with an OpenAgent URL command, for example,
http://localhost/Web+test.nsf/Change+Status?OpenAgent&Closed
Here is the code:
Sub Initialize
    Dim s As New NotesSession
    Dim db As NotesDatabase
    Dim dc As NotesDocumentCollection
    Dim doc As NotesDocument
    Dim arg As String, p1 As Long
    arg = s.DocumentContext.Query_String(0)
    p1 = Instr(arg, ")")
    If p1 = 0 Then
        Print "Need argument 'Open' or 'Closed'"
        Exit Sub
    Else
        arg = Lcase(Mid$(arg, p1 + 1))
        If arg <> "open" And arg <> "closed" Then
            Print "Argument must be 'Open' or 'Closed'"
            Exit Sub
        End If
    End If
    arg = Ucase(Left$(arg, 1)) + Right$(arg, Len(arg) - 1)
    Set db = s.CurrentDatabase
    Set dc = db.UnprocessedDocuments
    Set doc = dc.GetFirstDocument
    Do While Not(doc Is Nothing)
        doc.Status = arg
        Call doc.Save(False, True)
        Set doc = dc.GetNextDocument(doc)
    Loop
    Print "<b>Status changed to " + arg + " in all documents</b>"
End Sub

**JavaScript**

JavaScript executes in browsers and the Notes client providing a simple, effective way to program the user interface. You can place JavaScript code in [JavaScript events](#) and for Web-only applications in [HTML](#). Domino Designer objects map to the standard [JavaScript object model](#). Through JavaScript you can access the [Java/CORBA classes](#).
JavaScript events

The following Domino objects have JavaScript event handlers: form, subform, page, field, action, button, and action hotspot. You can attach code to these event handlers in the Programmer’s pane. The code executes when the event occurs in a browser that supports the event, and, limitedly, in the Notes client.

Table of JavaScript events

The following table lists the JavaScript event handlers in Lotus Domino Designer. "Browser only" means that the event handler only works in browsers that support it. "Browser and Notes" means that the event handler works in browsers that support it and the Notes client.

<table>
<thead>
<tr>
<th>Event</th>
<th>Form, subform, page</th>
<th>Field</th>
<th>Action, button, hotspot</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSHeader</td>
<td>Browser and Notes</td>
<td>Uses the JSHeader for the form, subform, or page</td>
<td></td>
</tr>
<tr>
<td>onBlur</td>
<td>NA</td>
<td>Browser and Notes</td>
<td>Browser only</td>
</tr>
<tr>
<td>onChange</td>
<td>NA</td>
<td>Browser and Notes</td>
<td>NA</td>
</tr>
<tr>
<td>onClick</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser and Notes</td>
</tr>
<tr>
<td>onDbClick</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onFocus</td>
<td>NA</td>
<td>Browser and Notes</td>
<td>Browser only</td>
</tr>
<tr>
<td>onHelp</td>
<td>Browser and Notes</td>
<td>NA</td>
<td>Browser only</td>
</tr>
<tr>
<td>onKeyDown</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onKeyPress</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onKeyUp</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onLoad</td>
<td>Browser and Notes</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>onMouseDown</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onMouseMove</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onMouseOut</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onMouseOver</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onMouseUp</td>
<td>Browser only</td>
<td>Browser only</td>
<td>Browser only</td>
</tr>
<tr>
<td>onReset</td>
<td>Browser only, form only</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>onSelect</td>
<td>NA</td>
<td>Browser only</td>
<td>NA</td>
</tr>
<tr>
<td>onSubmit</td>
<td>Browser and Notes</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>onUnload</td>
<td>Browser and Notes</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Events that work in browsers and the Notes client

Restrict code that must work in both a browser and the Notes client to the following events:
- For document preprocessing and post-processing, use onLoad, onUnload, and onSubmit.
- For processing on entering and exiting fields, use onFocus, onBlur, and onChange.
- For an action, button, or action hotspot, use onClick.

JSHeader event

JSHeader is a special event handler that loads code such as functions and global variable declarations that all the events in the object can access. This code goes under the <HEAD> tag in Domino-generated HTML.

onSubmit event

The onSubmit event occurs in a browser and the Notes client when the FileSave @command executes. In a browser, you can return false from the onSubmit event to abort the save operation.
Enabling JavaScript in the Notes client

To run JavaScript in a Notes client, the user must select "Enable JavaScript" under "Additional Options" after choosing File - Preferences - User Preferences. To expand or limit security, the user must select "Using JavaScript" under "What Others Do" after choosing File - Security - User Security.

Examples: JavaScript events

1. This form onLoad event sets the value of the Status field to "open" when a document loads. It works in a browser and the Notes client.

   
   document.forms[0].Status.value = "open"

2. This version of the onLoad event solicits a value for the Status field.

   
   reply = prompt("Status = 'Open' or 'Closed'?", "Open")
   while (reply != "Open" && reply != "Closed") {
       reply = prompt("Enter 'Open' or 'Closed'", "Open")
   }
   document.forms[0].Status.value = reply

3. This Status field onBlur event prevents the user from exiting the field without entering "Open" or "Closed." It works in a browser and the Notes client.

   
   r = document.forms[0].Status.value
   while (r != "Open" && r != "Closed") {
       r = prompt("Status must be 'Open' or 'Closed'", "Open")
   }
   document.forms[0].Status.value = r

4. The code shown in the last example could be placed instead in the onSubmit event. An action or hotspot containing the following @commands activates the code.

   
   @Command([FileSave]);
   @Command([FileCloseWindow])

5. This function is placed in the JSHeader event of a form.

   
   function testEvent(eventName) {
       alert("This is the " + eventName + " event")
   }

   The function executes whenever another event on the form (including a field, action, or hotspot) calls it, for example:

   TestEvent("onLoad")

JavaScript in HTML

You can use JavaScript anywhere HTML can be placed by using the standard SCRIPT LANGUAGE tag as shown below.

<SCRIPT LANGUAGE="JavaScript">
    code goes here</SCRIPT>

Mark the code as HTML by selecting it then choosing Text - Pass-thru HTML.

Lotus Domino passes the HTML to browsers as is.

Check "Render pass through HTML in Notes" in the Form or Page Properties box to have the Notes client process the HTML. If you do not check "Render pass through HTML in Notes," the HTML appears as plain text. You can hide it using the following formula for "Hide paragraph if formula is true":

@ClientType = "Notes"

If you refer to a field, the JavaScript code must follow the field on the form.

You can embed computed fields and computed text in HTML text on a form, subform, or page.

You can place JavaScript in a URL using the "javascript:" protocol, for example:

<A HREF="javascript:code goes here">prompt</A>
You can run JavaScript from an agent by sending it to the browser as HTML using print statements. For example, in LotusScript:

```
Print "<SCRIPT LANGUAGE=JavaScript>"
Print "code goes here"
Print "</SCRIPT>"
```

To run an agent from JavaScript, set the href property of the location object to the URL for opening the agent. The URL can be relative to the current host. For example:

```
location.href = "/dbname.nsf/agentname?OpenAgent&arg1=val"
```

**Examples: JavaScript in HTML**

1. This is text on a form marked as passthru HTML. It prompts for a name and puts it in the LoginName field. The text must follow the LoginName field on the form.
   ```html
   <SCRIPT LANGUAGE="JavaScript">
   n = prompt("Domino login name?", "Anonymous")
   document.forms[0].LoginName.value = n
   </SCRIPT>
   ```

2. This passthru text uses an anchor tag to put JavaScript in a URL using the "javascript:" protocol.
   ```html
   <A HREF="javascript:alert('You are Anonymous unless you supply a login name')">Read me</A>
   ```

3. This passthru text contains the user name as a computed value. The computed value is inserted by choosing Create - Computed Text and specifying "@UserName" for the formula.
   ```html
   <SCRIPT LANGUAGE=JavaScript>
   alert("Your user name is <Computed Values>")
   </SCRIPT>
   ```

4. This LotusScript agent terminates by sending JavaScript to the browser. The JavaScript displays a message and then loads the "Main View."

```
Sub Initialize
   Dim s As New NotesSession
   Dim db As NotesDatabase
   Dim dc As NotesDocumentCollection
   Dim doc As NotesDocument
   Dim arg As String, p1 As Long
   arg = s.DocumentContext.Query_String(0)
   p1 = Instr(arg, ":")
   If p1 = 0 Then
      Print "Need argument 'Open' or 'Closed'"
      Exit Sub
   Else
      arg = Lcase(Mid$(arg, p1 + 1))
      If arg <> "open" And arg <> "closed" Then
         Print "Argument must be 'Open' or 'Closed'"
         Exit Sub
      End If
      arg = Ucase(Left$(arg, 1)) + Right$(arg, Len(arg) - 1)
   End If
   Set db = s.CurrentDatabase
   Set dc = db.UnprocessedDocuments
   Set doc = dc.GetFirstDocument
   Do While Not(doc Is Nothing)
      doc.Status = arg
      Call doc.Save(False, True)
      Set doc = dc.GetNextDocument(doc)
   Loop
   Print "<SCRIPT LANGUAGE=JavaScript>"
   Print "alert("Status changed to " & arg & ":" & arg & ":" in all documents")"
   Print "location.href = "/Web+test.nsf/Main+View?OpenView"
   Print "</SCRIPT>"
End Sub
```
Sub Initialize
    Dim s As New NotesSession
    Dim db As NotesDatabase
    Dim dc As NotesDocumentCollection
    Dim doc As NotesDocument
    Dim context As NotesDocument
    Dim arg As String
    Set context = s.DocumentContext
    arg = s.DocumentContext.Query_String(0)
    pl = Instr(arg, ":")
    If pl = 0 Then
        Print "Need argument 'open' or 'closed'"
    Else
        arg = Lcase(Mid$(arg, pl + 1))
        If arg <> "open" And arg <> "closed" Then
            Print "Argument must be 'open' or 'closed'"
        End If
    End If
    Set db = s.CurrentDatabase
    Set dc = db.UnprocessedDocuments
    Set doc = dc.GetFirstDocument
    Do While Not(doc Is Nothing)
        doc.Status = arg
        Call doc.Save(False, True)
        Set doc = dc.GetNextDocument(doc)
    Loop
    Print "<SCRIPT LANGUAGE=JavaScript>"
    Print "alert(""Status changed to ":" & arg & 
        " in all documents")"
    Print "<a href="/Web+test.nsf/Main+View?OpenView""
    Print "</SCRIPT>"
End Sub

JavaScript object model

Browser implementation of the object model depends upon the browser. The Notes client implements the object model with some exceptions. For information on the Notes implementation, see the Notes Release 5 Client Document Object Model in http://www.lotus.com/ldd/doc.

The JavaScript objects map to the Domino design elements as discussed in the following sections.

Navigator
The navigator object applies to the currently opened form, page, view, or frameset. The appName, appCodeName, appVersion, platform, and userAgent properties apply to the invoking browser or client and return the same information no matter what the base Domino Designer object is.

Window
The window object applies to the currently opened form, page, view, or, for a frameset, the frame that is in focus. Suppose you design two pages named "Page One" and "Page Two," and a frameset with two frames named “High” and “Low,” where “High” opens "Page One" and "Low" opens "Page Two." If you open "Page One" on its own, window.status means the status property of "Page One." If you open the frameset, window.status means "Page One" when the focus is on "High" and "Page Two" when the focus is on "Low."

Frame
Domino frames can be accessed by name and through the frames array. The name is as specified in the Frame Properties box. Use window.top to access the window representing the frameset. For example, if a
frameset has two frames named "High" and "Low," you can refer to the first frame as window.top.High or window.top.frames[0], and the second as window.top.Low or window.top.frames[1].

The name property initially contains the Domino frame name and is empty if the window does not represent a Domino frame.

Nested frames represent nested Domino framesets. For example, if the "Low" frame contains another frameset with two frames named "Left" and "Right," you can refer to "Left" as window.top.Low.Left, window.top.frames[1].frames[0], or some combination of names and frame elements.

Use the "parent" property to access the parent window of a frame. If the focus is in the "Right" frame in the above example, window.parent refers to window.top.Low and window.parent.Left refers to the adjacent window.top.Low.Left.

**History and location**
The history and location objects apply to whatever the parent window applies.

**Document**
The document object represents the contents of the currently open Domino form, page, or view. The document object contains the following:

- The applets array includes Domino action bar, view, and rich text applets as well as applets you import. Applets are represented by the <APPLET> tag in HTML. You can refer to an applet by name, for example, document.AppletName, if you specify a name under the HTML tab of Applets properties.
  
  For more information, see the help topic Adding an applet, or "Designing Pages" in *Application Development with Domino Designer*.

- The links array includes Domino actions, link hotspots, and action hotspots, which are represented in HTML by the <A> tag. You can refer to action hotspots by name, for example, document.HotspotName, if you specify a name under the HTML tab of the HotSpot Properties box. Domino link hotspots do not contain event handlers. However, you can specify a handler in "Other" under the <HTML> tag of the HotSpot Properties box.

- The images array includes Domino attachments, image resources, and pictures, which are represented in HTML by the <IMG> tag. You can refer to pictures by name, for example, document.PictureName, if you specify a name under the HTML tab of the Pictures Properties box.

  You can refer to a Domino image resource with a URL that specifies the database name or replica ID followed by the name of the image resource, for example, /Web+Test.nsf/newdam.gif.

- The forms array typically has one element, document.forms[0] named _DominoForm, which is represented in HTML by the <FORM> tag. It has an array of named elements that contains any objects of type Button, Text, Textarea, Password, Select, Radio, Checkbox, Hidden, and FileUpload.

- The Button object represents Domino buttons, which are represented in HTML by the <INPUT TYPE=button> tag. You can refer to buttons by name, for example, document.forms[0].ButtonName, if you specify a name under the HTML tab of the Button Properties box.

- The Text object represents Domino fields of type Text, Date/Time, Number, Names, Authors, and Readers, which are represented in HTML by the <INPUT> tag. You can refer to Text objects by their Domino field names, for example, document.forms[0].FieldName.

- The TextArea object represents Domino fields of type RichText. It is represented in HTML by the <TEXTAREA> tag. You can refer to TextArea objects by their Domino field names, for example, document.forms[0].FieldName.

- The Password object represents Domino fields of type Password, which are represented in HTML by the <INPUT TYPE=password> tag. You can refer to Password objects by their Domino field names, for example, document.forms[0].FieldName.

- The Select object represents Domino fields of type Dialog list, Listbox, and Combobox, which are represented in HTML by the <SELECT> tag. You can refer to Select objects by their Domino field names, for example, document.forms[0].FieldName.
• The Radio object represents Domino fields of type Radio button, which are represented in HTML by
the <INPUT TYPE=radio> tag. You can refer to Radio objects by their Domino field names, for
example, document.forms[0].FieldName.

• The Checkbox object represents Domino fields of type Checkbox, which are represented in HTML by
the <INPUT TYPE=checkbox> tag. You can refer to Checkbox objects by their Domino field names, for
example, document.forms[0].FieldName.

• The Hidden object represents Domino fields when the "Hide paragraph when document is" attribute is
selected. A hidden field is represented in HTML by the <INPUT TYPE=hidden> tag, no matter what
kind of field it is, if "Generate HTML for all fields" is selected in the Form Properties box. Otherwise,
hidden fields are not accessible to JavaScript in a browser; they are never accessible to JavaScript in the
Notes client. You can refer to Hidden objects by their Domino field names, for example,
document.forms[0].FieldName.

• The FileUpload object has no direct representation as a Domino field type. You can generate this object
in a browser by creating a field (for example, of type Text) and specifying "INPUT TYPE=file" for
"Other" under the HTML tab. A FileUpload object cannot be accessed in the Notes client.

The names of Domino fields are case sensitive. In Edit mode, Domino fields are accessible to JavaScript in
a browser and the Notes client with restrictions as noted for hidden fields. In Read mode, Domino fields
are not accessible to JavaScript in a browser unless "Generate HTML for all fields" is selected in the Form
Properties box. In Read mode, Domino fields are never accessible to JavaScript in the Notes client.

Examples: JavaScript object model

1. This JSHeader event handler declares a function that writes the product of the Width and Length
fields to the SquareFeet field.

   function getSquareFeet() {
     with (window.document.forms[0]) {
       SquareFeet.value = Width.value * Length.value
     }
   }

   This onFocus event handler for both the Length and Width fields calls the function:
   getSquareFeet();

2. This Domino form onLoad event handler sets the window status display and then displays browser
and platform information.

   window.status = "Domino window opened"
   n = navigator
   alert (n.appName + " " + n.appVersion + " on " + n.platform)

3. This Domino hotspot onClick event handler loads a new URL by writing to the href property of the
location object.

   location.href = "http://localhost/Web+test.nsf/Main+View"

4. These onLoad event handlers are for two pages in a frameset. When the focus is on the first frame,
the status bar displays "Page one." When the focus is on the second frame, the status bar displays
"Page two."

   window.status = "Page one"
   window.status = "Page two"

5. This onClick event handler for a page action displays the names of all the frames in a non-nested
frameset.

   for (n = 0; n < top.length; n++) {
     alert(top.frames[n].name)
   }

6. This onClick event handler for a page action displays the names of all the frames in a frameset that
may contain one level of nesting.

   for (n = 0; n < top.length; n++) {
     w = top.frames[n]
     if (w.length == 0) {
       ...
alert (w.name)
}
else {
    for (nn = 0; nn < w.length; nn++) {
        ww = w.frames[nn]
        alert (w.name + ": " + ww.name)
    }
}
}

7. This `onClick` event handler for a page action displays the names of the applets on the page.
for (n = 0; n < window.document.applets.length; n++) {
    alert (window.document.applets[n].name)
}

8. This code, placed in "Other" under the `<HTML>` tag of the HotSpot Properties box for a link hotspot, queries the user for a confirmation before going to the link.
`onClick="return confirm('Go?')"

9. This code is for the `onClick` event handler of a picture named ThePicture. When the user clicks the picture, it changes to the image resource in "Web Test.nsf" named newdam.gif.
`document.ThePicture.src = "/Web+Test.nsf/newdam.gif"

10. This `onClick` event handler for a button displays information about all the elements of the elements array in the current JavaScript form.
    
    ```
    e = window.document.forms[0].elements
    for (n = 0; n < e.length; n++) {
        if (e[n].name == "") ee = "NO NAME"
        else ee = e[n].name
        t = e[n].type
        if (e[n].value == "") v = "NO VALUE"
        else v = e[n].value
        alert (ee + 
                t + 
                v)
    }
    ```

11. This `onBlur` event handler for FieldTwo forces FieldTwo to be blank if FieldOne contains the value "NONE."
    
    ```
    one = window.document.forms[0].FieldOne.value
    two = window.document.forms[0].FieldTwo.value
    if (one == "NONE" && two != "") {
        window.document.forms[0].FieldTwo.value = ""
        alert ("FieldTwo must be blank if FieldOne is NONE")
    }
    ```

12. This `onClick` event handler for a button displays the selected values of a "Dialog list" field.
    ```
    with (window.document.forms[0].MyList) {
        for (n=0; n<length; n++) {
            if (options[n].selected) {
                alert (options[n].text)
            }
        }
    }
    ```

13. This `onClick` event handler for a button displays all the hidden fields in the document. This only works in browsers and only if "Generate HTML for all fields" is selected in the Domino Forms Properties box.
    ```
    with (window.document.forms[0]) {
        for (n=0; n<elements.length; n++) {
            if (elements[n].type == "hidden") {
                alert (elements[n].name + "\n" + elements[n].value)
            }
        }
    }
    ```
Domino objects

You can access through JavaScript the Domino Objects available through Java and CORBA. See Java/CORBA classes in Domino Designer Programming Guide, Volume 3.

Take the following steps:

1. On the form or page that will include the JavaScript code, create an applet that extends AppletBase. See "Running a Java program" and the associated examples in Domino Designer Programming Guide, Volume 3 for coding such an applet. See "Including Java Applets in Applications" in Application Development with Domino Designer for instructions on including applets on forms and pages. The applet can do anything. It is there just as a gateway to the Domino Objects.

2. In the Java Applets Properties box, select "Applet uses Notes CORBA classes."

3. For Notes client deployment, select "Enable Java access from JavaScript" in the User Preferences box. Also click the "Security Options" button, then select the "Java applet security" radio button, and make sure "Access to Notes Java classes" is selected for the appropriate users.

4. For access to the classes through CORBA, make sure the Server record in the Domino Directory correctly records who can "Run restricted Java/JavaScript" and "Run unrestricted Java/JavaScript" in the "IIOP Restrictions" section under the Security tab.

5. In the JavaScript code, refer to the applet to access the classes and methods for the Domino objects. You first want to call AppletBase.openSession() or AppletBase.openSession("username", "password") to obtain a Session object. For example, if the applet extending AppletBase is the first applet on the form or page, and a user name and password are not needed, the following code works:

   ```javascript
   session = document.applets[0].openSession()
   ```

You can then use the returned Session object to get at the other Domino objects. If you instantiate other Domino objects in the Java applet, you can use them directly in the JavaScript code.

In the Notes client, you can call AppletBase.getContext(Session) to get a NotesAppletContext object. This object contains the following methods: getServer() returns the name of the server containing the applet or a blank string for a workstation; getDatabase() returns the file path and name of the database containing the applet; getDocument() returns the universal ID of the document containing the applet or null if the document has not been saved.

Examples: Domino objects

These examples are on a page that contains one Java applet whose source code is:

```java
import lotus.domino.*;
public class dummy extends Applet{}
```

1. This onClick event handler for a button uses the Domino objects to display the current platform. The code gets to the Domino objects through the first applet in the current document; this applet must use the Domino objects. The objects employed are AppletBase.openSession and Session.getPlatform.

   ```javascript
   var session = window.document.applets[0].openSession()
   alert (session.getPlatform())
   ```

2. This onClick event handler for a button uses the Domino Objects to get the context for the current document. This code works only on a Notes client.

   ```javascript
   var s = window.document.applets[0].openSession()
   var x = window.document.applets[0].getContext(s)
   alert (x.getServer())
   alert (x.getDatabase())
   alert (x.getDocument())
   ```

3. This onClick event handler for a button uses the Domino Objects to display the title of the current database and the value of each Subject item. The field thisDb is a computed field containing "@Subset(@DbName; -1)" as the value. For browsers, "Generate HTML for all fields" must be selected in the Form Properties box.
Web services

Note: Web services are new with Release 7.

Domino supports Web services as defined in the W3C documents Simple Object Access Protocol (SOAP) 1.1 (http://www.w3.org/TR/2000/NOTE-SOAP-20000508) and Web Services Description Language (WSDL) 1.1 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315). See also Web Services Architecture (http://www.w3.org/TR/wss-arch) and Web Services Activity (http://www.w3.org/2002/ws/).

Domino limits Web services as follows:
- Only provider entities are supported.
- Binding must be SOAP (Simple Object Access Protocol) using HTTP POST protocols.

Development in Domino Designer is through a "Web service" design element coded in Java or LotusScript. The code defines and implements the service. The Web service also maintains the definition as a WSDL (Web Services Description Language) document.
- The compiled Web service, like an agent, is a stand-alone program in a Domino database. To be used, the Web service must be on -- or be replicated to -- a server that is enabled for Web access. It can then be accessed with the following Domino URLs:
  - OpenWebService -- using HTTP POST protocols, invokes the Web service. If HTTP GET protocols are used, OpenWebService returns some information about the service.
  - WSDL -- queries the Web service for its WSDL document.
- A Web service can be tested through an HTTP session in Notes or Designer preview.
- A Web service has the same security capabilities as an agent.
- You can import an existing WSDL document to generate a skeleton Web service. The skeleton code corresponds to the Web service definition. You then add implementation code.
- Alternatively you can start a Web service from scratch, writing your own Java or LotusScript code to create the Web service definition.
- In Domino Designer, you can show or export the WSDL document that defines the Web service. For Java, you can export the source, create a new class, edit the project files, and compile the project files (as for a Java agent).

The following sections describe:
- Web services terminology
- Web services in Domino Designer
- Web services on a Domino server
- Java and LotusScript mappings

Web services terminology

A Web service is a self-contained, self-describing, modular application, based on XML, that can be published to and invoked from the Web. Domino Web services are based on the following W3C standards:
WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented messages.

SOAP defines XML-based information used for exchanging structured and typed information between peers in a decentralized, distributed environment. A SOAP message is fundamentally a one-way transmission from a SOAP sender to a SOAP receiver, but SOAP messages are expected to be combined by applications to implement more complex interaction patterns.

A WSDL document defines a Web service and incorporates the following elements:

- **Service** -- The Web service as a whole. In a WSDL document, the service is described by a `<definitions>` element at the root. Children are `<types>`, `<message>`, `<portType>`, `<binding>`, and `<service>`. The `<service>` element has a "name" attribute that names the service.

- **Types** -- The data types used by the service. In a WSDL document, the data types are described in a `<types>` element containing an `<xsd:schema>` element. See http://www.w3.org/TR/xmlschema-2 for descriptions of the data types.

- **Message** -- An abstract definition of data being communicated to or from the service. There may be any number of messages. In a WSDL document, each message is described by a `<message>` element. Each `<message>` element contains one or more `<part>` elements to define the data by name and type.

- **Port type** -- An abstract set of operations supported by the service. In a WSDL document, a port type is described by a `<portType>` element. Each `<portType>` element contains one or more `<operation>` elements.

- **Operation** -- An abstract description of an action supported by the service. In a WSDL document, an operation is described by an `<operation>` element. Each `<operation>` element contains `<input>` and `<output>` elements that specify associated messages.

- **Binding** -- A concrete protocol and data format specification for a port type. In a WSDL document, a binding is described by a `<binding>` element. A `<binding>` element has a "type" attribute that specifies the name of the port type. For SOAP encoding, a `<binding>` element contains a `<soap:binding>` element with "style" ("rpc" or "document") and "transport" ("http://schemas.xmlsoap.org/soap/http") attributes. A `<binding>` element contains `<operation>` elements that describe the data format for each operation.

- **Port** -- A single endpoint defined as a combination of a binding and a network address. In a WSDL document, a port is defined by a `<port>` element under the `<service>` element. The `<port>` element has a "binding" attribute to name the binding and an "address location" attribute to name the network endpoint to be associated with the binding.

### Web services in Domino Designer

The Web service design element exists below "Agents" under "Shared Code" in the Domino Designer UI. The design element has a properties box with Basics, Security, and Advanced tabs, and an editor pane. The functional part of a Web service is a compiled program. A Web service also maintains a WSDL document that reflects the program interface. An imported WSDL document is maintained as is. If coding changes affect the interface, the WSDL document is regenerated when a Web service is saved. Note that when the WSDL document is regenerated for a LotusScript Web service, all LotusScript identifiers which become WSDL names appear in upper-case. This is because LotusScript is a case-insensitive language, and identifier names are maintained internally in uppercase only.

The Web service design element can be locked and unlocked.

### Basics tab

The Basics tab of the properties box contains:

- **Name, Alias, Comment** -- Identify the Web service just as you would any other design element such as an agent.

- **Options** -- You can check "Warn if the WSDL interface is modified."

- **PortType class** -- Specify the class that contains the operations (the implementation code) for the port type. (You have to create the class first.)
Security tab

Security of Web services is the same as for agents:

- Run as web user -- The Java or LotusScript code runs with the effective user name of the user invoking the Web service.
- Run on behalf of -- The Java or LotusScript code runs on the authority of a specified user. However, the signer of the Web service must have unrestricted rights on the server or have the right to run on behalf of anyone.
- Compile Java code with debugging information -- For Java agents, generates code suitable for debugging.
- Allow remote debugging -- For LotusScript agents, generates code suitable for remote debugging.
- Profile this web service -- For Java or LotusScript agents, generates code suitable for profiling.
- Set runtime security level (one of the following):
  - 1. Do not allow restricted operations
  - 2. Allow restricted operations
  - 3. Allow restricted operations with full administration rights
- Default access for this web service:
  - All readers and above -- You can select or deselect this option.
  - Enumerated -- If the above is deselected, you can enumerate who you want to have default access.
  - Allow Public Access users to use this web service -- Allows default access to those who have public access to documents in the database.

Advanced tab

The top of the Advanced tab of the properties box specifies the programming model and SOAP message format. Select one of the following combinations. The descriptions below explain how the options affect the WSDL document:

- RPC programming model, RPC/encoded
  - For <wsdl:message>, <wsdl:part>: type= specifies an XSD type or a type defined in <wsdl:types>
  - For <wsdlsoap:binding>: style= &quot;rpc&quot;
  - For <wsdl:operation>, <wsdl:input> and <wsdl:output>, <wsdlsoap:body>:
    - encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/"
  - For <wsdl:operation>, <wsdl:input> and <wsdl:output>, <wsdlsoap:body> : use= "encoded"
- RPC programming model, RPC/literal
  - For <wsdl:message>, <wsdl:part>: type= specifies an XSD type
  - For <wsdlsoap:binding>: style= &quot;rpc&quot;
  - For <wsdl:operation>, <wsdl:input> and <wsdl:output>, <wsdlsoap:body> : use= "literal"
- RPC programming model, Doc/literal
  - <wsdl:types> defines an element for each input and output
  - For <wsdl:message>, <wsdl:part>: element= specifies an element defined in <wsdl:types>
  - For <wsdlsoap:binding>: style= "document"
  - For <wsdl:operation>, <wsdl:input> and <wsdl:output>, <wsdlsoap:body> : use= "literal"
- RPC programming model, Wrapped
  - <wsdl:types> defines a single complex element for each operation’s inputs and a single complex element for each operation’s outputs, with no attributes; each complex element is named for its corresponding operation
  - For <wsdl:message>, <wsdl:part>: element= specifies an element just defined in <wsdl:types> (each <wsdl:message> has a single <wsdl:part> where name= "parameters")
  - For <wsdl:portType>, <wsdl:operation>: parameterOrder=""
  - For <wsdlsoap:binding>: style= "document"
- For `<wsdl:operation>`, `<wsdl:input>` and `<wsdl:output>`, `<wsdl:soap:body>`: use="literal"

**Message programming model, document/literal**
- For `<wsdl:message>`: each has a name= attribute but no `<wsdl:part>` (The actual SOAP body schema is not public. It is a private contract between consumer and provider.)
- For `<wsdlsoap:binding>`: style="document"
- For `<wsdl:operation>`, `<wsdl:input>`, `<wsdl:output>`, and `<wsdl:soap:body>`: use="literal"
- For `<wsdl:operation>`: these are generated only by service methods that specify the following `interface`: for LotusScript - `Function methodName (doc As NotesDomDocumentNode) As NotesDomDocumentNode;` for Java - `org.w3c.dom.Document methodName (org.w3c.dom.Document doc)`

In general, use "RPC" to deserialize the data into concrete data types. Use "Message" for MOM (message oriented model) service operations that use specific parameter signatures to pass actual XML message syntax from the SOAP body rather than deserializing the data into concrete data types. See "Which style of WSDL should I use?" by Russell Butek at http://www.ibm.com/developerworks/webservices/library/ws-whichwSDL.

The Advanced tab of the properties box also contains:
- Include operation name in SOAP action -- This setting affects the "soapAction" attribute of `<wsdl:operation>` under `<Wsdl:binding>` for each operation:
  - Deselect for soapAction="" 
  - Select for soapAction="nameOfOperation" (where nameOfOperation is the same as the "name" attribute for `<operation>`)
- Port type name -- Specify the name of the port type for accessing the service. This specification corresponds to the "name" attribute for `<wsdl:portType>` in the WSDL document. Domino allows one port type per service.
- Service element name -- Specify the name of the service. This specification corresponds to the "name" attribute of `<wsdl:service>` in the WSDL document.
- Service port name -- Specify the port for accessing the service. This specification corresponds to the "name" attribute of `<wsdl:port>` under `<wsdl:service>` in the WSDL document. Domino allows one port per service.

**Editor pane**
The editor pane of the Web service consists of either Java or LotusScript code.
- Data types -- The Java or LotusScript code defines a class for each data aggregate. In the WSDL document, these classes appear as `<complexType>` elements, and also `<simpleType>` elements with enumeration restrictions (facets). Primitive data types and arrays of primitive data types are mapped directly as specified below under "Java and LotusScript mappings."
- Implementation (or PortType) class -- The Java or LotusScript code defines a class that contains a public method, function, or sub for each operation in the Web service. In the WSDL document, these operations appear as `<wsdl:operation>` elements under `<wsdl:portType>`. The name of the class must match "PortType class" in the Basics tab of the properties box (the term "PortType" comes from the W3C specification for WSDL documents). In the simple case where an operation has one output part that is not also an input part: the parameters in the Java or LotusScript code map to the `<wsdl:input>` message for the operation; the return value maps to the `<wsdl:output>` message. In other cases, for Java: output parts and inout parts map to parameters that are classes that implement the interface `javax.xml.rpc.holders.Holder` (`StringHolder`, `FloatHolder`, and so on); for LotusScript: output and inout parts similarly map to `HOLDER` classes defined in `lssxd.lss` (`STRING_HOLDER`, `SINGLE_HOLDER`, and so on). Operations may also return errors to the caller, either through a fault sub-class that corresponds to some `<wsdl:fault>` message specified for the operation in the WSDL document, or through direct use of the WS_FAULT base class (LotusScript), or the `lotus.domino.types.Fault` or `java.lang.Exception` classes (Java).
When you create a new Java Web service, a skeleton implementation class named Untitled is created. If you import a WSDL document into a Web service, an implementation class is generated based on the WSDL content. The public prototype methods, functions, and subs in the class correspond to the Web service operations defined for the first <port> element of the first <service> element in the WSDL document. Other classes or types referenced in the prototype methods may also be generated from the WSDL document, along with their public data members. If you change a prototype interface, the corresponding WSDL document changes when the Web service is saved. You can monitor and prevent such changes with the "Warn if the WSDL interface is modified" property.

If you create an implementation class from scratch, its prototype methods, functions, and subs determine the messages and operations defined in the WSDL document.

- Libraries -- LotusScript code can be stored in libraries with the following restrictions. Namespace constants must be in the same location as the implementation class. Enumeration classes and associated "List" objects and constants must be in the same location as the implementation class.

- Java caveats -- Java code is cached and shared over multiple instances of a Web service, enhancing performance but imposing the following restraints: do not use static data as changes made by one instance affect all instances; avoid synchronization as the scope of the classes and objects being synchronized may cross instances.

**Web services on a Domino server**

A Web service must exist in a database on a Domino server to be used. Replication is a simple way to move locally designed Web services to servers. Web services on production servers should be properly protected by appropriately signing the Web service and setting the Security tab.

To invoke a Domino Web service, a consumer uses the Domino URL command for opening the design element in a SOAP-encoded Web services HTTP POST request. Domino executes the appropriate implementation code based on the request and sends a SOAP-encoded response to the consumer.

The syntax for the Domino URL command for a Web service is as follows:

http://servername/databasename.nsf/webservicename?OpenWebService

If you send the URL command for opening a Web service to a Domino server in an HTTP GET request (for example, by typing the command in a browser), Domino responds with the Web service’s port name and the names of the operations for the port.

The following Domino URL command retrieves the WSDL document for the Web service in response to an HTTP GET:

http://servername/databasename.nsf/webservicename?WSDL

The SOAP address for a Web service is the Domino server responding to the WSDL URL command. The SOAP address in a WSDL document generated by Domino Designer is not valid because the Web service is not properly exposed on a server at that time.

**Java and LotusScript mappings**

**Implementation code**

The implementation code for a Web service is in its own public class:

- The class must be the one identified as "PortType class" in the Basics tab of the properties box.

- The class contains one public method, function, or sub for each operation in the Web service. The name of the method, function, or sub is the name of the operation in the WSDL document. Private methods, functions, and subs are not exposed.

- Input message parts are passed as parameters to the methods, functions, or subs. Output message parts are passed as method or function return values, or as parameters from methods or subs. Fault message parts specify service-specific return values, which an operation can issue when an error occurs.
When a consumer invokes an operation for a Web service, Domino invokes the corresponding method, function, or sub, passing the input message as parameters. When the method or function exits normally, Domino uses the return value or parameters as the output message for the operation.

**Java declarations and initialization**
The Java code may include the following import for fault data types, XML Schema data types, and holder classes not defined in standard Java packages.

```java
import lotus.domino.types.*
```

The lotus.domino package contains the class WebServiceBase which contains the static method getCurrentSession. Use this method to get a Session object representing the Domino session, typically in the no-parameter constructor for the implementation class. The Domino Web services runtime engine invokes the implementation class no-parameter (default) constructor and ignores all other constructors. The default constructor may not be explicit, but when it is, it must be declared *public*. Other constructors may also be present, for use by the implementation.

```java
Session.getAgentContext() gets the Web service context. The following AgentContext properties apply to Web services: getCurrentAgent() gets the current Web service; getCurrentDatabase() gets the current database. For getCurrentAgent(), the following Agent properties apply to the current Web service: getName(), getOwner(), getCommonOwner(), getOnBehalfOf(), getComment(), getHttpURL(), getNotesURL(), getParent(), and getLockHolders().
```

System.out prints to the server console and to log.nsf on the server.

**LotusScript options, declarations, and initialization**
The (Options) may include the following INCLUDE statement. The included file defines data types for mapping to those in the WSDL document.

```
%INCLUDE "1xsd.lss"
```

The (Declarations) typically include Dim statements for NotesSession and NotesDatabase (for the current database) objects.

Use Sub NEW in the implementation class for code that is always executed prior to the invoked operation, such as setting the NotesSession object. Creating a new NotesSession object gets a Domino session. Sub NEW may not have arguments in its signature. The same restriction applies to Sub NEW for all classes received or returned by any Web Service method (including classes contained in other classes).

The following NotesSession properties apply to Web services: CurrentAgent gets the current Web service; CurrentDatabase gets the current database. For CurrentAgent, the following NotesAgent properties apply to the current Web service: Name, Owner, CommonOwner, OnBehalfOf, Comment, HttpURL, NotesURL, Parent, and LockHolders.

MessageBox prints to the server console and to log.nsf on the server.

The lxsd.lss file defines the following classes: BOOLEAN_HOLDER, BOOLEANARRAY_HOLDER, BYTE_HOLDER, BYTEARRAY_HOLDER, DOUBLE_HOLDER, DOUBLEARRAY_HOLDER, INTEGER_HOLDER, INTEGERARRAY_HOLDER, LONG_HOLDER, LONGARRAY_HOLDER, SINGLE_HOLDER, SINGLEARRAY_HOLDER, STRING_HOLDER, STRINGARRAY_HOLDER, VARIANT_HOLDER, VARIANTARRAY_HOLDER. These classes are used for output and inout parameters. They have a public variable named "Value" that you can get and set. In addition, lxsd.lss defines classes for XML Schema data types as well as holder classes for them, for example, XSD_BOOLEAN and XSD_BOOLEAN_HOLDER. See the XSD data types under "Data descriptions" below.
**Data descriptions**

The following basic data types map one-for-one between the WSDL document and the corresponding Java or LotusScript code. Except as noted, the LotusScript XSD data types are defined in lsxsd.lss and have the following function and sub:

```java
string = GetValueAsString()
Call SetValueFromString(string)
```

The following LotusScript XSD classes are proxy classes: XSD_BASE64BINARY, XSD_DATE, XSD_DATETIME, XSD_HEX_BINARY, and XSD_TIME. Like the other XSD classes, they have a public variable named "Value" that you can get and set. However, the "Value" variable is not of type String.

The information for the classes XSD_BASE64BINARY and XSD_HEX_BINARY is passed in NotesStream format. These classes have the following function and sub:

```java
NotesStream = GetValueAsNotesStream()
Call SetValueFromNotesStream(NotesStream)
```

The information for the classes XSD_DATE, XSD_DATETIME, and XSD_TIME is passed in NotesDateTime format. These classes have the following function and sub:

```java
NotesDateTime = GetValueAsNotesDateTime()
Call SetValueFromNotesDateTime(NotesDateTime)
```

The data types map both ways (WSDL document to code and vice-versa) except as noted.

<table>
<thead>
<tr>
<th>WSDL</th>
<th>Java</th>
<th>LotusScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsd:anyType</td>
<td>java.lang.Object</td>
<td>XSD_ANYTYPE</td>
</tr>
<tr>
<td>xsd:anyURI</td>
<td>lotus.domino.types.URI</td>
<td>XSD_ANYURI</td>
</tr>
<tr>
<td>soapenc:base64</td>
<td>byte[] (from WSDL only)</td>
<td>ByteArray_Holder (from WSDL only)</td>
</tr>
<tr>
<td>soapenc:base64Binary</td>
<td>byte[] (from WSDL only)</td>
<td>ByteArray_Holder (from WSDL only)</td>
</tr>
<tr>
<td>xsd:base64Binary</td>
<td>byte[]</td>
<td>XSD_BASE64BINARY</td>
</tr>
<tr>
<td>soapenc:boolean</td>
<td>java.lang.Boolean (from WSDL only)</td>
<td>XSD_BOOLEAN (from WSDL only)</td>
</tr>
<tr>
<td>xsd:boolean</td>
<td>boolean</td>
<td>XSD_BOOLEAN (to WSDL only)</td>
</tr>
<tr>
<td>soapenc:byte</td>
<td>java.lang.Byte (from WSDL only)</td>
<td>XSD_BYTE (from WSDL only)</td>
</tr>
<tr>
<td>xsd:byte</td>
<td>byte</td>
<td>XSD_BYTE</td>
</tr>
<tr>
<td>xsd:date</td>
<td>java.util.Date</td>
<td>XSD_DATE</td>
</tr>
<tr>
<td>xsd:dateTime</td>
<td>java.util.Calendar</td>
<td>XSD_DATETIME</td>
</tr>
<tr>
<td>soapenc:decimal</td>
<td>java.math.BigDecimal (from WSDL only)</td>
<td>XSD_DECIMAL (from WSDL only)</td>
</tr>
<tr>
<td>xsd:decimal</td>
<td>java.math.BigDecimal</td>
<td>XSD_DECIMAL</td>
</tr>
<tr>
<td>soapenc:double</td>
<td>java.lang.Double (from WSDL only)</td>
<td>XSD_DOUBLE (from WSDL only)</td>
</tr>
<tr>
<td>xsd:double</td>
<td>double</td>
<td>XSD_DOUBLE</td>
</tr>
<tr>
<td>xsd:duration</td>
<td>lotus.domino.types.Duration</td>
<td>XSD_DURATION</td>
</tr>
<tr>
<td>xsd:ENTITY</td>
<td>lotus.domino.types.Entity</td>
<td>XSD_ENTITY</td>
</tr>
<tr>
<td>xsd:ENTITIES</td>
<td>lotus.domino.types.Entities</td>
<td>XSD_ENTITIES</td>
</tr>
<tr>
<td>WSDL</td>
<td>Java</td>
<td>LotusScript</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>soapenc:float</td>
<td>java.lang.Float (from WSDL only)</td>
<td>XSD_FLOAT (from WSDL only)</td>
</tr>
<tr>
<td>xsd:float</td>
<td>float</td>
<td>Single</td>
</tr>
<tr>
<td></td>
<td>java.lang.Float (to WSDL only)</td>
<td>XSD_FLOAT (to WSDL only)</td>
</tr>
<tr>
<td>xsd:gDay</td>
<td>lotus.domino.types.GDay</td>
<td>XSD_GDAY</td>
</tr>
<tr>
<td>xsd:gMonth</td>
<td>lotus.domino.types.GMonth</td>
<td>XSD_GMONTH</td>
</tr>
<tr>
<td>xsd:gMonthDay</td>
<td>lotus.domino.types.GMonthDay</td>
<td>XSD_GMONTHDAY</td>
</tr>
<tr>
<td>xsd:gYear</td>
<td>lotus.domino.types.GYear</td>
<td>XSD_GYEAR</td>
</tr>
<tr>
<td>xsd:gYearMonth</td>
<td>lotus.domino.types.GYearMonth</td>
<td>XSD_GYEARMONTH</td>
</tr>
<tr>
<td>xsd:hexBinary</td>
<td>lotus.domino.types.HexBinary</td>
<td>XSD_HEXBINARY</td>
</tr>
<tr>
<td>xsd:ID</td>
<td>lotus.domino.types.Id</td>
<td>XSD_ID</td>
</tr>
<tr>
<td>xsd:IDREF</td>
<td>lotus.domino.types.IDRef</td>
<td>XSD_IDREF</td>
</tr>
<tr>
<td>xsd:IDREFS</td>
<td>lotus.domino.types.IDRefs</td>
<td>XSD_IDREFS</td>
</tr>
<tr>
<td>soapenc:int</td>
<td>java.lang.Integer (from WSDL only)</td>
<td>XSD_INT (from WSDL only)</td>
</tr>
<tr>
<td>xsd:int</td>
<td>int</td>
<td>XSD_INT (to WSDL only)</td>
</tr>
<tr>
<td></td>
<td>java.lang.Integer (to WSDL only)</td>
<td>XSD_INT (to WSDL only)</td>
</tr>
<tr>
<td>soapenc:integer</td>
<td>java.math.BigInteger (from WSDL only)</td>
<td>XSD_INTEGER (from WSDL only)</td>
</tr>
<tr>
<td>xsd:integer</td>
<td>java.math.BigInteger</td>
<td>XSD_INTEGER</td>
</tr>
<tr>
<td>xsd:language</td>
<td>lotus.domino.types.Language</td>
<td>XSD_LANGUAGE</td>
</tr>
<tr>
<td>soapenc:long</td>
<td>java.lang.Long (from WSDL only)</td>
<td>XSD_LONG (from WSDL only)</td>
</tr>
<tr>
<td>xsd:long</td>
<td>long</td>
<td>XSD_LONG</td>
</tr>
<tr>
<td></td>
<td>java.lang.Long (to WSDL only)</td>
<td>XSD_LONG (to WSDL only)</td>
</tr>
<tr>
<td>xsd:Name</td>
<td>lotus.domino.types.Name</td>
<td>XSD_NAME</td>
</tr>
<tr>
<td>xsd:NCName</td>
<td>lotus.domino.types.NCName</td>
<td>XSD_NCNAME</td>
</tr>
<tr>
<td>xsd:negativeInteger</td>
<td>lotus.domino.types.NegativeInteger</td>
<td>XSD_NEGATIVEINTEGER</td>
</tr>
<tr>
<td>xsd:NMTOKEN</td>
<td>lotus.domino.types.NMToken</td>
<td>XSD_NMTOKEN</td>
</tr>
<tr>
<td>xsd:NMTOKENS</td>
<td>lotus.domino.types.NMTokens</td>
<td>XSD_NMTOKENS</td>
</tr>
<tr>
<td>xsd:nonNegativeInteger</td>
<td>lotus.domino.types.NonNegativeInteger</td>
<td>XSD_NONNEGATIVEINTEGER</td>
</tr>
<tr>
<td>xsd:nonPositiveInteger</td>
<td>lotus.domino.types.NonPositiveInteger</td>
<td>XSD_NONPOSITIVEINTEGER</td>
</tr>
<tr>
<td>xsd:NOTATION</td>
<td>lotus.domino.types.Notation</td>
<td>XSD_NOTATION</td>
</tr>
<tr>
<td>xsd:normalizedString</td>
<td>lotus.domino.types.NormalizedString</td>
<td>XSD_NORMALIZEDSTRING</td>
</tr>
<tr>
<td>xsd:positiveInteger</td>
<td>lotus.domino.types.PositiveInteger</td>
<td>XSD_POSITIVEINTEGER</td>
</tr>
<tr>
<td>xsd:QName</td>
<td>javax.xml.namespace.QName</td>
<td>XSD_QNAME</td>
</tr>
<tr>
<td>soapenc:short</td>
<td>java.lang.Short (from WSDL only)</td>
<td>XSD_SHORT (from WSDL only)</td>
</tr>
<tr>
<td>xsd:short</td>
<td>short</td>
<td>XSD_SHORT (to WSDL only)</td>
</tr>
<tr>
<td></td>
<td>java.lang.Short (to WSDL only)</td>
<td>XSD_SHORT (to WSDL only)</td>
</tr>
<tr>
<td>soapenc:string</td>
<td>java.lang.String (from WSDL only)</td>
<td>String (from WSDL only)</td>
</tr>
<tr>
<td>xsd:string</td>
<td>java.lang.String</td>
<td>String (to WSDL only)</td>
</tr>
<tr>
<td>xsd:time</td>
<td>lotus.domino.types.Time</td>
<td>XSD_TIME</td>
</tr>
<tr>
<td>xsd:token</td>
<td>lotus.domino.types.Token</td>
<td>XSD_TOKEN</td>
</tr>
<tr>
<td>xsd:unsignedByte</td>
<td>lotus.domino.typesUnsignedByte</td>
<td>Byte</td>
</tr>
</tbody>
</table>
## Arrays

Arrays map to XML Schema `<complexType>` entities in WSDL, with the following attributes and subelements:

- `name = "ArrayOf_{xsd}_"` plus the name of the data type; for example, `ArrayOf_xsd_int`
- `restriction base = "soapenc:Array"`
- `attribute ref = "soapenc:arrayType"`
- `wsdl:arrayType = "xsd:"` plus the data type plus `"[]"`; for example, `xsd:int[]`

For example, an array of type `int` (Java) or `Long` (LotusScript) generates the following WSDL:

```xml
<wsdl:types>
  <schema targetNamespace="urn:DefaultNamespace"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="ArrayOf_xsd_int">
      <complexContent>
        <restriction base="soapenc:Array">
          <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:int[]"/>
        </restriction>
      </complexContent>
    </complexType>
  </schema>
</wsdl:types>
```

Arrays may also map to individual `<element>` entities in WSDL, where a `maxOccurs` attribute is present with a value greater than 1. This second form of array representation may appear in any externally-defined WSDL document, but will only appear in Designer-generated WSDL for a Java Web service, and only then for Java value type array members with indexed accessors.

## Classes

Classes (sometimes called value types) that pass to or from a Web service operation usually map to XML Schema `<complexType>` elements in WSDL, with subelements representing each exposed data member. For example:

```xml
<xsd:complexType name="telephone">
  <xsd:sequence>
    <xsd:element name="areaCode" type="xsd:int"/>
    <xsd:element name="exchange" type="xsd:int"/>
    <xsd:element name="number" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

For LotusScript, exposed class data members are those members declared public.

For Java, exposed class data members are either members that are public, or members that have public get and set methods characteristic of a Java Bean. Recognizable Java value type classes must have a public default constructor (no arguments).

## Enumerations

Enumerations map to XML Schema `<simpleType>` elements in WSDL, with the following characteristics:

- `restriction subelement with "base" attribute for the underlying type`
• enumeration facets (subelements, one or more), each specifying a member of the enumeration

For example, an enumeration of vegetables containing members of type String generates the following WSDL:

```xml
<xsd:simpleType name="vegetableType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Carrot"/>
        <xsd:enumeration value="Lettuce"/>
        <xsd:enumeration value="Ketchup"/>
    </xsd:restriction>
</xsd:simpleType>
```

Both LotusScript and Java implementations of an enumeration reflect a specific pattern comprised of:
• a constant for each enumeration member value
• a class for representing any enumeration member
• a collection of the enumeration member class instances, representing the enumeration itself
• initializer code that populates the enumeration just prior to Web service method invocation

All identifying characteristics (bolded in the examples) must be present in the implementation for any WSDL generation to detect and correctly represent an enumeration.

**Note:** the Java enumeration pattern prohibits the presence of a Java Bean set: method, i.e. a setValue method with one parameter, where the parameter type matches the getValue method return type.

**Faults**

Faults communicate error conditions back to the caller, and may be emitted by any Web service method. They may be explicit to the WSDL contract (specified in the WSDL document for one or more service operations), or just implicit to the WSDL contract (implemented-only, as part of some Service method’s signature).

Explicit faults appear in WSDL as optional `<wsdl:fault>` elements specified for an operation, appearing alongside `<wsdl:input>` and `<wsdl:output>` elements. For example:

```xml
<wsdl:operation name="getStockQuote">
    <wsdl:input message="getStockQuoteRequest"/>
    <wsdl:output message="getStockQuoteResponse"/>
    <wsdl:fault name="InvalidSymbolFault" message="InvalidSymbolFaultMessage"/>
</wsdl:operation>
```

A `<wsdl:fault>` element is typed by its `message` attribute, just like an operation’s `<wsdl:input>` or `<wsdl:output>` element.

Accordingly, a WSDL fault, through its associated message `part` type(s), maps to some class in the LotusScript or Java implementation, which is sub-classed from either WS_FAULT (LotusScript) or `lotus.domino.types.Fault` (Java).

This implementation fault subclass maps to the data members defined for the WSDL fault’s associated message, in exactly the same manner as implementation value types (see “Classes”, above) map to their associated WSDL constructs, usually WSDL `<complexType>`’s.

For example, given the following WSDL `<complexType>` (specified here as the type for the example "InvalidSymbolFaultMessage" referenced above):

```xml
<wsdl:types>
    <xsd:schema ...>
        <xsd:complexType name="InvalidSymbolFault">
            <xsd:sequence>
                <xsd:element name="TickerSymbol" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:schema>
</wsdl:types>
```
This complexType maps to LotusScript as:

```lotusscript
Class InvalidSymbolFaultMessage As WS_FAULT

Public TickerSymbol As String
Public ApplicationCode As Integer

Sub NEW
End Sub

End Class
```

and maps to Java as:

```java
public class InvalidSymbolFaultMessage extends lotus.domino.types.Fault {
    private java.lang.String tickerSymbol;
    private int applicationCode;

    public InvalidSymbolFaultMessage() {
    }
    public InvalidSymbolFaultMessage(java.lang.String tickerSymbol, int applicationCode) {
        this.tickerSymbol = tickerSymbol;
        this.applicationCode = applicationCode;
    }
    public java.lang.String getTickerSymbol() {
        return this.tickerSymbol;
    }
    public int getApplicationCode() {
        return this.applicationCode;
    }
    // helper methods...
}
```

When a Web service operation has no need to send additional, service-specific, data back to the caller, then explicit WSDL faults are not needed - an implicit fault will suffice, instead.

Implicit faults are always possible, are issued by the Web service framework for a variety of invocation errors, but may be emitted, too, by any service method exposed in the implementation PortType class.

Note, too, that adding or removing implicit faults from an implementation method signature is not treated as a change to the Web service contract, and so does not by itself cause WSDL regeneration when the Web service is saved.

Service method fault handling is the same for both explicit and implicit faults. All identifying characteristics are **bolded** in the examples.

**Lists**
The XML Schema list type is a form of `<simpleType>` whose values are represented as a single, white-space delimited list. For example, the following sequence of vegetables (each token separated by a space character):

```
Carrot Lettuce Ketchup
```

is a legal value for an instance of this list type:
<xsd:simpleType name="listOfVegetables">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>

List values lend themselves to implementation as arrays, and therefore map to array-like implementations in both LotusScript and Java. The identifying characteristics for WSDL generation are **bolded** in the example.

**Namespaces**

XML Schema uses namespace qualifiers to differentiate among like-named but differently-defined constructs, especially data types, for example:

```xml
<types>
  <xsd:schema targetNamespace="urn:MyAddressBook" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="telephone">
      <xsd:sequence>
        <xsd:element name="areaCode" type="xsd:int"/>
        <xsd:element name="exchange" type="xsd:int"/>
        <xsd:element name="number" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
</types>
```

Here, the "telephone" data type is specified to be part of the "MyAddressBook" namespace.

Domino Java Web service implementations represent mapped namespaces as Java packages, and so the above "telephone" data type would map to the following Java package and class:

```java
package MyAddressBook;
public class Telephone {
  ...
}
```

LotusScript Web service implementations have no package construct to use in this manner, and the legal length for a LotusScript identifier is limited, compared to Java. So WSDL XML Schema namespaces are mapped either as namespace constants together with short suffixes appended to LotusScript class names, or as a single, "DefaultNamespace" constant, applied as the default namespace.

For example, the above "telephone" data type would import from WSDL to LotusScript as the following:

```lotus
Const n1 = "MyAddressBook" 'string constant value can be many characters long
Class Telephone_n1...
End Class
```

It could also be specified using the "DefaultNamespace" constant, as:

```lotus
Const DefaultNamespace = "MyAddressBook" Class Telephone 'no suffix specified
...
End Class
```

**Unsupported constructs**

For this release, the following WSDL or XML Schema constructs have limited or no supported mappings to LotusScript or Java, and are rejected (or ignored, if indicated) by the Designer "Import WSDL" feature:

- **any** element
- **anyAttribute** element
- **choice** element, if nested, optional, or maxOccurs > 1 (arrays)
- **redefine** element
- **sequence** element, if maxOccurs > 1 (arrays)
- **union** simple type
- **vector** and **map** (APACHESOAP namespace) - no LotusScript mappings
- MIME attachments
• simple type facets - ignored (except for enumeration)
• attribute, element default values - ignored
• mixed content (character data alongside complex type subelements) -- ignored

Similarly, the following LotusScript and Java constructs have no supported mappings to WSDL XML Schema, and are rejected by the Designer “Export WSDL”, “Show WSDL”, or “Save” features, when exposed as some Web service operation argument or return type:
• LotusScript list and currency data types
• LotusScript class Sub NEW and DELETE (constructor/destructor) arguments
• LotusScript LSX objects
• backend classes (except for NotesDomDocumentNode for use with message programming model)
• most Java non-Bean classes

Examples: Web services

Example 1: Web service without complex data types
Here is a Web service with two operations. Each operation has an input message with 2 parts and an output message.

Property box settings
• PortType class = WebServiceOne
• Programming model = RPC
• SOAP message format = RPC/encoded
• Port type name = WebServiceOne
• Service element name = WebServiceOneService
• Service port name = Domino

LotusScript:
%INCLUDE "lsxsd.lss"
Class WebServiceOne

Sub NEW
End Sub

Function operation1(input1 As String, input2 As String) As String
End Function

Function operation2(input1 As String, input2 As String) As String
End Function

End Class

Java:
import lotus.domino.*;
import lotus.domino.types.*;

public class WebServiceOne {
    public java.lang.String operation1(
        java.lang.String input1,
        java.lang.String input2) {
        return null;
    }
    public java.lang.String operation2(
        java.lang.String input1,
java.lang.String input2) {
    return null;
}

WSDL

<xml version="1.0" encoding="UTF-8" ?
<wsdl:definitions targetNamespace="urn:DefaultNamespace"
 xmlns="http://schemas.xmlsoap.org/wsdl/
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:impl="urn:DefaultNamespace" xmlns:intf="urn:DefaultNamespace"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:message name="OPERATION2Request">
    <wsdl:part name="INPUT1" type="xsd:string" />
    <wsdl:part name="INPUT2" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="OPERATION1Response">
    <wsdl:part name="OPERATION1Return" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="OPERATION1Request">
    <wsdl:part name="INPUT1" type="xsd:string" />
    <wsdl:part name="INPUT2" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="OPERATION2Response">
    <wsdl:part name="OPERATION2Return" type="xsd:string" />
  </wsdl:message>
  <wsdl:portType name="WebServiceOne">
    <wsdl:operation name="OPERATION1" parameterOrder="INPUT1 INPUT2">
      <wsdl:input message="impl:OPERATION1Request"
            name="OPERATION1Request" />
      <wsdl:output message="impl:OPERATION1Response"
            name="OPERATION1Response" />
    </wsdl:operation>
    <wsdl:operation name="OPERATION2" parameterOrder="INPUT1 INPUT2">
      <wsdl:input message="impl:OPERATION2Request"
            name="OPERATION2Request" />
      <wsdl:output message="impl:OPERATION2Response"
            name="OPERATION2Response" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="DominoSoapBinding" type="impl:WebServiceOne">
    <wsdlsoap:binding style="rpc"
          transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="OPERATION1">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="OPERATION1Request">
        <wsdlsoap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            namespace="urn:DefaultNamespace" use="encoded" />
      </wsdl:input>
      <wsdl:output name="OPERATION1Response">
        <wsdlsoap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            namespace="urn:DefaultNamespace" use="encoded" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="OPERATION2">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="OPERATION2Request">
        <wsdlsoap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            namespace="urn:DefaultNamespace" use="encoded" />
      </wsdl:input>
      <wsdl:output name="OPERATION2Response">
        <wsdlsoap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            namespace="urn:DefaultNamespace" use="encoded" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
Example 2: Accessing a Domino session

This example demonstrates Example 1 with code added to access a Domino session.

**LotusScript:**
```
%INCLUDE "lsxsd.lss"

Dim session As NotesSession
Dim db As NotesDatabase

Class WebServiceOne

    Sub NEW
        Set session = New NotesSession
        Set db = session.GetCurrentDatabase
    End Sub

    Function operation1(input1 As String, input2 As String) As String
        End Function

    Function operation2(input1 As String, input2 As String) As String
        End Function

End Class
```

**Java:**
```
import lotus.domino.*;
import lotus.domino.types./*;

public class WebServiceOne {

    private Session session;
    private AgentContext ac;
    private Database db;

    public WebServiceOne() throws NotesException {
        session = WebServiceBase.getCurrentSession();
        ac = session.getAgentContext();
        db = ac.getCurrentDatabase();
    }

    public java.lang.String operation1(java.lang.String input1, java.lang.String input2) {
        return null;
    }

    public java.lang.String operation2(java.lang.String input1, java.lang.String input2) {
        return null;
    }
}
```
Example 3: Array
This example demonstrates the coding for an array of type float:

**LotusScript**

() As Single

**Java**

float[]

**WSDL**

<complexType name="ArrayOf_xsd_float">
    <complexContent>
        <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:float[]"/>
        </restriction>
    </complexContent>
</complexType>

Example 4: Complex data element
This example demonstrates the coding for a data element that contains two strings:

**WSDL**

<complexType name="TwoStrings">
    <sequence>
        <element name="string1" nillable="true" type="xsd:string" />
        <element name="string2" nillable="true" type="xsd:string" />
    </sequence>
</complexType>

**LotusScript**

Note that string1 and string2 must be public.

%INCLUDE "lxsxd.lss"

Class TwoStrings

    Public string1 As XSD_STRING
    Public string2 As XSD_STRING

    Sub NEW
        End Sub
    End Class

**Java**

public class TwoStrings {
    private java.lang.String string1;
    private java.lang.String string2;

    public TwoStrings() {

    }

    public java.lang.String getString1() {
        return string1;
    }

    public void setString1(java.lang.String string1) {
        this.string1 = string1;
    }

    public java.lang.String getString2() {
        return string2;
    }
}
public void setString2(java.lang.String string2) {
    this.string2 = string2;
}

Examples: Web services data descriptions

Example 1: Enumerations
This example illustrates an enumeration of vegetables containing members of type String. All identifying characteristics (bolded in the examples below) must be present in the implementation for any WSDL generation to detect and correctly represent an enumeration.

WSDL

```
<xsd:simpleType name="vegetableType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Carrot"/>
        <xsd:enumeration value="Lettuce"/>
        <xsd:enumeration value="Ketchup"/>
    </xsd:restriction>
</xsd:simpleType>
```

LotusScript enumeration pattern

[Declarations]
'member value constants:Const VegetableType_carrot$ = "Carrot"
'constant name prefix matches enum class nameConst VegetableType_lettuce$ = "Lettuce"
Const VegetableType_ketchup$ = "Ketchup"
'member instances:Dim Carrot_VegetableType As VegetableType
'instance name suffix matches enum class nameDim Lettuce_VegetableType As VegetableType
Dim Ketchup_VegetableType As VegetableType

'LotusScript list of members:Dim Enum_VegetableType List As VegetableType
'prefix "Enum_" includes a trailing underscore
'enumeration class:Class VegetableType
'class name matches list name suffix and list type Public Value As String  'first public property;
'type matches constants type above Sub NEW

End Sub
'initializer:
Sub Initialize (elem As String)
    Let Value = elem
    Set Enum_VegetableType(Cstr(Value)) = Me
End Sub

'optional helper functions
Function Equals (you As VegetableType) As Boolean...
Function ToString As String...
End Class

[global Sub's]
'initialization (called from Sub NEW of the PortType class)
Sub VegetableType_Initialize
    Set Carrot_VegetableType = New VegetableType
    Call Carrot_VegetableType.Initialize(VegetableType_Carrot)
    Set Lettuce_VegetableType = New VegetableType
    Call Lettuce_VegetableType.Initialize(VegetableType_Lettuce)
    Set Ketchup_VegetableType = New VegetableType
    Call Ketchup_VegetableType.Initialize(VegetableType_Ketchup)
End Sub

'optional helper functions
Function VegetableType_FromValue (value As String) As VegetableType...
Public Function VegetableType_FromString (value As String) As VegetableType...

Java enumeration pattern

```
public class VegetableType {
    private java.lang.String _value_;
    private static java.util.HashMap _table_ = new java.util.HashMap();
```
// Constructor
protected VegetableType(java.lang.String value) {
    _value_ = value;
    _table_.put(_value_,this);
}

// enum values; type matches private "value" data member type:
public static final java.lang.String _Carrot = "Carrot";
public static final java.lang.String _Lettuce = "Lettuce";
public static final java.lang.String _Ketchup = "Ketchup";
public static final VegetableType Carrot = new VegetableType(_Carrot);
public static final VegetableType Lettuce = new VegetableType(_Lettuce);
public static final VegetableType Ketchup = new VegetableType(_Ketchup);

// return type matches the type of the private "value" data member:
public java.lang.String getValue() { return _value_;}

// parameter type matches "getValue" return type:
public static VegetableType fromValue(java.lang.String value) throws java.lang.IllegalVegetableException {
    VegetableType enum = (VegetableType)_table_.get(value);
    if (enum==null) throw new java.lang.IllegalVegetableException();
    return enum;
}

public static VegetableType fromString(java.lang.String value) throws java.lang.IllegalVegetableException {
    return fromValue(value);
}

public java.lang.String toString() {
    return _value_;}

public boolean equals(java.lang.Object obj) {return (obj == this);}

public int hashCode() { return toString().hashCode();}

Example 2: Faults
This example shows service method fault handling is the same for both explicit and implicit faults:

LotusScript fault (explicit)
Class FaultServicePortType

Function getStockQuote( tickerSymbol As String, Fault1 As InvalidSymbolFaultMessage ) As Single

...
' WS_FAULT base class properties
Call Fault1.setFault(True) ' required for fault activation
Call Fault1.setFaultString("getQuote InvalidTickerFaultMessage") ' optional fault message

' fault subclass properties
Let Fault1.TickerSymbol = tickerSymbol
Let Fault1.ApplicationCode = "12345"

End Function

End Class

LotusScript fault (implicit)
Function getStockQuote( tickerSymbol As String, Fault1 As WS_FAULT ) As Single

...
' WS_FAULT base class properties
Call Fault1.setFault(True) ' required for fault activation
Call Fault1.setFaultString("getQuote InvalidTickerFaultMessage") ' optional fault message
End Function
Note: Any faults for a LotusScript service implementation method must appear at the end of the method argument (any WS_FAULT base class or its subclasses).

Java fault (explicit)

```java
class FaultServicePortType {
    public float getStockQuote(java.lang.String tickerSymbol) throws InvalidSymbolFaultMessage {
        ...
        throw new InvalidSymbolFaultMessage(tickerSymbol, 12345);
    }
}
```

Java fault (implicit)

```java
class FaultServicePortType {
    public float getStockQuote(java.lang.String tickerSymbol) throws lotus.domino.types.Fault {
        ...
        java.lang.Exception e = new java.lang.Exception("Thrown exception");
        throw lotus.domino.types.Fault.makFault(e); // static factory method
    }
}
```

or even simpler:

```java
class FaultServicePortType {
    public float getStockQuote(java.lang.String tickerSymbol) throws java.lang.Exception {
        ...
        throw new java.lang.Exception("Thrown exception");
    }
}
```

Note: Faults for a Java service implementation method appear in the conventional Java `throws` clause for the method.

Example 3: Lists

This example illustrates a list of vegetables, consisting of the three items Carrot, Lettuce, and Ketchup.

**WSDL**

```xml
<xsd:simpleType name="listOfVegetables">
    <xsd:list item-Type="xsd:string"/>
</xsd:simpleType>
```

**LotusScript list pattern**

```lotus
Class ListOfVegetables As XSD_LIST 'list base class
Public value() As String 'one public array property;
    'array type maps to a SimpleType

    'mandatory helper member and methods for de/serialization:
    Private initialized As Boolean
    Sub setListValueFromString (idx As Integer, value As String) If idx < 0 Then Error ErrArgOutOfRange
        If Not initialized Then
            Redim Me.value(0 To idx)
        End If
        Let initialized = True
    Else
        If idx > Ubound(Me.value) Then Redim Preserve Me.value(0 To idx)
        End If
        Let Me.value(idx) = Cstr(value)
    'cast must produce a value( ) array element type End Sub

    Function getValueAsString (idx As Integer) As String If Not initialized Then Error ErrArgOutOfRange
        getValueAsString = Cstr(value(idx))
    End Function

    Function getListLength () As Integer If Not initialized Then
        getListLength = 0
    Else
        getListLength = Ubound(value)+1
    End Function
```

Chapter 3. Programming Domino for Web Applications 109
End If
End Function

Note: If the "Public value( )" array member type is a mapped simple type class instead of a LotusScript built-in type, then some pattern statements change accordingly. For example, if the array member type is XSD_NCNAME, then in Sub setListValueFromString, the last statement:

Let Me.value(idx) = Cstr(value)

is replaced by two statements:

Set Me.value(idx) = New XSD_NCNAME 'creates an instance of the simple type class
Call Me.value(idx).setValueFromString(value)

Similarly, the last statement in Sub getListValueAsString is replaced by:

g getListValueAsString = value(idx).getValueAsString

Java list pattern

```java
public class ListOfVegetables implements lotus.domino.types.SimpleType {
    // SimpleType interface private java.lang.String[] value; // private instance member, array, 
    // named "value"
    // array type maps to a simple type // default constructor
    public ListOfWords() {
    }
    // constructor that accepts the private array member
    public ListOfWords(java.lang.String[] value) {
        this.value = value;
    }
    // simple type String constructor
    public ListOfWords(java.lang.String value) {
        StringTokenizer st = new StringTokenizer(value, "\r\n\t");
        this.value = new java.lang.String[st.countTokens()];
        for(int i = 0; st.hasMoreTokens(); i++) {
            this.value[i] = new java.lang.String(st.nextToken());
        }
    }
    // simple type toString method for serializing the array value
    public java.lang.String toString() {
        if (value == null)
            return null;
        String ret = new String();
        for (int i = 0; i < value.length; i++) {
            ret += value[i].toString();
            if ((i+i) < value.length)
                ret += " ";
        }
        return ret;
    }
    // public getter method for private array member
    public java.lang.String[] getValue() {
        return value;
    }
    // public setter method for private array member
    public void setValue(java.lang.String[] value) {
        this.value = value;
    }
}
```

110 Prgramming Guide, Volume 1: Overview and Formula Language
Server configuration for Web browsers

The Domino administrator is typically responsible for configuring the Domino server for proper rendering of Domino Web applications. However, developers of Web applications should be aware of the possible settings in the browser configuration file that may impact the performance or appearance of their applications.

The browser.cnf file

The browser.cnf file is installed automatically in the server data directory when you install Domino. It contains configuration settings for browsers used to render Domino Web applications. Specifically, this file is used by @BrowserInfo to determine browser properties.

There are two types of directives in this file: Property and Rule. The property directive defines a browser property that can be accessed by @BrowserInfo. The Rule directive specifies a regular expression pattern which is used to match the browser User-Agent header.

Typically, Web application developers will make use of this file to ensure that their applications work properly on the Domino Web server. However, because of the file’s location in the server data directory, it is likely that the Domino administrator will need to make any necessary modifications.

The following are browser.cnf settings that modify the behavior of the Web engine’s built-in HTML generation:

- CSSTableTabs - Boolean indicating whether browser can render CSS rules for table tabs
- DisableActionBarApplet - Boolean indicating whether browser can handle the ActionBar Applet
- DisableViewApplet - Boolean indicating whether browser can handle the View Applet
- DisableOutlineApplet - Boolean indicating whether browser can handle the Outline Applet
- DisableEditorApplet - Boolean indicating whether browser can handle the Editor Applet
- DHTMLSections - String indicating whether / how sections should render using client-side DHTML

Language cross-reference

Dynamic HTML sections

The Dynamic HTML (DHTML) used to render collapsable sections for display with Microsoft Internet Explorer has been enhanced and made available for use in other browsers which support DHTML. The enhancement consists of using the <DIV> tag instead of the <SPAN> tag to delimit the sections, and using standard JavaScript rather than IE-specific extensions for manipulating the sections.

Use of this enhancement is configured in the domino-data-directory\browser.cnf file. By default, the enhancement is turned off, meaning that the original DHTML is generated. To enable the enhancement, edit the browser.cnf file.

Here is an excerpt from the browser.cnf file describing the options:

```yaml
## DHTMLSections - String indicating whether / how sections should render using client-side DHTML
##
## Possible Values (case sensitive):
## None - client-side DHTML is not used
## Legacy - client-side DHTML generated as originally implemented on notes 6.0. This was an
```
### IE specific implementation.
### Standard - client-side DHTML using standards-based DHTML -- will work with IE 6 and
### Mozilla/5 based browsers.

### The property as shipped is configured to provide the same behavior as originally
### implemented in notes 6.0. Replace the rule with the ones commented out
### to extend the use of DHTML sections to other browsers and to avoid some IE 6
### problems with the legacy implementation.
### Note that some paragraph spacing in the Standard option may be different from the
### other options and from the Notes client.

Property DHTMLSections String None
  Rule Legacy MSIE [4-9]  # IE 4.x or higher
# Rule Legacy MSIE [4-5]
# Rule Standard MSIE [6-9]
  Rule Standard Mozilla/5
# Rule Standard Opera/7
Chapter 4. Formula Language Rules

Formula language provides syntax and @functions for evaluating constants and variables, and for performing simple logic. Variables can be fields in Notes documents or temporary variables (also called temporary fields) used only for the immediate formula.

This section provides these topics:
- Using the syntax rules
- Using variables
- Using constants
- Using operators
- Using @functions
- Using keywords
- Specifying form and view names in formulas
- Debugging formulas

For more information on how to use the formula language, see the Guidelines chapter, and the A-Z reference chapters for @Functions and @Commands.

Using the syntax rules

You understand formula language through its:
- Lexical elements
- General syntax rules

Lexical elements

A formula consists of one or more statements, each consisting of any of the following:
- Variables
- Constants
- Operators
- @Functions
- Keywords

See the applicable sections later in this chapter for descriptions of the statement components.

A value is a variable, a constant, the result of an @function, or the result of an expression formed by combining any of the foregoing elements with operators.

Examples: Lexical elements

1. D is a variable, := is an operator, and @Created is an @function without arguments. This formula assigns the creation date of a document to D.
   \[ D := @Created \]

2. @Trim is an @function with an argument. The argument, Subject, is a variable. This formula removes extraneous spaces from Subject.
   \[ @Trim(Subject) \]

3. @Prompt is an @function with three arguments. The first argument, [OK], is a keyword; the second and third arguments are text constants. This formula displays a dialog box.
   \[ @Prompt([OK]; "Update Complete"; "Your update has been posted") \]
4. SELECT is a reserved word and @All is an @function without arguments. This formula selects all documents for an operation, for example, for inclusion in a view.

   SELECT @All

5. LastName is a variable, + is an operator, "," is a constant, and FirstName is a variable. This formula concatenates LastName, a comma followed by a space, and FirstName.

   LastName + "," + FirstName

**General syntax rules**

A formula must follow these general syntax rules.

**Statement separators**

Separate multiple statements with semicolons.

   FIELD RegionalManager := AreaManager;
   FIELD AreaManager := @DeleteField

**Spaces**

You can place any number of spaces, including none, between operators, punctuation, and values. However, keywords must be delineated by at least one space, and spaces within text constants are significant.

For example, the following statements are equivalent.

   LastName + "," + FirstName;
   LastName+","+FirstName

In the following statement, at least one space must follow the reserved word SELECT.

   SELECT @All

**Case**

Case is not significant except within text constants. By convention, keywords such as FIELD are uppercase, and @function and @command names such as ProperCase are mixed uppercase and lowercase. You need not follow these conventions when typing, but Domino changes the case to conform to the conventions when saving a formula.

**Operators and values**

Two values must be separated by at least one operator.

**Using variables**

Variables are of two types:

- **Fields**
- **Temporary variables**

**Fields**

A formula has access to the fields in the document being processed. The name and type of each field is as specified in the database design.

**Data types**

Data types must be correct for the operation or @function being performed. For example, if TotalValue is a number field, you cannot display it directly with @Prompt because @Prompt requires a text argument. You must first convert the argument with @Text:

   @Prompt([OK]; "Value of MyNumber"; @Text(TotalValue));
Rich text fields
You can convert rich text fields to plain text with @Text as shown below:
plainText := @Text(Body);

Attachments and formatting, except for tabs and spaces, are lost.

Note: Conversion of rich text is new with Release 6.

You can also use @Abstract to convert rich text fields to plain text as shown below:
plainText := @Abstract([TryFit]; 100; ""; "Body");

Lists
Lists are fields that contain multiple values. Certain @functions and operators deal specifically with lists. For example, if Locations is a field that allows multiple values, the following statement returns the number of values in the list:
@Elements(Locations)

Field values
The value of a field is as specified in the document when a formula starts. The formula can modify the value of a field unless prohibited by access control. You must use the FIELD reserved word to modify a field -- otherwise, the variable is treated as a temporary variable. The FIELD reserved word can also be used to create a new field in the current document. The following formula writes a value to the text field Subject.
FIELD Subject := "No Subject"

Null fields
A null field is equivalent to the text constant "" (empty quotation marks). The following example tests the field named Subject in the current document. If the value of Subject is null, it is reset to "No Subject." Otherwise, its value does not change.
FIELD Subject := @If(Subject=""; "No Subject"; Subject)

Since "" is a text constant, you should avoid its use in non-text fields. In particular, editable non-text fields should use a default formula to ensure that the field contains a value of the correct type.

Deleting fields
Use @DeleteField to delete a field from a document.
FIELD BodyText := @DeleteField

Form fields
If the form used to create a document is not stored in the document, a field named Form is available and contains the name of the form. If the form is stored in the document, fields named $TITLE, $Info, $WindowTitle (if the design specifies a window title), and $Body are available. $TITLE contains the name of the form. The following example, which works in a button or hotspot, displays the name of the form used to create the current document.
@Prompt([OK]; "Form"; @If(@IsAvailable(Form); Form; $TITLE))

Choose File - Document Properties, and select Fields to see what fields are in a document.

You can remove a form stored in a document by deleting $TITLE, $Info, $WindowTitle, and $Body. You must then create a Form field and place in it the name of a form in the database. The following formula, which works as an agent, removes the form Travel Request stored in the current document and replaces it with the database form Travel Arrangements.
Temporary variables

A temporary variable exists only within a formula. Its scope is that formula and it has no attributes other than the ones assigned to it within the formula.

The syntax for creating a temporary variable is:

```
variableName := value
```

The variable takes the type of the value on the righthand side of the equation. This value can be any of the field types or boolean.

Boolean data types are returned by certain @functions and have a value of 1 (True) or 0 (False).

Variables with String data types cannot contain more than 2,048 characters. Concatenate multiple variables to handle larger amounts of text. For example, instead of trying to set the variable "Source" equal to a 3,400-character text block, set the variable "Source1" equal to the first 1,700-character text block and "Source2" to the second 1,700 characters. Concatenate the variables as follows:

```
@Prompt([OK];"Length";@Text(@Length((Source1+Source2))))
```

Using a variable name on the lefthand side of an equation results in a temporary variable unless preceded by the reserved word FIELD.

The following example makes extensive use of temporary variables to place the name of the current month in a field named MonthName. The steps are:

1. Place the current date in the temporary variable date.
2. Extract from date and convert to text month, the number of the month.
3. Create a text list "nMonths" with the values 1 through 12.
4. Create a text list "months" with the values January through December.
5. Replace the number value of the current month with its name.

```
date := @Created;
month := @Text(@Month(date));
nMonths := ["1" : "2" : "3" : "4" : "5" : "6" : "7" : "8" : "9" : "10" : "11" : "12" ];
"October" : "November" : "December" ];
FIELD MonthName := @Replace(month; nMonths; months)
```

Variables can be assigned new values as a formula progresses.

Note: Reassignment of a variable is new with Release 6. In previous releases, an attempt to reassign a variable resulted in an error.

The following example reassigns the temporary variable n at each iteration of an @While loop.

```
n := 1;
@While(n <= @Elements(Categories);
    @Prompt([OK]; "Category " + @Text(n); Categories[n]);
    n := n + 1
)
Using constants

Formulas use three types of constants:

- **Text constants**
- **Numeric constants**
- **Time-date constants**

Lists can also be specified in a constant format.

**Text constants**

Specify a text constant by enclosing characters, including spaces, numbers, and special characters, in quotation marks (""" or braces ({})).

**Note:** Braces as text constant delimiters are new with Release 6.

Using braces as text delimiters enables you to enter the text constant "Hello World" as \( "Hello World" \) instead of \( "\"Hello World\"" \).

Do not enter variable names in quotation marks or braces, or the names are treated as text.

Enter numbers in quotation marks or braces when you want them treated as text.

To include multiple characters, for example, spaces, use \[@Repeat\]

The backslash ( \ ) serves as an escape character in a text constant. To embed quotation marks in a text constant delimited by quotation marks, precede each embedded quotation mark with a backslash. To embed a right brace in a text constant delimited by braces, precede each embedded right brace with a backslash. To embed a backslash in a text constant, type two backslashes.

A compiled formula does not distinguish between quotation marks and braces. When you open a design element containing formulas, quotation marks delimit all constants including those previously specified with braces. A backward slash prefixes a quotation mark previously specified in a constant delimited by braces.

**Examples: Text constants**

In the following formula, the Author field contains "Mary Chen."

1. If you enter the following as a field default value formula, the word Cost appears as the value in the field.
   "Cost"
2. The formula is:
   "From: " + Author + " (" + @Text(@Created) + ")"
   The result is:
   From: Mary Chen (11/30/2000 02:39:55 PM)
3. The formula is:
   "From:" + @Repeat(" ", 8) + Author + " (" + @Text(@Created) + ")"
   The result is:
   From: Mary Chen (11/30/2000 02:52:33 PM)
4. The formula is:
   "Type "Yes\" or \"No\"
   The result is:
   Type "Yes" or "No"
5. The formula is:
{Type "Yes" or "No"}
The result is (same as the preceding example):
Type "Yes" or "No"

6. The formula is:
"Type \Yes\ or \No\"
The result is:
Type \Yes\ or \No\n
**Numeric constants**

A numeric constant consists of numeric and special characters, with no intervening spaces, that conform to the following rules:
- Integer -- The characters 0 - 9 without intervening spaces comprise a positive integer.
- Decimal -- A decimal point may be placed before, after, or within the numeric characters.
- Sign -- The first character of a number may be a plus or minus sign.
- Scientific notation -- A number may be suffixed by the letter E, an optional plus (default) or minus sign, and an integer.

The table below shows the acceptable formats for entering numbers.

<table>
<thead>
<tr>
<th>Type of Number</th>
<th>Constant</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>.123</td>
<td>0.123</td>
</tr>
<tr>
<td>Integer</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>Negative number</td>
<td>-123.4</td>
<td>-123.4</td>
</tr>
<tr>
<td>Scientific notation</td>
<td>123E2, 123E-2</td>
<td>12300, 1.23</td>
</tr>
</tbody>
</table>

**Time-date constants**

A time-date constant consists of a time and/or a date enclosed in square brackets formatted as follows:
- 12-hour time -- A time in the format [hh:mm:ss] followed by the letters AM or PM. The hours can range from 00 to 12. The seconds are optional and default to 00.
- 24-hour time -- A time in the format [hh:mm:ss]. The hours can range from 00 to 23. The seconds are optional and default to 00.
- Date -- A date in the operating system’s default date format. Some examples include:

<table>
<thead>
<tr>
<th>Region</th>
<th>Default date format</th>
<th>Entering &quot;[01/02/03]&quot; results in</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>[mm/dd/yyyy]</td>
<td>January 2, 2003</td>
</tr>
<tr>
<td>France</td>
<td>[dd/mm/yyyy]</td>
<td>February 1, 2003</td>
</tr>
<tr>
<td>Japan</td>
<td>[yyyy/mm/dd]</td>
<td>February 3, 2001</td>
</tr>
</tbody>
</table>

The year (yyyy) is optional and defaults to the current year. If you use yy to specify a year, in the 20th century, yy is 50 or greater and in the 21st century, yy is less than 50. The validity of a date format depends on the date separator that users choose in their operating system control panel. The default separator for Windows, UNIX, and Macintosh is a slash (/). The default separator for OS/2 is a hyphen (–).
- Time and date -- A time and a date in the format [time date] or [date time].

If time-date values are subtracted, the result is an integer that represents the difference between the times in seconds.
The table below shows the formats you can use to specify a date in a Notes field, assuming the operating system’s default date format is US English and the year is 2002.

**Tip:** Display Time must be selected in the Date/Time field properties box for the time to display. It is not selected by default.

<table>
<thead>
<tr>
<th>Time-date format</th>
<th>Constant</th>
<th>Date/Time field result</th>
<th>Text field result</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-hour time</td>
<td>[5:30]</td>
<td>05:30 AM</td>
<td>05:30:00 AM</td>
</tr>
<tr>
<td>12-hour time</td>
<td>[5:30 PM]</td>
<td>05:30 PM</td>
<td>05:30:00 PM</td>
</tr>
<tr>
<td>24-hour time</td>
<td>[17:30]</td>
<td>05:30 PM</td>
<td>05:30:00 PM</td>
</tr>
<tr>
<td>Date</td>
<td>[6/15/02]</td>
<td>06/15/2002</td>
<td>06/15/2002</td>
</tr>
<tr>
<td>Time-date</td>
<td>[6/15 5:30 PM]</td>
<td>06/15/2002 05:30 PM</td>
<td>06/15/2002 05:30:00 PM</td>
</tr>
<tr>
<td>Time-date</td>
<td>[5:30 PM 6/15]</td>
<td>06/15/2002 05:30 PM</td>
<td>06/15/2002 05:30:00 PM</td>
</tr>
<tr>
<td>Difference</td>
<td>[5:30 PM]-[5:30]</td>
<td>43200.00</td>
<td>43200</td>
</tr>
</tbody>
</table>

**Specifying time zones**

To specify a time zone in a Time zone field, you can select a zone from a drop-down list of options or use the functions `@TimeZoneToText` and `@TextToTimeInZone` to manipulate the formats in which time values display programmatically.

Notes converts and saves human-readable time zone values using the following time zone attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>Time zone offset from GMT.</td>
</tr>
<tr>
<td>DO</td>
<td>Daylight Saving Time (DST) observed flag. 1 means DST is in effect; 0 means it is not. If equal to 1, a value should be supplied for DL.</td>
</tr>
<tr>
<td>DL</td>
<td>DST law identifying the <code>&lt;StartMonth&gt;</code> <code>&lt;StartDayOfWeek&gt;</code> <code>&lt;StartWeek&gt;</code> <code>&lt;EndMonth&gt;</code> <code>&lt;EndWeek</code> <code>&lt;EndDayOfWeek&gt;</code>.</td>
</tr>
<tr>
<td>ZX</td>
<td>(Optional) Host-specific time zone index.</td>
</tr>
<tr>
<td>ZN</td>
<td>(Optional) Time zone name.</td>
</tr>
</tbody>
</table>

For example, if you select "(GMT -05:00) Eastern Time (US & Canada)" from the Time zone drop-down list, Notes saves the value as:

Z=5$c00$=1$DL=4 1 1 10 -1 1$ZX=25$ZN=Eastern

This indicates that the time zone is 5 hours before Greenwich Mean Time (-5). DST is in effect and starts in the month of April(4), during week 1, on day 1 of the week, which is a Sunday, and ends in October(10), during the last week (-1), also on a Sunday(1). The ZX and ZN attributes are values used exclusively by Notes.

If you use a Time zone field as the default value for a view column, it displays as a similar string of attribute and value pairs. Use `@TimeZoneToText` to convert the field to a human-readable format.

You can enter these attribute and value pairs as the default value for a Time zone field, or you can let the `@TimeToTextInZone` function do the work for you. For example, to display a box specifying the current time in Alaska, you could create a Hotspot button with this formula:

```plaintext
@Prompt([Ok];"Current time in Alaska";@TimeToTextInZone(@Now;"Z=9$DO=1$DL=4 1 1 10 -1 1"))
```
or you could create a Time zone field called zone from which the user could select the zone for which they would like to see the current time, then display that time in a message box by adding this formula to the Hotspot button:

```
@Prompt([Ok];"Current time there");@TimeToTextInZone(@Now;zone)
```

## Using operators

Operators assign values, modify values, and combine existing values into new values. The following sections describe:

- **Operator overview and precedence**
- **Order of evaluation**
- **Assignment operator**
- **List concatenation operator**
- **List subscript operator**
- **Unary operators**
- **Arithmetic operators**
- **Text operator**
- **Comparison operators**
- **Logical operators**
- **Operations on lists**

### Operators and precedence

The table below lists the operators and their precedence, where 1 is the highest precedence.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>:=</td>
<td>Assignment</td>
<td>NA</td>
</tr>
<tr>
<td>[]</td>
<td>List subscript</td>
<td>1</td>
</tr>
<tr>
<td>:</td>
<td>List concatenation</td>
<td>2</td>
</tr>
<tr>
<td>+</td>
<td>Positive</td>
<td>3</td>
</tr>
<tr>
<td>-</td>
<td>Negative</td>
<td>3</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>4</td>
</tr>
<tr>
<td>**</td>
<td>Permuted multiplication</td>
<td>4</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>4</td>
</tr>
<tr>
<td>*/</td>
<td>Permuted division</td>
<td>4</td>
</tr>
<tr>
<td>+</td>
<td>Addition, concatenation</td>
<td>5</td>
</tr>
<tr>
<td>*+</td>
<td>Permuted addition</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>5</td>
</tr>
<tr>
<td>*/</td>
<td>Permuted subtraction</td>
<td>5</td>
</tr>
</tbody>
</table>
### Order of evaluation for operations

The values involved in an operation must be of the same data type. Operations occur in the following order:

- **Parenthesis** -- You can explicitly force the order of evaluation with parentheses. Operations within parentheses occur first. For example:
  
  \[
  (5 - 3) \times (6 - 4) = 4
  \]

- **Precedence** -- Operations not in parentheses occur in order of precedence starting with precedence 1. For example, multiplication has greater precedence than subtraction so \( 3 \times 6 \) occurs first:
  
  \[
  5 - 3 \times 6 - 4 = -17
  \]

- **Left to right** -- Operations of equal precedence occur left to right. For example:
  
  \[
  8 / 4 \times 2 = 4
  \]

### Assignment operator

The assignment operator \( (\text{:=}) \) assigns a value on the righthand side to a variable on the lefthand side. The variable assumes the type of the value on the righthand side.

This example assigns the numeric value 1 to the temporary variable \( n \).

\[
\text{n} := 1
\]

This example increments the temporary variable \( n \) by 1.

\[
\text{n} := \text{n} + 1
\]
This example assigns the text value "London" to the temporary variable city1.

city1 := "London"

The variable may be preceded by the reserved word `DEFAULT`, `ENVIRONMENT`, or `FIELD`. A variable not preceded by a reserved word is a temporary variable.

An assignment statement can be nested in an operation. The following example assigns "London" to the variable city1 as well as the value "LONDON" to city1Upper:

city1Upper := @UpperCase(city1 := "London")

This example, when used in a computed for display text field, displays the results of calculating the hotel, dinner, and nights editable number fields once a user enters their values and refreshes the document:

```plaintext
all := @Text(nights * each := hotel + dinner);
@If(hotel="","";"She spent " + @Text(each) + " per night and " + all + " in total on accommodations during the trip."

The `FIELD` reserved word can be used with a nested assignment. For example:

FIELD CityUpper := @UpperCase(FIELD City := "London")

The `DEFAULT` and `ENVIRONMENT` keywords cannot be used with a nested assignment statement.

Note: Nesting assignment statements is new with Release 6.

**List subscript operator**

The list subscript operator ([[)]) returns one element of a list.

**Note:** The list subscript operator is new with Release 6.

This example returns element 2 of the Categories field:

```
Categories[2]
```

A subscript consists of a numeric value in brackets. The numeric value can be a constant, variable, or expression. Decimals are rounded to integers. A subscript follows the list name.

The following example uses a variable subscript to iterate through a list.

```plaintext
n := 1;
@While(n <= @Elements(Categories));
  @Prompt([OK]; "Category " + @Text(n); Categories[n]);
  n := n + 1
@End
```

Note that the Categories field containing the list must be located above or to the left of the field containing this code or the formula returns an "Array index out of bounds" error.

The first element of a list is subscript [1]. A subscript that is less than [1] or that is greater than the number of elements in the list also returns the "Array index out of bounds" error.

The subscript operator is valid for any data type that allows lists (text, number, and date/time) even if the data entity is scalar. The subscript operator is only valid for data types that do not allow lists (richtext) when a subscript of [1] is used; this returns its current value unchanged.

The subscript operator cannot be used on the left side of an assignment statement. That is, you cannot assign a value to a subscripted element. You must build the complete list and then assign it. For example, if Categories is a 3-element list and you want to assign a new value to element 2:

List concatenation operator
The list concatenation operator (:) concatenates values into a list. The values must all be of the same type. This example is a three-member text list.
"London" : "New York" : "Tokyo"

The values can be constants, variables, and expressions, including other lists.
LNY := "London" : "New York";
LNY := "Tokyo"

Since list concatenation has the highest precedence next to subscripts, list elements that are expressions must be in parentheses. In the following example, the minus sign has the unintended effect of applying to both the third and fourth elements of the second list.
1:2:3:4 + 1:2:-3:4 = 2:4:0:8

Use parentheses to make the minus operation apply only to the third element of the second list.
1:2:3:4 + 1:2:(-3):4 = 2:4:0:8

Unary operators
The unary operators (+ and -) specify the sign of a numeric value. An unsigned numeric value is positive. The following numeric values are equivalent:
5
+5
-(-5)

Arithmetic operators
The arithmetic operators (* / + -) combine two numeric values using multiplication, division, addition, and subtraction. The following operations all result in the numeric value 16:
4 * 4
64 / 4
12 + 4
20 - 4

Text operator
The text concatenation operator (+) combines two text values. The following operation results in the value of the variable CompanyName followed by a comma, a space, and Inc.
CompanyName + ", Inc."

Comparison operators
The comparison operators (=, <>, !=, <, >, <=, and >=) compare values of the same type and produce a logical result (True or False). The following operations all result in a logical value of True:
"London" = "Lon" + "don"
"London" != "Tokyo"
2 + 2 > 3

Logical operators
The logical operators (!, &, and |) combine logical values. The following operations all result in a value of True. The operations are shown twice, with and without parentheses. The parentheses clarify the order of evaluation but are unnecessary because the logical operations are lower in precedence than the surrounding comparison operations.
4 = 2 + 2 & 5 = 3 + 2
(4 = 2 + 2) & (5 = 3 + 2)
4 = 2 + 2 | 5 = 2 + 2
(4 = 2 + 2) | (5 = 2 + 2)
! 5 = 2 + 2
! (5 = 2 + 2)

**Operations on lists**

Operations on lists are of two types:

- **Pair-wise** -- Pair-wise operators act on two lists in parallel–element fashion. The first element of list 1 pairs with the first element of list 2, the second element of list 1 pairs with the second element of list 2, and so on. If one list has fewer elements than the other, the last element in the shorter list is repeated for operations with the remaining elements of the longer list. If list 1 consists of "A";"B";"C" and list 2 consists of "1";"2", the operation is performed as though list 2 contained "1";"2";"2." For pair-wise equality tests, only one match is needed for the statement to return True, or 1.

- **Permuted** -- Permutation operators act on two lists, pairing every possible combination of their values. The resulting list has an element for each pairing in the following order: list 1 element 1 paired with each element in list 2, list 1 element 2 paired with each element in list 2, and so on through the last element in list 1.

If an operation occurs on a list and a non-list value, the non-list value is paired with each element in the list.

The table below shows the pair–wise and permutation operators.

<table>
<thead>
<tr>
<th>Pair-wise operator</th>
<th>Permutation operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>**</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>*/</td>
<td>Division</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>&gt;</td>
<td>=&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;=</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>=</td>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>!=</td>
<td>!=</td>
<td>Not equal</td>
</tr>
</tbody>
</table>

The table below shows how pair–wise and permutation operators differ.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Statement</th>
<th>Yields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation, pair-wise</td>
<td>&quot;A&quot;;&quot;B&quot;;&quot;C&quot;+1;&quot;2&quot;;&quot;3&quot;</td>
<td>&quot;A1&quot;;&quot;B2&quot;;&quot;C3&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;A&quot;;&quot;B&quot;;&quot;C&quot;+1;&quot;2&quot;</td>
<td>&quot;A1&quot;;&quot;B2&quot;;&quot;C2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;A&quot;;&quot;B&quot;;&quot;C&quot;+1</td>
<td>&quot;A1&quot;;&quot;B1&quot;;&quot;C1&quot;</td>
</tr>
<tr>
<td>Concatenation, permutation</td>
<td>&quot;A&quot;;&quot;B&quot;;&quot;C&quot;+1;&quot;2&quot;;&quot;3&quot;</td>
<td>&quot;A1&quot;;&quot;A2&quot;;&quot;A3&quot;;&quot;B1&quot;;&quot;B2&quot;;&quot;B3&quot;;&quot;C1&quot;;&quot;C2&quot;;&quot;C3&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;A&quot;;&quot;B&quot;;&quot;C&quot;+1;&quot;2&quot;</td>
<td>&quot;A1&quot;;&quot;A2&quot;;&quot;B1&quot;;&quot;B2&quot;;&quot;C1&quot;;&quot;C2&quot;</td>
</tr>
</tbody>
</table>

124  Programming Guide, Volume 1: Overview and Formula Language
<table>
<thead>
<tr>
<th>Operation</th>
<th>Statement</th>
<th>Yields</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1:2:3+10:20</td>
<td>11:22:23</td>
</tr>
<tr>
<td></td>
<td>1:2:3+10</td>
<td>11:12:13</td>
</tr>
<tr>
<td>Text equality, pair-wise</td>
<td>&quot;A&quot;:&quot;B&quot;:&quot;C&quot;=&quot;B&quot;:&quot;C&quot;:&quot;A&quot;</td>
<td>0 False</td>
</tr>
<tr>
<td></td>
<td>&quot;A&quot;:&quot;B&quot;:&quot;C&quot;=&quot;B&quot;:&quot;C&quot;</td>
<td>1 True</td>
</tr>
<tr>
<td></td>
<td>&quot;B&quot;:&quot;B&quot;:&quot;C&quot;=&quot;B&quot;:&quot;C&quot;</td>
<td>1</td>
</tr>
<tr>
<td>Text equality, permutation</td>
<td>&quot;A&quot;:&quot;B&quot;:&quot;C&quot;=&quot;B&quot;:&quot;C&quot;:&quot;A&quot;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&quot;A&quot;:&quot;B&quot;:&quot;C&quot;=&quot;B&quot;:&quot;C&quot;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&quot;B&quot;:&quot;B&quot;:&quot;C&quot;=&quot;D&quot;:&quot;E&quot;</td>
<td>0</td>
</tr>
<tr>
<td>Number equality, pair-wise</td>
<td>1:2:3=2:3:1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1:2:3=2:3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2:3:3=2:3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2:3:3=3:1</td>
<td>0</td>
</tr>
<tr>
<td>Number equality, permutation</td>
<td>1:2:3*=2:3:1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1:2:3*=2:3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1:2:3*=4:5</td>
<td>0</td>
</tr>
<tr>
<td>Date equality, pair-wise</td>
<td>[1–1–90]:[2–2–90]:[3–3–90]=</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[3–3–90]:[2–2–90]:[1–1–90]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[1–1–90]:[2–2–90]:[3–3–90]=</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[2–2–90]:[3–3–90]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[2–2–90]:[3–3–90]</td>
<td>1</td>
</tr>
<tr>
<td>Date equality, permutation</td>
<td>[1–1–90]:[2–2–90]:[3–3–90]=</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[3–3–90]:[2–2–90]:[1–1–90]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[1–1–90]:[2–2–90]:[3–3–90]=</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[2–2–90]:[3–3–90]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[4–4–90]:[5–5–90]</td>
<td>0</td>
</tr>
</tbody>
</table>
Using @functions

Notes @functions are built-in formulas that perform specialized calculations and return a value. The following sections describe:

- Syntax
- Return value
- Side-effects
- @Commands
- Order of evaluation

For the individual @function and @command descriptions, see the chapters Formula Language @Functions A-Z and Formula Language @Commands A-Z.

Syntax

The general format of an @function is:

@function-name(argument1 ; argument2 ; ... argumentn);

An @function consists of the name of the @function followed by arguments, if any. The first character of the name of an @function is always @.

Parenthesis

Enclose @function arguments in parentheses.

@Abs(-4)

Omit parentheses for @functions without arguments.

@Created

Multiple arguments

Separate multiple arguments with a semicolon.

@IsCategory("Yes"; "No")
@Middle(Company; 4; 4)

Keyword arguments

Enclose keyword arguments in square brackets. @Abstract, @Command, @PostedCommand, @DocMark, @GetPortsList, @PickList, @MailSend, @Name, and @Prompt use keyword parameters.

@Prompt([OK]; "Response"; Y)
@Name([CN]; AUTHOR)
@Command([[EditClear]])

You can also assign a keyword to a variable. For instance, the following assignment is valid:

o := [OK];
@Prompt(o;"Database title";@DbTitle)

Argument data types

Specify the correct data type for each @function argument per the @function description. For example, the first argument to @Prompt must be a keyword. If the keyword is [OK], it must be followed by two arguments of type text.

@Prompt([OK]; "The answer is ..."; @Text(N));

Return values

An @function calculates a return value and replaces itself with the value. The use of the @function must be appropriate for its data type. For example, @Power can compute the value of a numeric field:
But must be wrapped in @Text to compute the value of a text field:
@Text(@Power(2; Exp))

**Side-effects**
A side-effect is an action that occurs outside the immediate scope of the formula. For example, @Prompt displays a dialog box in addition to returning a value. Make sure that a side-effect occurs at the correct point of a formula’s execution.

The following @functions have side-effects:

<table>
<thead>
<tr>
<th>@Function</th>
<th>Side-effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Command @PostedCommand</td>
<td>A Notes command, such as opening a database, is performed.</td>
</tr>
<tr>
<td>@DbColumn @DbCommand @DbLookup</td>
<td>Another view or database is accessed, and data is retrieved.</td>
</tr>
<tr>
<td>@DDEInitiate @DDEExecute</td>
<td>A DDE conversation is initiated (or terminated), or a DDE statement is executed.</td>
</tr>
<tr>
<td>@MailSend</td>
<td>A Notes mail memo is created and routed to another user or database.</td>
</tr>
<tr>
<td>@Prompt @PickList @DialogBox</td>
<td>A dialog box is displayed; data may be returned.</td>
</tr>
</tbody>
</table>

**@Commands**
The @Command and @PostedCommand functions execute a Notes command. The first argument to @Command or @PostedCommand is a keyword argument that specifies the Notes command. Depending on the Notes command, other arguments may be required.

You must use @PostedCommand in applications that run in Notes R3 and R4. The difference between @Command and @PostedCommand is the order of evaluation.

Because of their large number and special status, these @functions comprise a separate category called @commands. Each @command is named after the first argument to @Command or @PostedCommand, which is a keyword argument.

Most @commands mimic a menu command. For example:
@Command([AddDatabase]; "Legal1":"Trademrk.nsf")
@Command([AdminRegisterUser])
@PostedCommand([DesignForms])
@PostedCommand([EditDown]; "5")

You must be careful with @commands due to their side-effects and their order of evaluation.

See "Order of evaluation for formula statements" and "Side-effects" in this chapter for additional information.

You can use @commands in formulas for tools, events, button hotspots, action hotspots, and actions. You can use @commands in agents that have no target documents. See the individual @command descriptions for further restrictions.

Setting the NoExternalApps environment variable to 1 disables any formula containing an @command function. The user does not receive an error message -- the formula simply does not execute.
Order of evaluation for formula statements
Lotus Domino evaluates formulas from top to bottom and left to right, completing each statement before proceeding to the next, except that @PostedCommand and a few @Command functions are executed in order after all other @functions complete execution. Formula language contains several @functions for control logic.

Except for @commands, the formula language operates on back-end Notes objects. For example, a field named in a formula refers to the field as it exists in storage and the FIELD reserved word modifies a stored field. @Commands operate in the user interface; changes made there are not reflected in the back-end until a document is saved. You cannot intersperse back-end and user interface accesses of the same document and get correct results.

Evaluation of @commands
Execution of an @PostedCommand function occurs after all other @functions in the formula. For instance, look at the following formula:

@PostedCommand([CommandName]; Argument);
@If(Condition; TrueStatement; FalseStatement);
FIELD X := "Text"

The first statement is executed last.

Execution of an @command function occurs in the order it appears with some exceptions. The exceptions, like @PostedCommands, execute at the end of the formula. Each exception has an equivalent @command that executes immediately.

The following table lists the @commands that execute last and their corresponding @commands that execute immediately.

Note: The functions that get evaluated immediately are new with Release 6.

<table>
<thead>
<tr>
<th>Evaluated after @functions</th>
<th>Evaluated immediately</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditClear</td>
<td>Clear</td>
</tr>
<tr>
<td>EditProfile</td>
<td>EditProfileDocument</td>
</tr>
<tr>
<td>FileCloseWindow</td>
<td>CloseWindow</td>
</tr>
<tr>
<td>FileDatabaseDelete</td>
<td>DatabaseDelete</td>
</tr>
<tr>
<td>FileExit</td>
<td>ExitNotes</td>
</tr>
<tr>
<td>Folder</td>
<td>FolderDocuments</td>
</tr>
<tr>
<td>NavigateNextMain</td>
<td>NavNextMain</td>
</tr>
<tr>
<td>NavigateNextSelected</td>
<td>NavNextSelected</td>
</tr>
<tr>
<td>NavigateNextUnread</td>
<td>NavNextUnread</td>
</tr>
<tr>
<td>NavigatePrevMain</td>
<td>NavPrevMain</td>
</tr>
<tr>
<td>NavigatePrevSelected</td>
<td>NavPrevSelected</td>
</tr>
<tr>
<td>NavigatePrevUnread</td>
<td>NavPrevUnread</td>
</tr>
<tr>
<td>ReloadWindow</td>
<td>RefreshWindow</td>
</tr>
<tr>
<td>ToolsRunBackgroundMacros</td>
<td>RunScheduledAgents</td>
</tr>
<tr>
<td>ToolsRunMacro</td>
<td>RunAgent</td>
</tr>
<tr>
<td>ViewChange</td>
<td>SwitchView</td>
</tr>
</tbody>
</table>
Evaluated after @functions | Evaluated immediately
---|---
ViewSwitchForm | SwitchForm

@If function
@If executes one statement or another depending on whether a logical value is True or False:
@If(LogicalValue; TrueStatement; FalseStatement)

@Do function
@Do executes a number of statements in sequence and can be used as an execution path within an @If function:
@If(LogicalValue; @Do(TrueStatement1; TrueStatement2); FalseStatement)

Any @Command functions within an @Do function are executed in order after all other @functions, both within and without the @Do function, are executed.

@Return function
@Return stops execution of the formula:
@If(LogicalValue; @Return(""); ")

Using reserved words
Formula language includes a set of reserved words that perform special functions:

<table>
<thead>
<tr>
<th>Reserved word syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT fieldName := value</td>
<td>Associates a value with a field. If the field exists in the document being processed, its current value is used. If the field does not exist, the document is treated as if the field does exist and the DEFAULT value is used.</td>
</tr>
<tr>
<td>ENVIRONMENT variable := textValue</td>
<td>Assigns a value to an environment variable in the user’s notes.ini file (Windows, OS/2, UNIX) or Notes Preferences file (Macintosh).</td>
</tr>
<tr>
<td>FIELD fieldName := value</td>
<td>Assigns a value to a field in the current document. If the field does not exist, it is created; if it already exists, the contents are replaced.</td>
</tr>
<tr>
<td>REM [&quot;remarks&quot;]</td>
<td>Inserts documentation into the formula without affecting its function.</td>
</tr>
<tr>
<td>REM [[remarks]]</td>
<td></td>
</tr>
<tr>
<td>SELECT logicalValue</td>
<td>Specifies whether or not the current document is valid for processing in view selection, replication, and agent formulas.</td>
</tr>
</tbody>
</table>

A reserved word is always the first word in a statement. By convention, reserved words are entered in uppercase. You can enter them in lowercase, but Lotus Domino converts them to uppercase when saving a formula.

The fieldName and variable specifications to reserved words are names, not text constants. Do not enclose them in parentheses.

Examples: Using keywords
1. (DEFAULT) -- If the field KeyThought exists, whatever value is in that field is used for the computed field. If the field KeyThought does not exist, the value of Topic is used.
   DEFAULT KeyThought := Topic;
   KeyThought
   This formula is equivalent:
   @If(@IsAvailable(KeyThought); KeyThought; Topic)
2. (ENVIRONMENT) -- Converts a number to text and saves it as an environment variable.

   ENVIRONMENT OrderNumber := @Text(NewOrderNumber)

3. 3 (FIELD) -- This formula adds "Inc." to the value of the Company field.

   @If(@Matches(LowerCase(Company); "*, inc*"); @Return(" "); ");
   FIELD Company := Company + ", Inc.

4. (FIELD) -- This formula creates a new field called CompanyName to hold the name of the company
   plus "Inc." The field does not become visible unless you add it to the form design, but you can
   access it by naming it in formulas.

   FIELD CompanyName := Company + ", Inc.

5. (FIELD) -- This formula deletes the field CompanyName.
   FIELD CompanyName := @DeleteField;

6. (REM) -- This formula contains five lines of comments before the code.

   REM "6/15/95"
   REM "The following formula calculates the date"
   REM "for the DueDate field"
   REM "DueDate is the Date field + thirty days"
   REM;
   @Adjust(Date; 0; 0; 30; 0; 0)

7. (REM) -- This formula contains five lines of comments before the code.

   REM (1/15/01);
   REM (The following formula calculates the date);
   REM (for the "DueDate" field);
   REM ("DueDate" is the Date field + thirty days);
   REM;
   @Adjust(Date; 0; 0; 30; 0; 0)

8. (SELECT) -- This formula selects all documents in the database.

   SELECT @All

9. (SELECT) -- This view selection formula selects only documents composed from the Product
   Specification form or that are response documents.

   SELECT Form="Product Specification" | @IsResponseDoc

10. (SELECT) -- This example changes the value of the Status field to Closed except for documents
    whose Categories Field is "Unsigned Contracts."

    SELECT Categories != "Unsigned Contracts";
    FIELD Status := "Closed"

---

**Specifying form and view names in formulas**

When you specify a form or view name in a formula:

- Do not include the accelerator character, an underscore, in the name if it exists. The formula language
  treats an underscore as a literal underscore.
- Do include the cascade character, a backslash, but remember that it requires the escape character so
  must be entered as two backslashes.

The following formula demonstrates how to enter the view name "_Marketing\_Procedures":

```
@Command([SwitchView]; "Marketing\Procedures")
```

---

**Debugging formulas**

The formula language does not provide a formal debugging mechanism. You can use @Prompt to stop at
 certain points and to examine variables. This example uses @Prompt to set a checkpoint and then to
 examine a variable. After you establish that your code is running correctly, remove the debug statements.

```
@Prompt([OK]; "Checkpoint"; "About to calculate LastName");
LastName := @RightBack(@Left(@UserName; "/"); ");
@Prompt([OK]; "Value of LastName"; LastName)
```
In some cases, such as agents that do not run on the current document and selection formulas, @Prompt does not work. You do not receive a caution when you write or run the formula -- the statement simply does not run. To work around this problem, you can write the formula in an action, toolbar button, or some other object in which @Prompt works. After testing the formula there, remove the debug statements and copy and paste the formula into the desired object.

Alternatively, you can design debug fields into the form for the documents your formula processes. Your formula loads the debug fields during testing and you open the documents processed to examine the fields. After testing, remove the debug fields from the form as well as the debug statements from the formula.

FIELD DebugLastName := lastName
Chapter 5. Formula Language Coding Guidelines

Formulas are expressions that have program-like attributes. For example, you can assign values to variables and use a limited control logic. The formula language interface to Lotus Domino is through calls to @functions, @Commands, a subset of the @functions, provide access to the user interface.

This section provides these topics:
- Formulas
- Writing messages and getting user input
- Handling errors
- Working with @functions
- Working with @commands
- Performing string operations
- Performing arithmetic operations
- Performing time-date operations
- Accessing the user environment
- Accessing the current database and view
- Accessing the current document in the formula language
- Accessing data outside the current document and database
- Accessing external databases through LS:DO using @functions

See "Formula Language @Functions A-Z" and "Formula Language @Commands A-Z" for a reference.

Formulas

A formula consists of one or more statements that are executed in order. Depending on the object associated with the formula and other criteria, the formula may execute once or it may execute multiple times on selected documents (one execution per document).

Formulas do have language elements for loop iteration.

Agent formulas execute multiple times on selected documents, giving the effect of conditional, iterative execution. Data can be processed in lists, giving the same effect.

You can:
- Write formulas that evaluate to a result
- Write formulas that perform actions
- Work with lists
- Use conditional statements
- Use iterative statements

Writing formulas that evaluate to a result

The final statement of the following formulas must evaluate to a result:
- Replication formula -- Must evaluate to True (1) or False (0), and is applied to each document in the database.
- Form formula -- Must evaluate to the name of a form.
- Selection formula -- Must evaluate to True (1) or False (0), and is applied to each document in the view.
• Column formula -- Must evaluate to a value that can be converted to a text string.
• Hide action formula -- Must evaluate to True (1) or False (0).
• Formula pop-up -- Must evaluate to a text string.
• Window title formula -- Must evaluate to a text or numeric value, except that the formula can consist of a single field of any type.
• Section access formula -- Must evaluate to a name or list of names.
• Insert subform formula -- Must evaluate to a text value that is the name of a subform.
• Section title formula -- Must evaluate to a text or numeric value, except that the formula can consist of a single field of any type.
• Hidden paragraph formula -- Must evaluate to True (1) or False (0).
• Default value formula -- Must evaluate to a value suitable for storage in the current field.
• Input translation formula -- Must evaluate to a value suitable for storage in the current field.
• Default validation formula -- Must evaluate to success (1) or failure (0).
• Computed field formula -- Must evaluate to a value suitable for storage in the current field.
• Keyword field formula -- Must evaluate to a value or list of values suitable for storage in the current field.

These formulas may be as simple as a single field, constant, or @function, or may contain multiple statements, use temporary variables, modify fields, and produce side-effects. In all cases, however, their final statement must be a value suitable for the result.

The reserved word SELECT is prepended to the logical statement that terminates a replication or selection formula if SELECT is not explicitly specified. These formulas run against every document in the database (replication formula) or view (selection formula) to determine their inclusion or exclusion in the replication process or the view. The @All function returns the value True, so the formula "SELECT @All" includes all documents.

Examples: Writing formulas that evaluate to a result

1. This example is the default value formula for the From field in a standard discussion database. It consists of a single @function that returns the user's name.
   
   @UserName

2. This example is an adaptation of the input validation formula for the Subject field in a standard discussion database. It consists of an @If function that returns failure if the Subject field is blank (the user did not enter a value) and success if the Subject field contains something. The failure condition has the side-effect of printing a message.
   
   @If(Subject = ""; @Failure("You must enter a subject for your document."); @Success)

3. This example is an adaptation of the window title formula for the Memo form in a standard discussion database. It is a single statement, but it contains embedded @If commands. If the current document is new, the window title becomes "New Memo." If the current document already existed and the Subject field exists and is not blank, the window title becomes the value of the Subject field, prepended by ">> " if it has attachments. If the Subject field does not exist or is blank, the window title becomes the creation date.
   
   @If(@isNewDoc; "New Memo"; @If(@IsAvailable(Subject) & Subject != ""; @If(@Attachments; "x "; ") + Subject; @Text(@Created)))

Writing formulas that perform actions

The following formulas do not evaluate to a final, usable result, but depend on their field assignments and side-effects to function:

• Agent formula -- Executes on a database when triggered. An agent formula executes once on each selected document, as determined by criteria specified in the Agent Properties box and a SELECT reserved word included in the formula. The SELECT reserved word defaults to SELECT @All.
• Action formula -- Executes on a view or form when triggered.
• Button formula -- Executes on a form, navigator, or rich text field when triggered.
• Action hotspot -- Executes on a form, navigator, or rich text field when triggered.

Examples: Writing formulas that perform actions
This agent example substitutes "Wayside Drive" for "Wayside Street" in the Address field of documents based on the Main Form. The effective action is the FIELD Address assignment. If Address contains "Wayside Street," the new Address consists of the characters to the left of "Wayside Street," the string "Wayside Drive," and the characters to the right of "Wayside Street." Otherwise, Address is reset to its current value.

```sql
SELECT Form = "Main Form";
ws := "Wayside Street";
wd := "Wayside Drive";
FIELD Address := @If(@Contains(Address; ws); @Left(Address; ws) + wd +
@Right(Address; ws); Address)
```

Working with lists
A list is a named entity that can contain multiple values of the same type. Lists occur as follows:
• A field that allows multiple values may contain a list rather than a scalar value.
• Some @functions return a list.
• Constants and variables can be specified as lists as well as scalar values. The syntax is multiple elements separated by colons; for example, "London" : "New York" : "Tokyo" is a string list constant of three elements.
• Since list concatenation has the highest precedence, list elements that are expressions must be in parentheses if the expression applies only to that element. For example, write 1:2:(-3):4, not 1:2~3:4, if 3 is negative and 4 is not.
• Lists can be subscripted to read (but not write) elements. For example, Categories[2] is the second element of the Categories field.
• Operators combine lists on a pair-wise or permuted basis.
   For more information, see Operations on lists in "Formula Language Rules."

Lists provide limited iteration because a list operation applies the operation to each element of the list, like processing an array in a loop.

The following @functions deal specifically with lists.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Compare</td>
<td>Compares two lists pair-wise.</td>
</tr>
<tr>
<td>@Count(list)</td>
<td>Determines the number of elements in a list, returning 1 if the value it is evaluating is a null string or not a list.</td>
</tr>
<tr>
<td>@Elements(list)</td>
<td>Determines the number of elements in a list, returning 0 if the value it is evaluating is a null string or not a list.</td>
</tr>
<tr>
<td>@Explode(string)</td>
<td>Converts a text string into a text list. Spaces, commas, semicolons, and newlines separate elements in the string.</td>
</tr>
<tr>
<td>@Explode(string ; separators)</td>
<td>Converts a text string into a text list; the second parameter specifies the separators (except newlines) for elements in the string.</td>
</tr>
<tr>
<td>@Explode(string ; separators ; includeEmpty)</td>
<td>Converts a text string into a text list; @True as the third parameter includes empty list elements where consecutive separators occur.</td>
</tr>
<tr>
<td>@Explode(string ; separators ; includeEmpty ; newlineAsSeparator)</td>
<td>Converts a text string into a text list; @False as the fourth parameter excludes newlines as separators.</td>
</tr>
</tbody>
</table>
### Function Table

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Explode(dateRange)</td>
<td>Converts a date range into a list of dates. The argument must be a time-date value; the return value is a text list.</td>
</tr>
<tr>
<td>@Implode(list)</td>
<td>Converts a text list to a text string, using spaces to separate elements.</td>
</tr>
<tr>
<td>@Implode(list; separator)</td>
<td>As above, but the second parameter specifies the separator for elements in the string.</td>
</tr>
<tr>
<td>@IsMember(string; list)</td>
<td>Determines if a string is a member of a list. Returns True (1) or False (0).</td>
</tr>
<tr>
<td>@IsMember(list1; list2)</td>
<td>Determines if a list is contained in another list. Returns True (1) or False (0).</td>
</tr>
<tr>
<td>@IsNotMember(string; list)</td>
<td>Determines if a string is not a member of a list. Returns True (1) or False (0).</td>
</tr>
<tr>
<td>@IsNotMember(list1; list2)</td>
<td>Determines if a list is not contained in another list. Returns True (1) or False (0).</td>
</tr>
<tr>
<td>@Keywords(list1; list2)</td>
<td>Locates words in list1 that match words in list2. Word separators are &quot;,&quot; , ? ! ; : [ ]</td>
</tr>
<tr>
<td>@Keywords(list1; list2; separator)</td>
<td>As above, but the second parameter specifies the word separators.</td>
</tr>
<tr>
<td>@Max(number or numberlist)</td>
<td>Returns the largest number in the list.</td>
</tr>
<tr>
<td>@Max(number or numberlist; number or numberlist)</td>
<td>Returns the larger number of two numbers or a number list of the largest numbers resulting from a pair-wise computation of two number lists.</td>
</tr>
<tr>
<td>@Member(value; list)</td>
<td>Determines the position of a value in a string list.</td>
</tr>
<tr>
<td>@Min(number or numberlist)</td>
<td>Returns the smallest number in the list.</td>
</tr>
<tr>
<td>@Min(number or numberlist; number or numberlist)</td>
<td>Returns the smaller number of two numbers or a number list of the smallest numbers resulting from a pair-wise computation of two number lists.</td>
</tr>
<tr>
<td>@Nothing</td>
<td>Adds nothing to a transformed list.</td>
</tr>
<tr>
<td>@Replace(list1; list2; list3)</td>
<td>Replaces values in list1 that match values in list2 with the corresponding values in list3.</td>
</tr>
<tr>
<td>@Sort(list [order])</td>
<td>Sorts a list. Order is [Ascending] (default) or [Descending].</td>
</tr>
<tr>
<td>@Subset(list; n)</td>
<td>Extracts n number of values from the list. Use -n to extract right to left.</td>
</tr>
<tr>
<td>@Transform(list; name; formula)</td>
<td>Applies a formula to each element of a list.</td>
</tr>
<tr>
<td>@Unique(list)</td>
<td>Removes duplicate values from a string list.</td>
</tr>
<tr>
<td>@Unique</td>
<td>Returns a random, unique text value.</td>
</tr>
</tbody>
</table>

### Examples: Working with lists

1. (Subscript). @DbName returns a list where element 1 is the server name and element 2 is the file name of the current database. This example gets the file name.

   @Prompt([Ok]; "Database name";
   ('') + @DbName[2] + {'")

2. (Pair-wise operation). This example adds two numeric lists in a pair-wise operation. The resulting list has the four values 11, 22, 27, and 44.
list1 := 10 : 20 : 30 : 40;
list2 := 1 : 2 : (-3) : 4;
list3 := list1 + list2;
result := @Text(list1) + "+" + @Text(list2) + "=" + @Text(list3);
@Prompt([OkCancelList]; "Result"; ";"; ";"; result)

3. (Permutated operation). This example concatenates two lists in a permutated operation. The resulting list has the 12 values Blue Sedan, Blue Coupe, Blue Van, Blue Truck, Red Sedan, and so on through Yellow Truck.
cars := "Sedan": "Coupe": "Van": "Truck";
colors := "Blue": "Red": "Yellow";
result := colors + "": cars;
@Prompt([OkCancelList]; "Result"; ";"; ";"; result)

4. (@Elements). This example prints a message if the Categories field has no elements, and prints the list if it does.
@If(@Elements(Categories) = 0; @Prompt([Ok]; "Categories": "No categories");
@Prompt([OkCancelList]; "Categories": ";"; ";"; Categories))

5. (@Explode). This example explodes a string constant into a list using the default separators space, comma, and semicolon. The resulting list has the values Paris, London, Chicago, and Seoul.
cityString := @Explode("Paris,London,Chicago;Seoul");
@Prompt([OkCancelList]; "List of cities"; ";"; ";"; cityString)

6. (@Explode). This example explodes a string constant into a list using the separators comma and semicolon. The resulting list has the values Paris, London, New York, and Washington DC. New York and Washington DC are not separated into New, York, Washington, and DC because the space is not included as a separator.
cityList := @Explode("Paris,London,New York;Washington DC": ";";");
@Prompt([OkCancelList]; "List of cities"; ";"; ";"; cityList)

7. (@Explode). This example includes a blank entry between London and New York. If the third parameter is @False or is omitted, multiple consecutive separators are treated as one. Ensure that the commas are consecutive and not separated by a space.
cityList := @Explode("Paris,London, , New York;Washington DC": ";";
@True); @Prompt([OkCancelList]; "List of cities"; ";"; ";"; cityList)

8. (@Implode). This example implodes a list constant into a string variable using a space (the default) to separate the values. The resulting string has the value Minneapolis Detroit Chicago.
city := "Minneapolis": "Detroit": "Chicago";
cityString := @Implode(city);
@Prompt([Ok]; "Imploded string": cityString)

9. (@Implode). This example implodes a list constant into a string variable using a comma and a space to separate the values. The resulting string has the value Minneapolis, Detroit, Chicago.
city := "Minneapolis": "Detroit": "Chicago";
cityString := @Implode(city; ",");
@Prompt([Ok]; "Imploded string": cityString)

10. (@Implode). This example implodes a list field into a string using a colon to separate the values. If the Categories field is entered as Minneapolis, Detroit, Chicago, the resulting string is Minneapolis:Detroit:Chicago.
@Prompt([Ok]; "Categories": @Implode(Categories; ";"))

11. (IsMember). This agent example checks the selected documents to see if Adjusted is in the Categories list. If it is, Categories remains the same. If it is not, Adjusted is added to the Categories list.
FIELD Categories := @If(@IsMember("Adjusted": Categories); Categories;
@Explode(@Implode(Categories; ";"); ";Adjusted": ";"));
SELECT @All

12. (@IsNotMember). This example checks the selected documents to see if Adjusted and Signed off are in the Categories list. If both are not, both are added to the Categories list. If both are, Categories remains the same.
FIELD Categories := @If(@IsNotMember("Adjusted" : "Signed off"; Categories);
@Explode(@Implode(categories; ";") + ";Adjusted;Signed off"; ";") ; Categories);
SELECT @All

13. (@Keywords). This example finds which keywords are used in the Cities field.
   keywords := @Keywords(cities; "Paris" : "Moscow" : "Tokyo" : "Boston");
   @Prompt([Ok] ; ";Keywords" ; keywords)

14. (@Member). This example has the user pick a value from a list and displays the number of that
    value within the list.
   cars := "Sedan" : "Coupe" : "Van" : "Truck";
   car := @Prompt([OkCancelList] : [NoSort] ; "Cars" : "Pick one" ; "Sedan" ; cars);
   n := @Member(car ; cars);
   @Prompt([Ok] ; "Your selection is ..." ; "Number " + @Text(n))

15. (@Replace). This example replaces red with scarlet and blue with turquoise in the list derived from
    colors.
   colors := "red" : "blue" : "yellow" : "blue" : "black" : "red";
   from := "red" : "blue";
   to := "scarlet" : "turquoise";
   result := @Replace(colors ; from ; to);
   @Prompt([OkCancelList] : [NoSort] ; "Replacement list" ; "" ; ";" ; result)

16. (@Subset). This example places New Orleans, London, and Frankfurt into first3, and Singapore and
    Sydney into last2.
   first3 := @Subset(cities ; 3);
   last2 := @Subset(cities ; -2);
   @Prompt([OkCancelList] : [NoSort] ; "First three" ; "" ; "" ; first3);
   @Prompt([OkCancelList] : [NoSort] ; "Last two" ; "" ; "" ; last2)

17. (@Transform and @Nothing). This example returns a list with three elements: 1, 2, and 4.
   OriginalList := 1 : 4 : -4 : 16;
   @If(OriginalList = @Nothing ; @Nothing;
       @Transform(OriginalList ; "x" ;
         @If(x >= 0 ; @Sqrt(x) ; @Nothing)))

18. (@Unique). This example returns a list with four elements: red, blue, yellow, and black.
   colors := "red" : "blue" : "yellow" : "blue" : "black" : "red";
   result := @Unique(colors);
   @Prompt([OkCancelList] : [NoSort] ; "Unique list" ; "" ; ";" ; result)

Using conditional statements
@If lets you execute one statement or another, depending on whether a condition is True or False. A
condition is typically the comparison of values, but can also be a constant, a variable, or the result of an
@function. For example:

- @ViewTitle = "By Author" is True if the name of the current view is "By Author."
- @Elements(categories) > 0 is True if Categories has at least one element.
- 1 used as a condition means True. @True and @Yes return 1. Comparisons and @functions that evaluate
to a condition return 1 if the condition is True.
- 0 used as a condition means False. @False and @No return 0. Comparisons and @functions that
evaluate to a condition return 0 if the condition is False.

The @If statement has an odd number of parameters with a minimum of three, as follows:

- The condition is the first parameter and every other parameter thereafter if the @If statement has
  multiple conditions.
- The statement that is executed if the condition is True is the second parameter and every other
  parameter thereafter if the @If statement has multiple conditions.
- The statement that is executed if the condition is False is the last parameter.
The simplest @If statement has the following syntax:

@If(condition; True statement; False statement)

An @If statement with three conditions has the following syntax:

@If(condition1; True1; condition2; True2; condition3; True3; False)

The @If function is evaluated left to right, and the first condition that is True causes its corresponding True statement (that is, the next parameter) to be processed. No further evaluation or processing within the @If statement takes place. If none of the conditions are True, the False statement (that is, the last parameter) is processed.

The True and False statements take various forms, depending on their context:

- If the @If statement is the last statement in a formula that evaluates to a result, the True and False statements must evaluate to a result.
- If the @If statement is the righthand side of an assignment, the True and False statements must evaluate to a value suitable to assignment to the field or temporary variable on the left side.
- Otherwise, the True and False statements must cause an action.

Assignments for True and False statements take two forms:

- The "classic" form assigns the result of the entire @If statement where each True and False statement consists only of a value:
  
  \[ \text{variable} := \text{@If(condition; value1; value2)} \]

- Alternatively you can make each True and False statement a complete assignment:
  
  \[ \text{@If(condition; variable := value1; variable := value2)} \]

  **Note:** The second form is new with Release 6.

The parameters of @If can themselves be @If statements. Nested @If statements are useful to work around the limited logic constructs in the formula language, but make for complicated syntax.

The @Do function provides a means to execute multiple statements on a True or False condition.

For more information on conditional operators, see "Using operators" in "Formula Language Rules."

**Examples: Using conditional statements**

1. This agent example compares the number of elements in Categories to 0. If the field has elements, it is not changed (it is set to itself). If the field does not have elements, it is set to a constant string.
   
   FIELD Categories := @If(@Elements(Categories) > 0; Categories; "To be supplied ...")
   
   SELECT @All

2. This window title example first checks the return value of @IsNewDoc. If the document is new, the window title is set to the text constant "New Topic." If the document already exists, the window title is set to the Subject field if the current view is AuthorView. Otherwise, the window title is set to the Subject field followed by a string denoting the number of response documents.
   
   StandardTitle := Subject + @DocDescendants(" (No Responses)"; " (1 Response)"; " (% Responses)");
   
   @If(@IsNewDoc; "New Topic"; @ViewTitle = "AuthorView"; Subject; StandardTitle)

**Using iterative statements**

@DoWhile, @While, and @For let you execute statements iteratively (in a loop), depending on whether a condition is True or False. @For initializes, changes, and tests the condition as part of the @function, and is best used processing a range of numbers such as list subscripts. @While and @DoWhile test the
condition; typically you initialize the condition before the @While or @DoWhile statement, and change the condition with one of the @While or @DoWhile statements. @While tests the condition before executing its statements, and @DoWhile tests after.

**Note:** @While, @DoWhile, and @For are new with Release 6.

**Examples: Using iterative statements**
The three agents that follow all display the elements of the Categories field one at a time.

1. This agent uses an @For loop. The first parameter initializes the variable n to 1 and executes once. The second parameter tests whether n is less than or equal to the number of elements in Categories. The third parameter increments n. The fourth parameter is a statement that executes as long as the test remains True.
   ```
   For(n := 1;
   n <= @Elements(Categories);
   n := n + 1;
   @Prompt([Ok]; "Category " + @Text(n); Categories[n])
   ```

2. This agent uses an @While loop.
   ```
   n := 1;
   @While(n <= @Elements(Categories);
   @Prompt([Ok]; "Category " + @Text(n); Categories[n]);
   n := n + 1)
   ```

3. This agent uses an @DoWhile loop.
   ```
   @If(@Elements(Categories) = 0; @Return(0); "");
   n := 1;
   @DoWhile(
   @Prompt([Ok]; "Category " + @Text(n); Categories[n]);
   n := n + 1;
   n <= @Elements(Categories))
   ``
Use \texttt{@StatusBar} to write messages that will display in the status bar to the user. By writing messages to the status bar you can keep users informed of the internal processing being performed by an application.

\textbf{Examples: Writing messages to the user}

1. This example writes a text constant as the title and a text expression in a dialog box. The expression is a concatenation of text constants and time-date values converted to text.
   \begin{verbatim}
   @Prompt([Ok]; "Current time and date"; "The date is " + @Text(@Now; "DOS0") + ", " + 
   "The time is " + @Text(@Now; "TOS1") + ".")
   \end{verbatim}

2. This example writes a text constant as the title of a dialog box. The content of the box is a text constant followed by the value of a multi-value field.
   \begin{verbatim}
   @Prompt([OkCancelList] : [NoSort]; "Field offices"; "Current field offices are located in the following cities:"; ""; @DbColumn("Field_offices")
   \end{verbatim}

3. This code, when added to the postsave event for a form, writes, “Saving with a modification date of 1/1/2001 01:38:11 PM” to the status bar, if the last modification date is January 1 and the "date" field contains the function \texttt{@Modified}:
   \begin{verbatim}
   @StatusBar("Saving with a modification date of " + date)
   \end{verbatim}

4. This code, when added to the postopen event for a form, writes, "You are creating a self-evaluation for Maria Rose” to the status bar at the bottom of the form when Maria Rose opens it in the Notes client:
   \begin{verbatim}
   @StatusBar("You are creating a self-evaluation for " + @Name([CN]; @V3UserName))
   \end{verbatim}

\textbf{Getting user input with \texttt{@Prompt} and \texttt{@PickList}}

Use the following forms of \texttt{@Prompt} and \texttt{@PickList} to get input from an interactive user:

- \texttt{@Prompt([YesNo]; title; prompt)} displays a dialog box with title text, prompt text, and Yes and No buttons. \texttt{@Prompt} returns True (1) if the user clicks Yes and False (0) if the user clicks No.
- \texttt{@Prompt([YesNoCancel]; title; prompt)} displays a dialog box with title text, prompt text, and Yes, No, and Cancel buttons. \texttt{@Prompt} returns True (1) if the user clicks Yes, False (0) if the user clicks No, and -1 if the user clicks Cancel.
- \texttt{@Prompt([OkCancelEdit]; title; prompt, default; choices)} displays a dialog box with title text, prompt text, and a box in which the user can type. \texttt{@Prompt} returns what the user types as a text value. If the user clicks Cancel, the formula terminates immediately.
- \texttt{@Prompt([Password]; title; prompt)} is as above, but echoes Xs instead of what the user types in.
- \texttt{@Prompt([OkCancelList] : [NoSort]; title; prompt; default; choices)} displays a box with the title text at the top of the box, the prompt text in the body of the box, and the choices text list below the prompt text, with the default highlighted. The last parameter must be a text list. \texttt{@Prompt} returns the list element, the user selects. If the user clicks Cancel, the formula terminates immediately. Do not specify \texttt{[NoSort]} if you want the list sorted. \texttt{@DbColumn} can be used to generate lists based on the current contents of views in specified databases.
- \texttt{@Prompt([OkCancelCombo]; title; prompt; default; choices)} is as above, but uses a drop-down list with the default in the box above the list.
- \texttt{@Prompt([OkCancelEditCombo]; title; prompt; default; choices)} is as above, but lets the user enter a value in the box above the list or select a value from the list.
- \texttt{@Prompt([OkCancelListMult]; title; prompt; default; choices)} is like \texttt{OkCancelList}, but allows the user to select multiple list elements and returns a list.
- \texttt{@Prompt([ChooseDatabase]; ""; ") displays a box that allows the user to select databases (server name, database file name, database title). Title text is at the top of the box, below it is a box for selecting the server, below it is a list box with the database titles, and below it is a box for the database file name.
- \texttt{@Prompt([LocalBrowse]; title; filetype)} displays a box that allows the user to select names from the local file system. Title text is at the top of the box, a list box for selecting files is at the left of the body of the box, below it is a list box for the type of file, and to the right is a list box for selecting the directory.
The *filetype* parameter is a text value, a number from 1 to 3, that specifies the types of files to display initially: "1" for NSF files only; "2" for NTF files only; "3" for files of all types.

- *@PickList([Custom] : [Single]; server : file; view; title; prompt; column)* displays a box with the title, prompt, and list of choices. The list is a view in a specified database. The user can select one (if [Single] is specified) or any number of elements (if [Single] is not specified); *@PickList* returns the values in the specified column for the selected list elements. This is like using *@DbColumn* to generate a list for *@Prompt*.

- *@PickList([Name] : [Single])* is as above, except that the database is an Address Book and the view is the "People" view. The user can select the Address Book in the *@PickList* dialog box.

Non-text values must be converted with *@Text* to be used as *@Prompt* and *@PickList* parameters. The text values can be constants, temporary variables, fields, or expressions.

The return value must be converted if it is to be used as a non-text value.

**Examples: Getting user input with *@Prompt* and *@PickList***

1. (YesNo). This validation formula queries the user concerning the TotalAmount field. If the user clicks No, a failure message is posted.
   ```plaintext
  @if(@Prompt([YesNo]; "Is this total within budget?"; @Text(TotalAmount; "C")); @Success; @Failure("Total not within budget"))
   ```
2. (OkCancelEdit). This button formula queries the user for a server and database, and opens the database. If the user clicks Yes without first entering a value in the edit box, the value returned is an empty string.
   ```plaintext
   server := @Prompt([OkCancelEdit]; "Server"; "Enter the name of a server"; "");
   database0 := @Prompt([OkCancelEdit]; "Database"; "Enter the name of a database on " + @If(server = ""; "your workstation"; server); "");
   database := @If(@Contains(database0; "."));
   @Command([FileOpenDatabase]; server : database)
   ```
3. (Password). This field validation formula gets a password from the user and compares it to the Password field in the document.
   ```plaintext
   pass := @Prompt([Password]; "Password"; "What is the password?");
   @If(pass = Password; @Success; @Failure("Password incorrect"))
   ```
4. (OkCancelList). This formula presents the user with a list of databases in a database catalog, and opens the database that the user selects. The first *@DbColumn* puts a list of the values in column 4 of the Databases by _Replica ID view in the temporary variable titles. The second *@DbColumn* puts a list of the values in column 2 of the Databases by _Replica ID view in the temporary variable servers. The third *@DbColumn* puts a list of the values in column 3 of the Databases by _Replica ID view in the temporary variable databases. The temporary variable list combines titles, servers, and databases for presentation to the user in *@Prompt*. The formula then parses the return value from *@Prompt* into a server name and database name for inclusion in the FileOpenDatabase *@command*.
   ```plaintext
   titles := @DbColumn(""; "doc"; "catalog.nsf"; "Databases by _Replica ID"; 4);
   servers := @DbColumn(""; "doc"; "catalog.nsf"; "Databases by _Replica ID"; 2);
   databases := @DbColumn(""; "doc"; "catalog.nsf"; "Databases by _Replica ID"; 3);
   list := titles + "+*=*+servers+" *=*+databases;
   member := @Prompt([OkCancelList]; "Open Database"; "Select a database"; ";"; list);
   server := @Left(@Right(member; "+"; "=*"); "+* ");
   database := @Right(member; "+* ");
   @Command([FileOpenDatabase]; server:database)
   ```
5. (OkCancelListMult). This button formula presents the user with a list of department names and sales totals. The user selects any number of elements from the list and the formula calculates a grand total.
   ```plaintext
   departments := @DbColumn(""; ""; "sales.nsf"; "Main View"; 1);
   totalSales := @DbColumn(""; ""; "sales.nsf"; "Main View"; 2);
   totalsList := @Text(totalSales; "C") + "+" + departments;
   sumList := @Prompt([OkCancelListMult]; "Total sales by department";
   "Select the ones you want to sum"; ""; totalsList);
   sum := @Sum(@TextToNumber(sumList)) @Prompt([Ok]; "Sum"; @Text(sum))
   ```
6. (Custom). This button formula presents the user with the "Databases by Replica ID" view from catalog.nsf on the server named doc. The user selects an element (row) from the list (view) and the formula opens that database with its name in column 3 of that row.

    name := @PickList([Custom]; "doc" : "catalog"; "Databases by Replica ID"; "Open database";
    "Select a database that is on server Doc"; 3);
    @Command([FileOpenDatabase]; "doc" : name)

**Filling out a form with @DialogBox**

@DialogBox displays a form of your design in a dialog box that has OK and Cancel buttons. When the user clicks OK, the contents of the fields in the dialog box transfer to any fields of the same name in the document in which @DialogBox is executing.

@DialogBox does not transfer rich text fields. These should not be included in your design.

The form that is displayed in the dialog box is best created using a layout region, and @DialogBox should use the [AutoVertFit] and [AutoHorzFit] options.

**Examples: Filling out a form with @DialogBox**

You have a form that has many fields, but a user creating a new document only fills in a few of them. You create another hidden form named "Dialog" that contains a layout region with the fields that the user typically fills in. At the top of the main form, you create a button with the following formula. When the user clicks the button, the formula displays a dialog box with the "Dialog" form. The user fills in the fields in the dialog box and clicks OK. The fields transfer to the main form.

    @DialogBox("Dialog"; [AutoVertFit] : [AutoHorzFit])

**Getting and setting environment variables**

You can set and retrieve the values of the environment variables in the notes.ini file (Windows™, OS/2®, and UNIX®).

- **@SetEnvironment(variable; value)** sets a named variable to the specified value. You can also use the `ENVIRONMENT` keyword and the two-parameter form of `@Environment`.
- **@Environment(variable)** retrieves the value of a named variable.

Environment values are text. Non-text values must be converted before being set and after being retrieved.

User environment variables are prepended with the $ character. If you add an environment variable with an editor or LotusScript, for example, and want to retrieve it with @Environment, the first character must be $.

Be sure you know which notes.ini file is affected by your formula. If the formula is in a database on a server, the formula runs on the server in the following cases: replication formula, agent for which the trigger is "After new mail has arrived" or "On schedule," selection formula, or column formula. Otherwise, the formula runs on the user's workstation. Replica copies access different NOTES.INI files, depending on which server or workstation contains the replica copy. Server access is subject to administrative restrictions.

Some uses for environment variables include the following:
- Pass temporary data among different formulas and databases
- Generate sequential numbers for one user

**Examples: Getting and setting environment variables**

1. This example converts a number to text and saves it as an environment variable.

    `ENVIRONMENT OrderNumber := @Text(NewOrderNumber);`

    These formulas are equivalent.
@Environment("OrderNumber"); @Text(NewOrderNumber);
@SetEnvironment("OrderNumber"); @Text(NewOrderNumber);

2. This example retrieves the value of an environment variable, converts it to a number, and stores it in a local variable.
OldOrderNumber := @TextToNumber(@Environment("OrderNumber");

3. This example is for a Computed when composed field. The formula maintains an environment variable named OrderNumber. When a new document is created, the formula retrieves the environment variable and increments it by 1. This algorithm does not work for databases that are replicated -- the database must exist on a single server or workstation and the formula must run on that same machine.
OldOrderNumber := @Environment("OrderNumber");
NewOrderNumber := @TextToNumber(@If(OldOrderNumber = ""; "0"; OldOrderNumber)) + 1;
ENVIRONMENT OrderNumber := @Text(NewOrderNumber);
NewOrderNumber;

4. The first formula is run once by each sales person. This formula prompts for the sales area and makes it the value of an environment variable named SalesArea. In the form for the sales documents, you make the second formula the default value formula for the SalesArea field. This formula retrieves the value of the environment variable. The salesperson does not need to enter the sales area for each new document, and neither is a particular sales area hard-coded into the default value formula.
ENVIRONMENT SalesArea := @Prompt([OkCancelList]; "Sales Area"; "What is your sales area?";
"Central"; "East" : "Central" : "West");
@Environment("SalesArea");

### Handling errors

Errors are of two types:
- **Syntax errors** are reported when you check or attempt to exit the formula you are writing.
- **Run-time errors** occur when the formula runs.

### Syntax errors

Syntax errors are reported when you either check or attempt to exit the formula you are writing. You receive a message specifying the error, and the line containing the error is highlighted. You must determine and then correct the error before proceeding. Common syntax errors include the following:
- Incorrectly spelled @function name or reserved word
- Missing or excessive parentheses
- Missing quotation mark on a string constant
- No semicolon between statements

You should write a formula a little at a time and check it often as you go along. Formula syntax tends to be complex, due to nesting.

### Run-time errors

Run-time errors occur when the formula runs. These errors can be categorized as follows:
- Unexpected -- These are development errors that your users should never see. For example, if you forget an @function parameter, the following message appears at run-time: "Insufficient arguments for @function." You should test your formula and attempt to correct all unexpected errors.
- Unreported -- These are results that are incorrect but are not reported as errors. For example, if you try to display a numeric value with @Prompt, @Prompt works but displays a blank. Again, your user should never see these errors. You should test your formula and ensure that all results are as expected.
- Expected -- These are errors that the user might cause at run-time. For example, if you prompt for the name of a database, the user might enter the name of a nonexistent database. You cannot prevent these errors, but you can anticipate and test for them in the formula and take appropriate actions.
The following @functions help you deal with run-time errors:

- @IsError(value) returns True (1) if a field, temporary variable, or expression contains an error.
- @Error generates an error.
- @Failure(message) displays a message when used in an input validation formula.
- @Success always returns the value 1.
- @Return(value) stops execution of the formula and returns a value.

Notes generates an error for a field if the built-in validation checking fails. For example, if you specify a field as numeric and the user enters a non-numeric value, Lotus Domino makes the value of the field an error. You can generate an error for a field by setting its value to @Error.

Notes reports errors in fields when the user attempts to save the document. For example, if a numeric field contains a non-numeric value, Lotus Domino generates the message "Cannot convert text to a number" when the user attempts to save a document.

To change the message or perform another action when an error occurs, test the field for an error with IsError in the field validation formula. You can generate your own error message with @Failure, but only in field validation formulas.

To incorporate your own error conditions for a field, return @Error if you detect an error condition.

Outside field formulas, for example, in an agent, button, or hotspot, you can check the contents of a field and react immediately to an error condition. For example, if you check a field with a button, you can change the field value or report the error before the user attempts to save the document. In checking for errors, be aware that the built-in validation checking generates an error as soon as the user enters a value in a field, but that a translation formula using @Error does not generate an error until the user attempts to save the document.

**Examples: Run-time errors**

1. This validation formula for the Price field causes a save operation for the document to fail with the specified message if the Price field contains an error.
   
   ```
   @If(@IsError(Price); @Failure("The Price field must be numeric"); @Success)
   ```

2. This example contains input translation and input validation formulas. The input translation formula puts an error into the Price field if its value is greater than 14.99. The input validation formula causes a save operation for the document to fail with the specified message if the Price field contains an error.
   
   ```
   REM "Input translation formula for Price field"
   @If(Price > 14.99; @ERROR; Price)
   REM "Input validation formula for Price field"
   @If(@IsError(Price); @Failure("The Price field must be numeric and 14.99 or less"); @Success)
   ```

3. This button formula checks the Price field for an error before applying a discount. If the field contains an error, the formula returns first giving the user a message.
   
   ```
   FIELD Price := @If(@IsError(Price) & @IsNumber(Price) & Price < 15; Price * 0.85; @Return(@Prompt([OkCancelEdit]; "Error in price field"; "Must be numeric and less than 15.00"; "")))
   ```

4. This button formula tests the Price field for an error. If the field contains an error, the formula resets it to a value taken from the user by @Prompt.
   
   ```
   FIELD Price := @If(@IsError(Price) & @IsNumber(Price) & Price < 15; Price; @TextToNumber(@Prompt([OkCancelEdit]; "Enter a number under $15.00"; "")))
   ```

5. This agent formula tests the return value of an @DbLookup statement for an error. If the statement causes an error, the formula returns the text "Not available."
result := @DbLookup("";"Snapper";"names.nsf";"People"; @Right(Name;" ") + ", " + @Left(Name;" "); "OfficePhoneNumber"); FIELD Phone := @If(@IsError(result);"Not available";result)

**Working with @functions**

All @functions evaluate to a value and can be placed in a formula anywhere a value of that type can be placed. When the formula executes, the value of the formula takes the place of the formula. Some formulas also have side-effects, that is, they cause actions to occur. For example, @Prompt causes a message box to appear.

Most @functions can be used in formulas for any Notes object, but some @functions are restricted in their applicability. The following table lists the @functions that are restricted and lists the Notes objects in which they can be used effectively. In addition, for an @function to return information on the current database, view, document, or field, these objects must be current.

<table>
<thead>
<tr>
<th>Restricted function</th>
<th>Function only works in these Notes objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>@All</td>
<td>Replication formula, agent, view selection formula</td>
</tr>
<tr>
<td>@AllChildren</td>
<td>Replication formula, view selection formula</td>
</tr>
<tr>
<td>@AllDescendants</td>
<td>Replication formula, view selection formula</td>
</tr>
<tr>
<td>@Command</td>
<td>Toolbar button, manual agent, action hotspot</td>
</tr>
<tr>
<td>@DbColumn (Domino data source)</td>
<td>Toolbar button, action, hotspot, field design, agent except mail</td>
</tr>
<tr>
<td>@DbLookup (Domino data source)</td>
<td>Toolbar button, action, hotspot, field design, agent except mail</td>
</tr>
<tr>
<td>@DeleteDocument</td>
<td>Agent</td>
</tr>
<tr>
<td>@DeleteField</td>
<td>Agent</td>
</tr>
<tr>
<td>@DocChildren</td>
<td>Column formula, window title formula</td>
</tr>
<tr>
<td>@DocDescendants</td>
<td>Column formula, window title formula</td>
</tr>
<tr>
<td>@DocLevel</td>
<td>Column formula, window title formula</td>
</tr>
<tr>
<td>@DocMark</td>
<td>Agent</td>
</tr>
<tr>
<td>@DocNumber</td>
<td>Column formula, window title formula</td>
</tr>
<tr>
<td>@DocParentNumber</td>
<td>Column formula, window title formula</td>
</tr>
<tr>
<td>@DocSiblings</td>
<td>Column formula, window title formula</td>
</tr>
<tr>
<td>@Failure</td>
<td>Field validation formula</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>All except formula pop-up hotspot</td>
</tr>
<tr>
<td>@Environment</td>
<td>All except formula pop-up hotspot when writing</td>
</tr>
<tr>
<td>FIELD</td>
<td>Toolbar button, agent, action hotspot, field design</td>
</tr>
<tr>
<td>@IsCategory</td>
<td>Column formula</td>
</tr>
<tr>
<td>@IsDocBeingLoaded</td>
<td>Form design, field design</td>
</tr>
<tr>
<td>@IsDocBeingMailed</td>
<td>Button, hotspot, field design</td>
</tr>
<tr>
<td>@IsDocBeingRecalculated</td>
<td>Button, hotspot, field design</td>
</tr>
<tr>
<td>@IsDocBeingSaved</td>
<td>Button, hotspot, field design</td>
</tr>
<tr>
<td>@IsExpandable</td>
<td>Column formula</td>
</tr>
<tr>
<td>@isNewDoc</td>
<td>Toolbar button, window title formula, form design, field design</td>
</tr>
<tr>
<td>@MailSend</td>
<td>Toolbar button, agent, action hotspot</td>
</tr>
<tr>
<td>Restricted function</td>
<td>Function only works in these Notes objects</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>@PickList</td>
<td>Toolbar button, manual agent, action hotspot, field design</td>
</tr>
<tr>
<td>@Platform</td>
<td>Toolbar button, manual agent, hotspot, view design except selection and column formulas, form design, field design</td>
</tr>
<tr>
<td>@Prompt</td>
<td>Toolbar button, manual agent, action hotspot, field design</td>
</tr>
<tr>
<td>@Responses</td>
<td>Window title formula, field design</td>
</tr>
<tr>
<td>@Return</td>
<td>Toolbar button, agent, hotspot, field design</td>
</tr>
<tr>
<td>SELECT</td>
<td>Replication formula, agent, view selection formula</td>
</tr>
<tr>
<td>@SetDocField</td>
<td>Toolbar button, agent, action hotspot, field design</td>
</tr>
<tr>
<td>@SetEnvironment</td>
<td>All except formula pop-up hotspot</td>
</tr>
<tr>
<td>@SetField</td>
<td>Toolbar button, agent, action hotspot, field design</td>
</tr>
<tr>
<td>@Success</td>
<td>Validation formula</td>
</tr>
<tr>
<td>@Unavailable</td>
<td>Agent</td>
</tr>
<tr>
<td>@ViewTitle</td>
<td>Agent</td>
</tr>
</tbody>
</table>

**Working with @commands**

@Commands are special @functions that perform immediate actions in the user interface. Most @commands mimic menu commands. For example, the following formula, if executed from a button, puts the current document in Edit mode and moves the insertion point down twice:

```plaintext
(Command([EditDocument]; "1");
(Command([EditDown]; "2")
```

The syntax for an @command is one of the following:

```plaintext
(Command([command-name]; arg1; arg2; ... argn)
@PostedCommand([command-name]; arg1; arg2; ... argn)
```

The name of the @function is @Command or @PostedCommand. The first argument is the name of the @command enclosed in brackets. The remaining arguments are the arguments to the @command.

For a list of @commands, see the @commands (A--Z) reference chapter.

You can use @commands in formulas for toolbar buttons, agents that do not specify target documents, events, button hotspots, and action hotspots. You cannot use @commands in a formula that does not interact with the user. These include replication, form, selection, column, hide action, window title, section title, section access, insert subform, hidden paragraph, default value, input translation, input validation, computed value, and keyword field formulas, and agents other than those that specify no target documents.

You cannot use most @commands in Web applications, since @commands are based on the Notes workstation user interface. About 30 @commands are supported but some behave differently. See "Programming Domino for Web Applications," "Formula language."

You cannot use @commands with LotusScript.

@Command functions execute in sequence with other @functions, with some exceptions. For example, the following formula executes the @command first:
@Command([EditDocument]; "1");
@Prompt([Ok]; "Edit mode"; "The document is now in Edit mode.")

@PostedCommand functions execute in sequence with each other after all other @functions execute. This emulates the behavior of @Command in Notes R3. For example, the following formula executes the @command last:
@PostedCommand([EditDocument]; "1");
@Prompt([Ok]; "Edit mode"; "The document will go into Edit mode.")

For additional information, see "Order of evaluation for formula statements" in the Formula Language Rules chapter.

You can check and respond to the return value of @Command (but not @PostedCommand). The return value is @True if the @command succeeds and @False if it fails. The following toolbar button formula returns if the FileOpenDatabase @command fails.
@If(@Command([FileOpenDatabase]; "NEWSUBJ"); ""; @Return(""));
@Command([Compose]; ""; "Main Topic");
@Command([EditGotoField]; "Subject");
@Command([EditInsertText]; "New subject");
@Command([EditGotoField]; "Body")

**Performing string operations**

Formula language @functions enable you to:

- Convert data types
- Concatenate, compare, and determine length
- Locate and extract substrings
- Trim, repeat, add a new line, and change case

**Converting data types**

Data must be the correct type for the operation involved. The following @functions convert data, and test for type.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Char(#number)</td>
<td>Converts an IBM® Code Page 850 code number to its corresponding character.</td>
</tr>
<tr>
<td>@IsNumber(#value)</td>
<td>Returns True (1) if a value is a number or number list.</td>
</tr>
<tr>
<td>@IsText(#value)</td>
<td>Returns True (1) if a value is a text string or text string list.</td>
</tr>
<tr>
<td>@IsTime(#value)</td>
<td>Returns True (1) if a value is a time-date or time-date list.</td>
</tr>
<tr>
<td>@Text(#value)</td>
<td>Converts a value to a text string.</td>
</tr>
<tr>
<td>@Text(#value; #format)</td>
<td>Converts a numeric or date-time value to a text string according to a specified format.</td>
</tr>
<tr>
<td>@TextToNumber(#string)</td>
<td>Converts a text string to a number.</td>
</tr>
<tr>
<td>@TextToTime(#string)</td>
<td>Converts a text string to a date-time value.</td>
</tr>
<tr>
<td>@TimeToTextInZone(#date-time; #time zone)</td>
<td>Converts a date-time value to a string that includes time zone information.</td>
</tr>
<tr>
<td>@TimeZoneToText(#time zone)</td>
<td>Converts a time zone value to a human-readable string.</td>
</tr>
<tr>
<td>@ToNumber(#string or #value)</td>
<td>Converts a number or text string to a number.</td>
</tr>
<tr>
<td>@ToTime(#string or #date-time value)</td>
<td>Converts a number or date-time value to a date-time value.</td>
</tr>
</tbody>
</table>
Examples: Converting data types

1. (@TextToNumber; @Text). This example reads a number interactively using @Prompt, calculates its logarithm, and displays the answer interactively using @Prompt. @Prompt works only with text values, so the input value i must be converted to a number before its log can be calculated, and the answer n must be converted to a text string before it can be displayed.
   \[ i := \text{@Prompt([OkCancelEdit]; "Enter a number");} \]
   \[ n := \text{@Log(@TextToNumber(i));} \]
   \[ @If(@IsError(n); @Prompt([OkCancelEdit]; "Enter a number"); "+ i + " is not a number")\]

2. (@Text). This example converts the current date-time as returned by @Now to text according to the date-time format T1S1 and displays it. T1 means the time only, and S1 means the hour and minute only. At six in the evening, the user sees an information box with "The time is ..." as the header and "06:00 PM" as the contents.
   \[ @Prompt([Ok]; "The time is ..."; @Text(@Now; "T1S1")) \]

3. (@Text). This example converts the number 800 to text according to the number format C,2 and displays it. C means the currency format (leading dollar sign) and 2 means two decimal places. The user sees an information box with "$800 dollars" as the header and "$800.00" as the contents.
   \[ @Prompt([Ok]; "$800 dollars"; @Text(800; "C,2")) \]

4. (@Text, @TextToTime). This example converts the text string "Today" to today’s date, then converts it to a text string for @Prompt. For example, if today is July 8, 1995, the user would see an information box with "Today’s Date" as the header and "07/08/95" as the contents.
   \[ @Prompt([Ok]; "Today’s Date"; @Text(@TextToTime("Today"))) \]

5. (@IsNumber). This example adds Fields A and B. If either field is not numeric, it is first converted to the numeric value 0. While a user cannot enter a non-numeric value into a numeric field, a numeric field that is left blank and has no default value formula defaults to a null text string. The @IsNumber function in this formula traps such an occurrence.
   \[ @If(@IsNumber(A); A; 0) + @If(@IsNumber(B); B; 0) \]

6. (@Char). This example converts the integer 65 to an uppercase A.
   \[ @Prompt([Ok]; "IBM Code Page 850 code 65"; @Char(65)) \]

Concatenating, comparing, and determining length

The + operator concatenates strings. The =, <>, ! =, >=, <, >, <=, and >= operators compare strings. The following @functions determine length and compare strings:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Compare</td>
<td>Compares two lists pair-wise.</td>
</tr>
<tr>
<td>@Length(string)</td>
<td>Returns the length of a string in characters.</td>
</tr>
<tr>
<td>@Length(stringlist)</td>
<td>Returns the length of each element of a string list in characters.</td>
</tr>
<tr>
<td>@Like</td>
<td>Uses _ (underline) to match any single character and % (percent sign) to match any sequence of characters in accordance with ANSI standard X3.135-1992.</td>
</tr>
<tr>
<td>@Like(string; pattern)</td>
<td>Determines whether two strings match. Conforms to ANSI SQL standard.</td>
</tr>
<tr>
<td>@Like(string; pattern; esc)</td>
<td>Same as above with an escape character.</td>
</tr>
<tr>
<td>@Matches</td>
<td>Uses ? to match any single character and * to match any sequence of characters, plus a number of other wildcards. @Matches uses \ as an escape character.</td>
</tr>
<tr>
<td>@Matches(string; pattern)</td>
<td>Determines whether two strings match. Wildcards can be used to expand the scope of the comparison.</td>
</tr>
</tbody>
</table>
Examples: Concatenating, comparing, and determining length

1. (+). This example concatenates the two strings to form ABCDEF.
   @Prompt([[Ok]; "Concatenation"; "ABC" + "DEF")

2. (+). This example concatenates the two input strings.
   Input1 := @Prompt([[OkCancelEdit]; "Concatenation - first element";
   "Enter any text in the box"; "ABC");
   Input2 := @Prompt([[OkCancelEdit]; "Concatenation - second element";
   "Enter any text in the box"; "DEF"); @Prompt([[Ok]; "Concatenation - result";
   Input1 + Input2)

3. (-). This example returns True to YesNo if the two input strings are equal, False if they are not equal.
   Input1 := @Prompt([[OkCancelEdit]; "Comparison - first element";
   "Enter any text in the box"; "ABC");
   Input2 := @Prompt([[OkCancelEdit]; "Comparison - second element";
   "Enter any text in the box"; "DEF");
   YesNo := @If(Input1 = Input2; "The strings are equal"; "The strings are not equal");
   @Prompt([Ok]; "Comparison - result"; YesNo)

4. (@Length). This example displays 9, the length of abcdefghi.
   @Prompt([Ok]; "Length of abdefghi"; @Text(@Length("abcdefghi"))

5. (@Length). This example creates a number list. Each element of this list contains the length of the corresponding element in TextList.
   @Length(TextList)

6. (@Matches). This example returns True to YesNo if Input equals abc.
   Input := @Prompt([[OkCancelList]; "@Matches Input"; "Choose one"; "abc"; "abc" : 
   "bcd" : "cde" : "xyz" : "123");
   YesNo := @If(@Matches(Input; "abc"); " matches abc"; " does not match abc");
   @Prompt([Ok]; "@Matches Result"; Input + YesNo)

7. (@Matches). This example returns True to YesNo if every character in Input is alphabetic, that is, in the range a-z. The set [a-z] specifies the character range and the preceding + means any number of occurrences of the set.
   Input := @Prompt([[OkCancelList]; "@Matches Input"; "Choose one"; "abc"; "abc" : 
   "bcd" : "cde" : "xyz" : "123");
   YesNo := @If(@Matches(Input; +[a-z]); " matches +[a-z]"; " does not match +[a-z]");
   @Prompt([Ok]; "@Matches Result"; Input + YesNo)

8. (@Matches). This example returns True to YesNo if every character in Input is not alphabetic, that is, is outside the set [a-z]. The specification ![a-z] means not in the set of characters a-z, and the preceding + means any number of occurrences of that set.
   Input := @Prompt([[OkCancelList]; "@Matches Input"; "Choose one"; "abc"; "abc" : 
   "bcd" : "cde" : "xyz" : "123");
   YesNo := @If(@Matches(Input; ![a-z]); " matches ![a-z]"; " does not match ![a-z]");
   @Prompt([Ok]; "@Matches Result"; Input + YesNo)

9. (@Matches). This Example returns True to YesNo if Input contains the characters bc in sequence surrounded by any number of any characters.
   Input := @Prompt([[OkCancelList]; "@Matches Input"; "Choose one"; "abc"; "abc" : 
   "bcd" : "cde" : "xyz" : "123");
   YesNo := @If(@Matches(Input; *bc*); " matches*bc*"); " does not match *bc*");
   @Prompt([Ok]; "@Matches Result"; Input + YesNo)

10. (@Matches). This Example returns True to YesNo if Input starts with a and is three characters long or starts with 1 and is three characters long.
    Input := @Prompt([[OkCancelList]; "@Matches Input"; "Choose one"; "abc"; "abc" : 
    "bcd" : "cde" : "xyz" : "123");
    YesNo := @If(@Matches(Input; a??|1??); " matches a??|1??"; " does not match a??|1??");
    @Prompt([Ok]; "@Matches Result"; Input + YesNo)

11. (@Like). This agent example checks for two sets of strings in the textBody field of each document. The first string is any text that contains "acquisition" and "Acme" in that order. The second string is
any text that contains "Acme" and "51%" in that order. The second @Like statement uses the slash (/) as an escape character so the percent sign (%) can be specified. Don’t use the backslash (\) as an escape character.

\@If(@Like(textBody; "%acquisition%Acme%")); @Prompt([Ok];
"Found reference to \"acquisition\"; Subject); "");
@If(@Like(textBody; "%Acme%51/%%"; "/")); @Prompt([Ok];
"Found reference to \"51\%\"; Subject); "");
SELECT @All

12. This action compares a list to the value "N" and displays the result. Boston and Moscow result in -1 (less than N), Tokyo results in 1 (greater than N), and n and N result in 0.

list := "Boston" : "Tokyo" : "Moscow" : "N" : "n";
result := @text(@compare(list; "N"; [CaseInsensitive]));
@Prompt([OkCancelList] : [NoSort];
"Result"; "; "; list + (+ result + ")"

### Locating and extracting substrings

The following @functions locate and extract substrings:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Begins(string ; sub)</td>
<td>Determines whether a string begins with a substring.</td>
</tr>
<tr>
<td>@Contains(string ; sub)</td>
<td>Determines whether a string contains a substring.</td>
</tr>
<tr>
<td>@Contains(string ; list)</td>
<td>Determines whether a string contains any substring in a list.</td>
</tr>
<tr>
<td>@Ends(string ; sub)</td>
<td>Determines whether a string ends with a substring.</td>
</tr>
<tr>
<td>@FileDir(pathname)</td>
<td>Extracts the directory part of a file path name.</td>
</tr>
<tr>
<td>@Left(string ; n)</td>
<td>Extracts the leftmost n characters from a string.</td>
</tr>
<tr>
<td>@Left(string ; sub)</td>
<td>Extracts the leftmost characters from a string up to a substring, searching left to right.</td>
</tr>
<tr>
<td>@LeftBack(string ; n)</td>
<td>Extracts the leftmost characters from a string up to the nth character from the right.</td>
</tr>
<tr>
<td>@LeftBack(string ; sub)</td>
<td>Extracts the leftmost characters from a string up to a substring, searching right to left.</td>
</tr>
<tr>
<td>@Middle(string ; off ; n)</td>
<td>Extracts n characters from a string starting at an offset, searching left to right.</td>
</tr>
<tr>
<td>@Middle(string ; sub ; n)</td>
<td>Extracts n characters from a string starting at a substring, searching left to right.</td>
</tr>
<tr>
<td>@Middle(string ; off ; sub)</td>
<td>Extracts characters from a string starting at an offset and stopping at a substring, searching left to right.</td>
</tr>
<tr>
<td>@Middle(string ; sub ; sub)</td>
<td>Extracts characters from a string starting and stopping at substrings, searching left to right.</td>
</tr>
<tr>
<td>@MiddleBack(str ; off ; n)</td>
<td>Extracts n characters from a string starting at an offset, searching right to left.</td>
</tr>
<tr>
<td>@MiddleBack(str ; sub ; n)</td>
<td>Extracts n characters from a string starting at a substring, searching right to left.</td>
</tr>
<tr>
<td>@MiddleBack(str ; off ; sub)</td>
<td>Extracts characters from a string starting at an offset and stopping at a substring, searching right to left.</td>
</tr>
<tr>
<td>@MiddleBack(str ; sub ; sub)</td>
<td>Extracts characters from a string starting and stopping at substrings, searching right to left.</td>
</tr>
<tr>
<td>@ReplaceSubstring(source ; from ; to)</td>
<td>Replaces &quot;from&quot; with &quot;to&quot; in source. If &quot;from&quot; and &quot;to&quot; are lists, replaces corresponding entries in order.</td>
</tr>
<tr>
<td>@Right(string ; n)</td>
<td>Extracts the rightmost n characters from a string.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@Right(string; sub)</td>
<td>Extracts the rightmost characters from a string up to a substring, searching left to right.</td>
</tr>
<tr>
<td>@RightBack(string; n)</td>
<td>Extracts the rightmost characters from a string up to the nth character from the left.</td>
</tr>
<tr>
<td>@RightBack(string; sub)</td>
<td>Extracts the rightmost characters from a string up to a substring, searching left to right.</td>
</tr>
<tr>
<td>@Word(string; sep; n)</td>
<td>Extracts word n from a string where words are the text between specified separators.</td>
</tr>
<tr>
<td>@Word(list; sep; n)</td>
<td>Extracts word n from each string in a list where words are the text between specified separators.</td>
</tr>
</tbody>
</table>

Examples: Locating and extracting substrings

1. (@Contains). This example returns True to R if Substring is anywhere in String. The search is case sensitive.

   ```
   String := @Prompt([OkCancelEdit]; "String"; "Enter a string"; ");
   Substring := @Prompt([OkCancelEdit]; "Substring"; "Enter a beginning substring"; ");
   Yes := Substring + " is in " + String;
   No := Substring + " is not in " + String;
   R := @Contains([OkCancelEdit]; String; Substring);
   @IF(R; @Prompt([Ok Cancel Edit]; "Yes"); Yes; @Prompt([Ok Cancel Edit]; "No"); No)
   ```

2. (@Contains). This Example returns True to R if Substring1 or Substring2 is anywhere in String. The search is case sensitive.

   ```
   String := @Prompt([OkCancelEdit]; "String"; "Enter a string"; ");
   Substring1 := @Prompt([OkCancelEdit]; "Substring"; "Enter substring 1"; ");
   Substring2 := @Prompt([OkCancelEdit]; "Substring"; "Enter substring 2"; ");
   Yes := Substring1 + " or " + Substring2 + " is in " + String;
   No := Substring1 + " and " + Substring2 + " are not in " + String;
   R := @Contains([OkCancelEdit]; String; Substring1; Substring2);
   @IF(R; @Prompt([Ok Cancel Edit]; "Yes"); Yes; @Prompt([Ok Cancel Edit]; "No"); No)
   ```

3. (@Left). This example returns to R the leftmost N characters in String.

   ```
   String := @Prompt([OkCancelEdit]; "String"; "Enter a string"; ");
   Number := @Prompt([OkCancelEdit]; "Number of characters"; "Enter a number of characters"; ");
   N := @TextToNumber([Ok Cancel Edit]; Number);
   R := @Left(String; N); @Prompt([Ok Cancel Edit]; "Leftmost characters"); R
   ```

4. (@Left). This example returns to R the characters left of Substring in String.

   ```
   String := @Prompt([OkCancelEdit]; "String"; "Enter a string"; ");
   Substring := @Prompt([OkCancelEdit]; "Substring"; "Enter a substring"; ");
   R := @Left(String; Substring); @Prompt([Ok Cancel Edit]; "Characters left of " + Substring; R)
   ```

5. (@RightBack, @Left). If the common name in the ComposedBy field is "Judith Woo," this formula calculates "Woo, Judith." @RightBack returns the last name and @Left returns the first name.

   ```
   @RightBack([Name]; [ComposedBy]; "); " + "); " + @Left([Name]; [ComposedBy]; "); "
   ```

6. (@Middle). This example returns N characters to R from String starting after Substring.

   ```
   String := @Prompt([OkCancelEdit]; "String"; "Enter a string"; ");
   Substring := @Prompt([OkCancelEdit]; "Substring"; "Enter a substring"; ");
   Number := @Prompt([OkCancelEdit]; "Number of Characters"; "Enter the number of characters"; ");
   N := @TextToNumber([Ok Cancel Edit]; Number);
   R := @Middle(String; Substring; N); @Prompt([Ok Cancel Edit]; Number + " characters starting after " + Substring; R)
   ```

7. (@ReplaceSubstring). This agent example makes three substitutions in the textBody field of the affected documents. The third substitution is in case the second substitution causes a sentence to end with two periods.

   ```
   FIELD textBody := @ReplaceSubstring(textBody; "Acme": "mousetrap": "); " + "); " + "Acme, Inc." : "mouse detention device": ");
   SELECT @A1
   ```
8. (@Word). This example extracts word n from string s, using a space as the word separator.

\[ s := @Prompt([OkCancelEdit]; "String"; "Enter a string of words"; "); \\
 n := @Prompt([OkCancelEdit]; "Word"; "Enter the number of the word to extract"; "); \\
 ss := @Word(s; "); @TextToNumber(n); @Prompt([Ok]; "Substring"; "Word " + n + " is \"" + ss + \""") \]

9. (@FileDir) This example extracts "c:\market\data\" from the specified path name.

\[ @FileDir("c:\market\data\europe.dat") \]

**Trimming, repeating, adding a new line, and changing case**

The following @functions trim strings, repeat characters, add a new line (carriage return), and change case:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@LowerCase(string)</td>
<td>Converts all uppercase characters in a string to lowercase.</td>
</tr>
<tr>
<td>@NewLine</td>
<td>Inserts a new line (carriage return) into a text string.</td>
</tr>
<tr>
<td>@ProperCase</td>
<td>Converts the first character of each word in a string to uppercase and the remaining characters to lowercase.</td>
</tr>
<tr>
<td>@Repeat(string, number)</td>
<td>Repeats a string a number of times.</td>
</tr>
<tr>
<td>@Trim(string)</td>
<td>Removes leading, trailing, and redundant spaces from a string.</td>
</tr>
<tr>
<td>@Trim(list)</td>
<td>Removes leading, trailing, and redundant spaces from each element of a string list, and removes blank elements from the list.</td>
</tr>
<tr>
<td>@UpperCase(string)</td>
<td>Converts all lowercase characters in a string to uppercase.</td>
</tr>
</tbody>
</table>

**Examples: Trimming, repeating, adding a new line, and changing case**

1. (@Trim). This example returns [Now is the time], removing all extraneous spaces.

\[ Untrimmed := " Now is the time "; \\
 Trimmed := @Trim(Untrimmed); \\
 @Prompt([Ok]; "Untrimmed"; "; + Untrimmed + "); \\
 @Prompt([Ok]; "Trimmed"; "; + Trimmed + "); \]

2. (@Trim, @ProperCase). This example converts the words in Name to initial uppercase characters and deletes leading, trailing, and extraneous spaces. If ”jane j smith” is entered for Name, this formula converts it to ”Jane J Smith.”

\[ @Trim(@ProperCase(Name)) \]

3. (@Repeat). This example returns ”Great Month! Great Month! Great Month!” in the Comments field if the Sales field is 100,000 or greater.

\[ FIELD Comments := @If(Sales >= 100000; @Repeat("Great Month! "; 3); Sales >= 50000; "Good Month"; "I want to see you in my office"); SELECT @All \]

4. (@NewLine). This example returns the user name and the date separated by a new line.

\[ @UserName + @NewLine + @Text(@Now) \]

5. (@LowerCase). This example returns the user name in lowercase.

\[ @LowerCase(@UserName) \]

6. (@UpperCase). This example returns the user name in uppercase.

\[ @UpperCase(@UserName) \]
Performing arithmetic operations

The * / + - operators multiply, divide, add, and subtract. Multiplication and division have precedence over addition and subtraction; otherwise, evaluation is left to right. Parentheses can be used to change the order of evaluation. The following are the arithmetic @functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Abs(number)</td>
<td>Calculates the absolute (unsigned) value of a number.</td>
</tr>
<tr>
<td>@ACos(cosine)</td>
<td>Calculates the arc (inverse) cosine of a cosine.</td>
</tr>
<tr>
<td>@ASin(sine)</td>
<td>Calculates the arc (inverse) sine of a sine.</td>
</tr>
<tr>
<td>@ATan(tangent)</td>
<td>Calculates the arc (inverse) tangent of a tangent.</td>
</tr>
<tr>
<td>@ATan2(x; y)</td>
<td>Calculates the arc (inverse) tangent using the tangent y/x of an angle.</td>
</tr>
<tr>
<td>@Cos(angle)</td>
<td>Calculates the cosine of an angle (in radians).</td>
</tr>
<tr>
<td>@Exp(number)</td>
<td>Calculates e raised to the power of a number.</td>
</tr>
<tr>
<td>@FloatEq(number ; number ; range)</td>
<td>Compares two numbers for equality within a confidence range.</td>
</tr>
<tr>
<td>@Integer(numlist)</td>
<td>Truncates the elements of a number list to integers.</td>
</tr>
<tr>
<td>@Integer(number)</td>
<td>Truncates a number to an integer.</td>
</tr>
<tr>
<td>@Log(number)</td>
<td>Calculates the common (base 10) logarithm of a number.</td>
</tr>
<tr>
<td>@Ln(number)</td>
<td>Calculates the natural (base e) logarithm of a number.</td>
</tr>
<tr>
<td>@Max(number ; number)</td>
<td>Calculates the larger of two numbers.</td>
</tr>
<tr>
<td>@Max(numlist)</td>
<td>Calculates the largest number in a list.</td>
</tr>
<tr>
<td>@Min(number ; number)</td>
<td>Calculates the smaller of two numbers.</td>
</tr>
<tr>
<td>@Min(numlist)</td>
<td>Calculates the smallest number in a list.</td>
</tr>
<tr>
<td>@Modulo(number ; number)</td>
<td>Calculates the remainder of a number divided by a second number.</td>
</tr>
<tr>
<td>@Modulo(numlist ; numlist)</td>
<td>In a pairwise list operation, calculates the remainder of a number divided by a second number.</td>
</tr>
<tr>
<td>@Pi</td>
<td>Calculates the value of Pi.</td>
</tr>
<tr>
<td>@Power(base ; exp)</td>
<td>Calculates the value of a base raised to the power of an exponent.</td>
</tr>
<tr>
<td>@Random</td>
<td>Returns a random number in the range 0 to 1, inclusive.</td>
</tr>
<tr>
<td>@Round(number)</td>
<td>Rounds a number to the nearest integer.</td>
</tr>
<tr>
<td>@Round(number ; factor)</td>
<td>Rounds a number to the nearest specified factor.</td>
</tr>
<tr>
<td>@Round(numlist)</td>
<td>Rounds each number in a list to the nearest integer.</td>
</tr>
<tr>
<td>@Round(numlist ; factor)</td>
<td>Rounds each number in a list to the nearest specified factor.</td>
</tr>
<tr>
<td>@Sign(number)</td>
<td>Returns 1 for a positive number, -1 for a negative number, and 0 for zero.</td>
</tr>
<tr>
<td>@Sin(angle)</td>
<td>Calculates the sine of an angle (in radians).</td>
</tr>
<tr>
<td>@Sqrt(number)</td>
<td>Calculates the square root of a number.</td>
</tr>
<tr>
<td>@Sum(num; num; ...)</td>
<td>Calculates the sum of numbers and number lists.</td>
</tr>
<tr>
<td>@Tan(angle)</td>
<td>Calculates the tangent of an angle (in radians).</td>
</tr>
</tbody>
</table>
Examples: Performing arithmetic operations

1. (*, precedence) This example prints 15 for the first @Prompt because the multiplication 4 * 3 is evaluated first. It prints 21 for the second @Prompt because the parentheses force the evaluation of 3 + 4 first.
   @Prompt([Ok]; "3 + 4 * 3"); @Text(3 + 4 * 3); @Prompt([Ok]; "(3 + 4) * 3"); @Text((3 + 4) * 3)

2. (/ *) This example prints 0.333333333333333 for the first @Prompt, performing the division and rounding the result to 15 decimal places. It prints 1.2635268885E+17 for the second @Prompt, presenting the 11 most significant digits of the result as a fraction multiplied by 1017.
   @Prompt([Ok]; "1 / 3"); @Text(1 / 3); @Prompt([Ok]; "123456789 * 1023456789");
   @Text(123456789 * 1023456789)

3. (@Abs) This example calculates the difference between Score1 and Score2 as an unsigned number, no matter which is larger.
   @Abs(Score1 - Score2)

4. (@Abs) This example calculates the absolute difference between Sales and CostOfSales, and formats it in a text field placing parentheses around a negative result.
   GP := @Abs(Sales - CostOfSales); @If(Sales >= CostOfSales; @Text(GP); 
   "(" + @Text(GP) + ")")

5. (@Sign) This agent example displays the Total field. If the field value is negative, its absolute value is placed in parentheses; if the field value is zero, the word "Zero" is displayed.
   sign := @Sign(Total);
   display := @If(sign = 1; @Text(Total); sign = -1; "(" + @Text(@Abs(Total)) + "); "Zero");
   @Prompt([Ok]; "Total"; display);
   SELECT @All

6. (@Sum) This example prints 15, the sum of the list One23, the variable Four, and the constant 5.
   One23 := 1 : 2 : 3; Four := 4;
   S := @Sum(One23; Four; 5); @Prompt([Ok]; "Sum of 1-5"; @Text(S))

7. (@Integer) This example truncates 3.12 to 3 and 6.735 to 6.
   @Prompt([Ok]; "@Integer(3.12)"); @Text(@Integer(3.12)); @Prompt([Ok]; "@Integer(6.735)");
   @Text(@Integer(6.735))

8. (@Integer) This example truncates Sales and Commission to integers in a list.
   @Integer(\text{Sales : Commission})

9. (@Round) This example rounds 3.12 to 3, 6.735 to 7, and 7.5 to 8; 753 by tens to 750; and the list elements 3.12, 6.735, and 7.5 to 3, 6, and 7 respectively (converting them to a text string for display).
   @Prompt([Ok]; "@Round(3.12)"; @Text(@Round(3.12))); @Prompt([Ok]; "@Round(6.735)");
   @Text(@Round(6.735))); @Prompt([Ok]; "@Round(7.5)"; @Text(@Round(7.5))); @Prompt([Ok];
   "@Round(753; 10)"; @Text(@Round(753; 10))); @Prompt([Ok]; "@Round(3.12 : 6.735 : 7.5)"");
   @Implode(@Text(@Round(3.12 : 6.735 : 7.5))))

10. (@Max) This example prints 99, the maximum of 99, 2, and 3; 3, the maximum of 1 and 3; and 99 6 78, the maximum of the pair-wise elements in the two lists.
    @Prompt([Ok]; "@Max(99 : 2 : 3)"); @Text(@Max(99 : 2 : 3)); @Prompt([Ok]; "@Max(1 : 3)");
    @Text(@Max(1 : 3)); @Prompt([Ok]; "@Max(99 : 2 : 3; 5 : 6 : 7 : 8)"; @Implode(@Text(@Max(99 : 2 : 3; 5 : 6 : 7 : 8))))

11. (@Min) This example prints 2, the minimum of 99, 2, and 3; 1, the minimum of 1 and 3; and 5 2 3 3, the minimum of the pair-wise elements in the two lists.
    @Prompt([Ok]; "@Min(99 : 2 : 3)"); @Text(@Min(99 : 2 : 3)); @Prompt([Ok]; "@Min(1 : 3)");
    @Text(@Min(1 : 3)); @Prompt([Ok]; "@Min(99 : 2 : 3; 5 : 6 : 7 : 8)"; @Implode(@Text(@Min(99 : 2 : 3; 5 : 6 : 7 : 8))))

12. (@Modulo) This example prints 1, the remainder of 4/3; -2, the remainder of -14/3 (the remainder is negative when the dividend is negative); and 1 2 3 3, the remainders of the pairwise division of the first list by the second in the third line.
Performing time-date operations

A time-date value consists of a year, month, day, hour, minute, and second. You can use a time-date value "as is" in a time-date field, but must convert it with @Text to use it as a string. You can convert a string to a time-date value with @TextToTime.

A time-date constant is a date, a time, or both, in brackets. The date is the month, day, and year separated by a slash (/) or a hyphen (-) for OS/2. Year is optional and defaults to the current year; a 2-digit year means the 20th century if 50 or greater, and the 21st century if less than 50.
The time is the hour, minute, and second (optional; defaults to 0) separated by colons. You can use 24-hour time or add "PM" for afternoon hours. You can add the time zone to indicate another time zone. Separate the components with spaces. Some examples of time-date constants are [6/30/97], [5:30:00 PM], [17:30:00], [17:30 EST], and [6/30 5:30 PM].

Dates can be compared and subtracted. Subtraction yields a numeric value representing seconds. To measure the difference between two dates in days, divide the result by 86,400, which is the number of seconds in a day. For example, if you have two date fields, date1 containing [07/01/01] and date2 containing [07/05/01], the following returns 345,600:
\[
\text{date2-date1}
\]

To display the result as 4 days instead of 345,600 seconds, use the following code:
\[
\text{(date2-date1)}/86,400
\]

For more information on calculating dates, see \texttt{@Adjust}

The following \texttt{@functions} determine and manipulate time-date values.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Accessed</td>
<td>Returns the time-date the document was last accessed.</td>
</tr>
<tr>
<td>@Adjust (\text{time-date}; ; y; ; m; ; d; ; h; ; m; ; s)</td>
<td>Adjusts a time-date by the negative or positive values of the remaining parameters.</td>
</tr>
<tr>
<td>@BusinessDays</td>
<td>Returns the number of business days in one or more date ranges.</td>
</tr>
<tr>
<td>@Created</td>
<td>Returns the time-date the document was created.</td>
</tr>
<tr>
<td>@Date (y; ; m; ; d)</td>
<td>Returns the date for year, month, and day.</td>
</tr>
<tr>
<td>@Date (y; ; m; ; d; ; h; ; m; ; s)</td>
<td>Returns the date for year, month, day, hour, minute, and second.</td>
</tr>
<tr>
<td>@Date(time-date)</td>
<td>Returns the date for a time-date.</td>
</tr>
<tr>
<td>@Day(time-date)</td>
<td>Extracts the day of the month from a time-date.</td>
</tr>
<tr>
<td>@Hour(time-date)</td>
<td>Extracts the hour from a time-date.</td>
</tr>
<tr>
<td>@Minute(time-date)</td>
<td>Extracts the minute from a time-date.</td>
</tr>
<tr>
<td>@Modified</td>
<td>Returns the time-date the document was last edited and saved.</td>
</tr>
<tr>
<td>@Month(time-date)</td>
<td>Extracts the month from a time-date as 1-12.</td>
</tr>
<tr>
<td>@Now</td>
<td>Returns the current time-date.</td>
</tr>
<tr>
<td>@Now([ServerTime])</td>
<td>Returns the current time-date for the server containing the current database.</td>
</tr>
<tr>
<td>@Now([ServerTime]; \textit{serverNames})</td>
<td>Returns the current time-date for specified servers.</td>
</tr>
<tr>
<td>@Second(time-date)</td>
<td>Extracts the seconds from a time-date.</td>
</tr>
<tr>
<td>@Time (y; ; m; ; d)</td>
<td>Returns the time for year, month, and day.</td>
</tr>
<tr>
<td>@Time (y; ; m; ; d; ; h; ; m; ; s)</td>
<td>Returns the time for year, month, day, hour, minute, and second.</td>
</tr>
<tr>
<td>@Time(time-date)</td>
<td>Returns the time for a time-date.</td>
</tr>
<tr>
<td>@Today</td>
<td>Returns today’s date.</td>
</tr>
<tr>
<td>@Tomorrow</td>
<td>Returns tomorrow’s date.</td>
</tr>
<tr>
<td>@Weekday(time-date)</td>
<td>Returns the day of the week for a time-date as 1-7 (Sunday through Saturday).</td>
</tr>
<tr>
<td>@Year(time-date)</td>
<td>Extracts the year from a time-date.</td>
</tr>
<tr>
<td>@Yesterday</td>
<td>Returns yesterday’s date.</td>
</tr>
<tr>
<td>@Zone</td>
<td>Returns the time zone setting of the current computer.</td>
</tr>
</tbody>
</table>
Examples: Performing time-date operations

1. (Created) This agent example writes "Archive" to the Status field if the current document was created before 1995, and writes "Current" otherwise.
   
   ```
   SELECT @TA1;
   FIELD Status := @If(@Created < [01/01/95 12:00:00 AM]; "Archive"; "Current");
   ```

2. (Modified, Date, Today, Yesterday) This agent example writes "Today," "Yesterday," or "Old" to the ViewStatus field, depending on the date that the current document was last modified.
   
   ```
   FIELD ViewStatus := @If(@Date(@Modified) = @Today; "Today");
   @Date(@Modified) = @Yesterday; "Yesterday");
   ```

3. (Modified, Date, Weekday, Today, Adjust, Yesterday) This agent example modifies the preceding example to exclude weekends in assigning "Yesterday." If today is Monday, y is set to today’s date minus 3 days; otherwise y is set to yesterday’s date. Then y instead of @Yesterday is tested against the @Modified date.
   
   ```
   d := @Date(@Modified);
   y := @If(@Weekday(@Today) = 2; @Adjust(@Today; 0; 0; -3; 0; 0); @Yesterday);
   FIELD ViewStatus := @If(d = @Today; "Today");
   ```

4. (Now, Month, Year, Day) This example of a computed text field displays today’s date in letterhead form. For example, 6/30/95 displays as "June 30, 1995."
   
   ```
   months := "January"; "February"; "March"; "April"; "May"; "June"; "July";
   "August"; "September"; "October"; "November"; "December";
   month := @Subset(@Subset(months; @Month(@Now)); -1);
   year := @Year(@Now);
   day := @Day (@Now);
   month + " " + @Text(day) + ", " + @Text(year)
   ```

5. (Adjust, Weekday, Created) This example of a computed time field displays a date two days from the creation date for the document. The calculation eliminates the weekend by adding 4 days if the creation date is a Friday.
   
   ```
   increment := @If(@Weekday(@Created) = 6; 4; 2);
   @Date(@Adjusted; @Created; 0; increment; 0; 0; 0))
   ```

6. (BusinessDays) This agent displays the number of days in 2001 excluding Saturdays, Sundays, and 10 holidays.
   
   ```
   @Prompt([Ok];
   @Text(
   @BusinessDays([01/01/2001]; [12/31/2001]; 1 : 7;
   [01/01/2001]; [01/15/2001]; [02/16/2001]; [05/28/2001]; [07/04/2001];
   );
   "Business days in 2001")
   ```

Accessing the user environment

The user environment is the server or workstation containing the database for: replication formulas, agents with the triggers "After new mail has arrived" or "On schedule," selection formulas, or column formulas. Otherwise, the user environment is the Notes workstation of the user running the formula.

User names can be either distinguished or non-distinguished, and distinguished names can be canonical or abbreviated. Use @Name to change the form of a user name.
The following @functions return or process information on the user environment.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Domain</td>
<td>Returns the name of the user’s Lotus Notes/Domino mail domain.</td>
</tr>
<tr>
<td>@MailDbName</td>
<td>Returns the server name and path name for the user’s mail database. This @function evaluates to a 2-member list.</td>
</tr>
<tr>
<td>@Name[key]; name)</td>
<td>Changes the form of a user name. The keywords include [CN] to extract the common name from a distinguished name, [Abbreviate] to abbreviate a distinguished name in canonical form, [Canonicalize] to do the reverse, and [ToKeyword] to put the name components in reverse order separated by backslashes (for categorized views).</td>
</tr>
<tr>
<td>@OptimizeMailAddress(address)</td>
<td>Removes unnecessary domains from an address.</td>
</tr>
<tr>
<td>@Password(string)</td>
<td>Encodes a string. You cannot determine the original string from the encoded result.</td>
</tr>
<tr>
<td>@Platform</td>
<td>Returns the platform that the user is running on: Macintosh, NetWare, OS2V1, OS2V2, UNIX, Windows/16, or Windows/32.</td>
</tr>
<tr>
<td>@StatusBar</td>
<td>Writes a message or messages to the status bar.</td>
</tr>
<tr>
<td>@UserAccess</td>
<td>Given a server and file name, indicates the user’s level of access to the database.</td>
</tr>
<tr>
<td>@UserNamesList</td>
<td>Returns a text list containing the user’s common name, hierarchical names, ACL roles (if any), and if the database is on a server, any groups the user belongs to.</td>
</tr>
<tr>
<td>@UserPrivileges</td>
<td>Returns a text list of the user’s privileges.</td>
</tr>
<tr>
<td>@UserRoles</td>
<td>For databases on servers, returns a list of roles for the current user.</td>
</tr>
<tr>
<td>@Version</td>
<td>Returns (as a string) the version of Notes that is running.</td>
</tr>
</tbody>
</table>

Examples: Accessing the user environment

1. This view selection formula limits the view to documents where the From_1 field matches the name of the current user. Both From_1 and @UserName are reduced to the common name portion of the hierarchical name to better ensure a match.
   
   SELECT @Name([CN]; @UserName) = @Name([CN]; From_1)

2. In this column formula, the @Name function extracts the common name from the “From” field.
   
   Subject + " (" + @Name([CN]; From) + @DocDescendants(""; ", % response); ", % responses)"

3. This formula displays information about the user environment. The return value from @MailDbName is imploded because it is a 2-element list containing server name and path name.
   
   @Prompt([Ok]; "User name";
   @Name([CN]; @UserName));
   @Prompt([Ok]; "Mail database"; @Implode(@MailDbName));
   @Prompt([Ok]; "Platform"; @Platform);
   @Prompt([Ok]; "Notes version"; @Version)

4. This is the formula for the first column in the “By Author” view. It converts the From field, which typically contains a distinguished name, to the form last name, comma, first name.
   
   AuthorName := @If(10IsAvailable(From); "Anonymous"; @Name([CN]; From));
   Name := @Trim(@Word(AuthorName; ";"; 1));
   LastName := @RightBack(Name; " ");
   FirstName := @LeftBack(Name; " ");
   CombinedName := LastName + ", " + FirstName;
   @If(CombinedName = ", "; Name; CombinedName)

5. This is the input validation formula for the “Password” field on a form. The author can see the password as it is being typed, but when the document is saved, the password is encoded and cannot be read.
   
   @Password(Password)
Accessing the current database and view

You have immediate access to the database in which the formula is running, except for toolbar button formulas, which have no database context. You have immediate access to the view in which the formula is running if you are in the context of a view. In the context of a document, you have immediate access to the view from which the document was opened.

Database and view attributes

The following table lists @functions that return database and view attributes.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DbManager</td>
<td>Returns the users, groups, and servers that currently have manager access to the database. Returns a list.</td>
</tr>
<tr>
<td>@DbName</td>
<td>Returns the names of the current Domino server and database. Returns a 2-element list.</td>
</tr>
<tr>
<td>@DbTitle</td>
<td>Returns the title of the current database.</td>
</tr>
<tr>
<td>@ReplicaID</td>
<td>Returns the replica ID of the current database.</td>
</tr>
<tr>
<td>@ServerName</td>
<td>Returns the name of the server containing the current database.</td>
</tr>
<tr>
<td>@ViewTitle</td>
<td>Returns the title of the current view.</td>
</tr>
</tbody>
</table>

Window title and column formula @functions

A number of @functions provide response hierarchy and other information on views. In a view, main documents are numbered 1, 2, 3, and so forth. Each set of response or response-to-response documents have second and third numbers starting at 1. By default, the complete number for a response document appears as a decimal. For example, the second response to the third document is document number 3.2 in the view; the first response to the second response is document number 3.2.1.

These @functions work only in Window Title and Column formulas, and some, as noted, are restricted to either one or the other. The return value in every case is a string.

For details on more advanced formulas for columns, see "Advanced options for columns" in the Application Development in Domino Designer book.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DocChildren</td>
<td>Returns the number of immediate children of the current document.</td>
</tr>
<tr>
<td>@DocChildren(def)</td>
<td>As above, but returns def. Use % in def to represent the number.</td>
</tr>
<tr>
<td>@DocChildren(zero; def)</td>
<td>As above, but returns zero if there are no children.</td>
</tr>
<tr>
<td>@DocChildren(one, zero; def)</td>
<td>As above, but returns one if there is one child.</td>
</tr>
<tr>
<td>@DocDescendants</td>
<td>Returns the number of descendants, including children and children of children, of the current document.</td>
</tr>
<tr>
<td>@DocDescendants(def)</td>
<td>As above, but returns def. Use % in def to represent the number.</td>
</tr>
<tr>
<td>@DocDescendants(zero; def)</td>
<td>As above, but returns zero if there are no descendants.</td>
</tr>
<tr>
<td>@DocDescendants(one, zero; def)</td>
<td>As above, but returns one if there is one descendant.</td>
</tr>
<tr>
<td>@DocLevel</td>
<td>Returns the level of the current document in the current view.</td>
</tr>
<tr>
<td>@DocNumber</td>
<td>Returns the number of the current document or category within the current view.</td>
</tr>
<tr>
<td>@DocNumber(sep)</td>
<td>As above, but separates the components of the number with sep rather than a period.</td>
</tr>
</tbody>
</table>
### Examples: Accessing the current database and view

1. This example displays the database title, its server and database name, its replica ID, and the names of users who have manager access.

```plaintext
@Prompt([Ok]; "Title": @DbTitle);
@Prompt([Ok]; "Server and database": @Trim(@Implode(@DbName)));
@Prompt([Ok]; "Replica ID": @ReplicaID);
@Prompt([Ok]; "Managers": @Implode(@DbManager; ", ")
```

2. This window title formula displays "New Title" for a new document; the Subject field if the view title is "AuthorView"; or the Subject field plus the number of response documents otherwise.

```plaintext
StandardTitle := Subject + @DocDescendants(" (No Responses)"; " (1 Response)"; " (% Responses)")
@If(@isNewDoc; "New Topic"; @ViewTitle = "AuthorView"; Subject; StandardTitle)
```

3. This column formula displays the Subject field, the user name, and the number of response documents.

```plaintext
Subject + " (" + @Name([CN]; From) + @DocDescendants(""; ", % response"; ", % responses")
```

### Accessing the current document in the formula language

For form actions, buttons, hotspots, and field formulas, the current document is the one that’s open. For view actions, the current document is the one that’s highlighted (not checked). For agents, the current document is the one being acted on according to the build selection and \texttt{SELECT reserved word} criteria.

To read from a field in the current document, name the field or use the \texttt{@GetField function} which returns the value of the field whose name you specify. When you specify a field name, case does not matter, but the name must exactly match the field name. This function returns null if the value of the specified field is null or the specified field does not exist.

To write to a field in the current document, you must use the \texttt{FIELD reserved word} or the \texttt{@SetField function}. You cannot simply name the field.

- The \texttt{FIELD reserved word} is used in an assignment statement and has the following format. If you omit the \texttt{FIELD reserved word}, the assigned variable is treated as a temporary variable.

  \texttt{FIELD field-name := expression}
• @SetField writes to a field and has the same effect as an assignment statement using the FIELD reserved word. @SetField can be nested in another statement; you cannot do this with the FIELD reserved word. The field name is expressed as a text value, so it can be a variable as well as being the exact name in parentheses. One restriction is that @SetField works only on an existing field; if the field you want to write to does not exist in the document, "declare" it at the beginning of the formula by using it in a FIELD assignment. @SetField has the following format.

@SetField( field-expression-name; expression )

The DEFAULT reserved word provides a value in the event that a field is not in the document. If the field is available, its value is used. If the field is unavailable, the DEFAULT value is used.

DEFAULT field-name := expression

The @MailSend function mails a document. @MailSend without parameters mails the current document, which must contain a field named SendTo containing the recipients. @MailSend with parameters constructs the document to be mailed from the parameters.

@MailSend
@MailSend( to; cc; bcc; subject; body; fields; flags )

The @DeleteField function deletes a field. You specify it as the expression in a FIELD assignment.

FIELD field-name := @DeleteField

The @DocMark([NoUpdate]) function prevents formula changes to the document from being written to storage. The document is the same after processing by the formula as before. This @function only affects agents.

This table lists @functions that return document and field attributes.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@AllChildren</td>
<td>Selects immediate responses to matched documents; use only in a selection formula.</td>
</tr>
<tr>
<td>@AllDescendants</td>
<td>Selects all responses to matched documents; use only in a selection formula.</td>
</tr>
<tr>
<td>@AttachmentLengths</td>
<td>Returns the size in bytes of each attachment.</td>
</tr>
<tr>
<td>@AttachmentModifiedTimes</td>
<td>Returns the date on which the file attached to the current document was last modified.</td>
</tr>
<tr>
<td>@AttachmentNames</td>
<td>Returns the file names of all attached files.</td>
</tr>
<tr>
<td>@Attachments</td>
<td>Returns the number of attached files.</td>
</tr>
<tr>
<td>@Author</td>
<td>Returns the abbreviated names of all authors.</td>
</tr>
<tr>
<td>@DocFields</td>
<td>Returns the names of all the fields in the document.</td>
</tr>
<tr>
<td>@DocLength</td>
<td>Returns the size in bytes of the document.</td>
</tr>
<tr>
<td>@DocumentUniqueID</td>
<td>Returns the document’s unique ID, which is unique across all replicas of the document; in a field, creates a doclink to the current document.</td>
</tr>
<tr>
<td>@GetFocusTable</td>
<td>Returns the name, current row, or current column of the table that is in focus.</td>
</tr>
<tr>
<td>@InheritedDocumentUniqueID</td>
<td>Returns the unique ID of the parent document; in a field, creates a doclink to the current document.</td>
</tr>
<tr>
<td>@IsAvailable field</td>
<td>Returns True (1) if a field exists in the document.</td>
</tr>
<tr>
<td>@IsDocBeingEdited</td>
<td>Returns True (1) if the document is in Edit mode.</td>
</tr>
<tr>
<td>@IsDocBeingLoaded</td>
<td>Returns True (1) if the document is being loaded.</td>
</tr>
</tbody>
</table>
### Examples: Accessing the current document

1. This example of a computed field value performs an arithmetic operation involving two other fields in the document. These fields must exist in the document, must be numeric, and must be initialized to a numeric value.
   
   ```
   TotalSales - CostOfSales
   ```

2. This agent example performs an arithmetic operation on two fields in the current document, and assigns the result to a third field. The two referenced fields must exist; GrossSales can be new.

   ```
   FIELD GrossSales := TotalSales - CostOfSales;
   SELECT @All
   ```

3. This agent example performs an arithmetic operation on two fields in the current document, and either assigns the value to a third field or sends a mail message. The first statement initializes GrossSales and is not necessary if you are certain the field already exists.

   ```
   FIELD GrossSales := 0;
   gs := TotalSales - CostOfSales;
   @If(gs > 0; @SetField("GrossSales"; gs); @MailSend("Ian Perron"; ""; "; " ; "No gross sales"; "Gross sales are zero or less for "; Subject));
   SELECT @All
   ```

4. This column formula example evaluates to the value of KeyThought for documents that contain that field. If a document does not contain a KeyThought field, it "defaults" to the value of Topic.

   ```
   DEFAULT KeyThought := Topic;
   KeyThought
   ```

5. This is another way of coding the above example.

   ```
   @If(@IsAvailable(KeyThought); KeyThought; Topic)
   ```

6. This agent example deletes the GrossSales field.

   ```
   @If ( @IsUnavailable(GrossSales); @Return(""; ");
   FIELD GrossSales := @DeleteField;
   SELECT @All
   ```

7. This agent example calculates the GrossSales field, then displays the result and does not mark the document for update. As a result, no change takes place in the document in storage. The changes are saved if @DocMark is omitted or "@DocMark([Update])" is specified.

   ```
   FIELD GrossSales := TotalSales - CostOfSales;
   @Prompt([Ok]; "Gross sales for " + Subject; @Text(GrossSales));
   @DocMark([NoUpdate]);
   SELECT @All
   ```

8. This example displays all the fields in the current document.

   ```
   @Prompt([OkCancelList]; "Fields"; "Fields in document"; "; @DocFields);
   SELECT @All
   ```

---

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@IsDocBeingMailed</td>
<td>Returns True (1) if the document is being mailed.</td>
</tr>
<tr>
<td>@IsDocBeingRecalculated</td>
<td>Returns True (1) if the document is being recalculated.</td>
</tr>
<tr>
<td>@IsDocBeingSaved</td>
<td>Returns True (1) if the document is being saved.</td>
</tr>
<tr>
<td>@IsNewDoc</td>
<td>Returns True (1) if the document is not yet saved.</td>
</tr>
<tr>
<td>@IsResponseDoc</td>
<td>Returns True (1) if the document is a response.</td>
</tr>
<tr>
<td>@IsUnavailable(field)</td>
<td>Returns True (1) if a field does not exist in the document.</td>
</tr>
<tr>
<td>@NoteID</td>
<td>Returns &quot;NT&quot; followed by the note ID of the document.</td>
</tr>
<tr>
<td>@PolicyIsFieldLocked</td>
<td>Returns True (1) if the field is locked by an administration policy.</td>
</tr>
<tr>
<td>@Responses</td>
<td>Returns the number of responses to the current document in the current view.</td>
</tr>
</tbody>
</table>
9. This window title formula displays "New Document" for a new document. For an existing document, the formula displays the Subject field and the number of responses.
   \[\text{If}(\text{IsNewDoc}; \text{"New Document"}; \text{Subject} + \text{" with "} + \text{Text(Responses)} + \text{" response(s")})\]

10. This view selection formula selects all documents except those for which the Form field contains "Profile" or "Log."
    `SELECT !@Contains(Form; "Profile" ; "Log")`

11. This view selection formula selects all documents for which the Subject field contains "acme" in any case, plus all their descendants.
    `SELECT @Contains(@LowerCase(Subject); "acme") | @AllDescendants`

12. This form action formula displays the names and lengths of all attachments in a document, or "No attachments" if the document has no attachments.
    `@If(@Attachments > 0; @Prompt([OkCancel]; "Attachments"; "Attachment names and lengths"; "\"; @AttachmentNames + " (\" + @Text(@AttachmentLengths) + \" bytes\") ); @Prompt([Ok]; "Attachments"; "No attachments") )`

13. This onHelp event returns the name, row, and column of a table that is currently in focus.
    `row := @GetFocusTable([CellRow]);
    \text{If}(row = "0"); @Prompt([Ok]; "*No table*"; "Not in a table");
    \text{Do} (column := @GetFocusTable([CellColumn]);
    name0 := @GetFocusTable([TableName]);
    name := @If(name0 = "$" ; "No name"; name0);
    @Prompt([Ok]; "*" + name + "*";
    "Row * + row + ", column * + column") )

---

**Accessing data outside the current document and database**

The following @functions get data values from a specified database. You cannot set values with these @functions.

- **@[DbLookup]** looks up a specified value in the first sorted column of a specified view in a specified database. For each document that matches the search value, @DbLookup returns the value of a specified field on the document or column in the view.

- **@[DbColumn]** returns all the values in a specified column in a specified view in a specified database.

The first three parameters are the same for both functions:

- "Notes" : "NoCache" specifies that the operation is on a Domino database and is not cached. You can specify "" for the first list element because "Notes" is the default. If the data is stable and/or you are accessing the database many times, you can specify the second list element as "" to not use a cache. You can specify the entire parameter as "" to mean a Notes database and a cache. If you specify a cache, you can specify "ReCache" on subsequent lookups to the same data source to refresh the cache.

- **server : database** specifies the server and database you are accessing. Specify the first list element as "" to mean the local Domino directory. Specify the entire parameter as "" to mean the current database. You can specify the entire parameter as the replica ID of a database. Notes will search locally and on servers, and use the first replica it finds. Get the replica ID by choosing File - Database - Design Synopsis, Replication.

- **view** specifies the view through which to access the database.

For @DbLookup, the fourth parameter is the key, the value to search for in the first sorted column in the view. @DbLookup finds every document that matches the key value. Specify [PartialMatch] in the sixth parameter to match the key against the beginning characters of the column value rather than the entire column value.
For @DbLookup, the fifth parameter is either the name of a field in the database or the number of a column in the view. @DbLookup returns a list of the values in the fields or columns of the found documents. Specify [ReturnDocumentUniqueID] in the sixth parameter to return instead the UNID of the document.

For @DbColumn, the fourth parameter is the number of a column. @DbColumn returns a list of all values in the column.

The following @functions get and set field values in another document in the current database. However, you must know the unique ID of the document.

- **@GetDocField(unid; fieldName)** gets the value of a field given its unique ID.
- **@SetDocField(unid; fieldName; value)** sets the value of a field given its unique ID.

These functions are suited to accessing and setting values in the parent of a response document, since the unique ID of the parent is in the $Ref field of the child. In documents that have "Formulas inherit values from selected document" set, you can use @InheritedDocumentUniqueID as the formula in a hidden field in the base document, then use that field name as the formula in a hidden field in the inherited document. Otherwise, you must devise a scheme for storing and retrieving the unique IDs of the documents that you want to access.

The following @functions get and set field values in one or more profile documents in the current database. Profile documents are hidden documents used to store information that can then be shared across the database. You can create one profile document per user or one profile document for the entire database.

- **@GetProfileField** retrieves a field from a profile document and caches the field’s value for the remainder of the session.
- **@SetProfileField** sets the value of a field in a profile document.

**Examples: Accessing data outside the current document and database**

1. This example looks up a person name in the People view of the local Address Book (names.nsf) and gets the office phone number in the document containing that name.
   ```java
   inputName := @Prompt([OkCancelEdit]; "User name": "Enter user name as FIRST LAST": ");
   adjName := @Right(inputName; ") + ") + @Left(inputName; "");
   phoneNumber := @DbLookup("Notes": "NoCache": ": " "People": adjName: "OfficePhoneNumber";
   @Prompt([Ok]: "Office phone number": inputName + ")s office phone is ") + phoneNumber)
   ```
   2. This example is the same as the first, with two exceptions. The replica ID of an Address Book database is used instead of a specific name. Notes will search for the replica locally and then on servers, and use the first database it finds with that replica. The last parameter to @DbLookup is column 2 instead of the OfficePhoneNumber. Effectively this is the same, because column 2 contains the phone number.
   ```java
   inputName := @Prompt([OkCancelEdit]; "User name": "Enter user name as FIRST LAST": ");
   adjName := @Right(inputName; ") + ") + @Left(inputName; "");
   phoneNumber := @DbLookup("Notes": "NoCache": ": " "People": adjName: 2);
   @Prompt([Ok]: "Office phone number": inputName + ")s office phone is ") + phoneNumber)
   ```
   3. This example looks up the member names for a group in the Groups view of the local Address Book.
   ```java
   groupName := @Prompt([OkCancelEdit]; "Group name": "Enter group name": ");
   members := @DbLookup("Notes": "NoCache": ": " "Names": "Groups": groupName: "Members";
   @Prompt([OkCancelList]; "Group members": "Members of ") + groupName: "); members)
   ```
   4. This example gets the associated person name in the local Address Book, given an office phone number. It is designed to work where one number serves exactly one person. Since the database does not have a view sorted by phone number, the example uses @DbColumn to get all phone numbers (column 2) and all persons (column 1), then finds the person that corresponds to the number.
Accessing external databases through LS:DO using @functions

The following @functions access an external database through ODBC and return a value or list of values:

- **@DbColumn** returns all the values in one column of a table, or all the distinct values.
- **@DbLookup** returns selected values in one column of a table by matching keys.
- **@DbCommand** passes a command to an external DBMS and returns the result.

@DbColumn and @DbLookup can only retrieve data. They can’t add, delete, or modify data, or perform other operations. @DbCommand can retrieve data or send other SQL statements that can change data. LotusScript provides a wider range of capabilities, including the ability to update the external database.

The first four parameters are the same for all three @functions and establish access to the database through ODBC. The parameters are:

- "ODBC" as a string constant; or "ODBC" : "NoCache"
- Name of the data source as defined in the table of data sources (odbc.ini in Windows)
- User ID, list of two user IDs, or a null string, depending on the external data source
- Password, list of two passwords, or a null string, depending on the external data source
- (@DbColumn and @DbLookup) Name of the table to be accessed
- (@DbCommand) Command string to be executed
- (@DbColumn and @DbLookup) Name of the column to be accessed
- Option for handling null data returned by the data source
- (@DbLookup) Name of the column containing the key
- (@DbLookup) Value of the key as the appropriate data type, or a list
- (@DbColumn and @DbLookup) List of two elements: "Distinct" as a string argument or null string; "Ascending" or "Descending" as a string argument

Where user IDs and passwords are required, you can specify null strings and let the user supply them when the @function executes.

Examples: Accessing external databases through LS:DO using @functions

1. This formula gets the PARTNO column of the MANUALS table.
   ```LotusScript
   @DbColumn("ODBC";"Oracle";"";"";"MANUALS";"PARTNO";"";"";"Ascending")
   ```
2. This formula gets the TITLE from the row of the MANUALS table where PARTNO is 17-895A.
   ```LotusScript
   @DbLookup("ODBC";"Oracle";"";"";"MANUALS";"TITLE";"PARTNO";"17-895A")
   ```
3. This formula gets the PARTNO column value for every row of the MANUALS table, where the numeric value in the ONHAND column is less than 100.
   ```LotusScript
   @DbCommand("ODBC";"Oracle";"";"";"SELECT PARTNO FROM MANUALS WHERE ONHAND <100")
   ```
Chapter 6. Formula Language @Functions A-Z

This documentation shows the syntax and usage for all the @functions, in alphabetical order. It also includes examples, wherever appropriate.

To find the context in which a particular @function works, see the following topics:
• Where does this @function work (Part 1)
• Where does this @function work (Part 2)

For information on ECL security in the formula language, see the listing of @Functions with ECL security.

Where does this @function work? (Part 1 A -- D)

The following table lists each @function (A -- D) and indicates whether or not the @function works in the following contexts:
• Toolbar button formula
• Selection formula
• Column formula
• Agent (triggered by "Action menu or list selection")
• Agent (run "Before new mail arrives," "After new mail has arrived," "When documents are pasted," or "After documents are created or modified")
• Agent (run "On schedule")
• Hide-when formula
• Section editor formula

For a listing of the other @functions in Part 1, see the following:

"Where does this @function work? (Part 1 E -- K)"
"Where does this @function work? (Part 1 L -- R)"
"Where does this @function work? (Part 1 S -- Z)"

If you need to know whether an @function works in a different context, see Where does this @function work? (Part 2)."

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.

<table>
<thead>
<tr>
<th>@Abs</th>
<th>Toolbar button</th>
<th>Selection</th>
<th>Column</th>
<th>Agent manual</th>
<th>Agent mail paste modify</th>
<th>Agent scheduled</th>
<th>Hide-when</th>
<th>Section editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Abstract</td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste</td>
<td>Agent modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Accessed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ACos</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AddToFolder</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Adjust</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AdminECLIsLocked</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@All</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AllChildren</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AllDescendants</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Ascii</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ASin</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ATan</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ATan2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AttachmentLengths</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AttachmentModifiedTimes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AttachmentNames</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Attachments</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Author</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Begins</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@BrowserInfo</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@BusinessDays</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Certificate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Char</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@CheckAlarms</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
<td>Section editor</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>@CheckFormulaSyntax</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ClientType</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Command</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Compare</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ConfigFile</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Contains</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Cos</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Count</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Created</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Date</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Day</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DB2Schema</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbColumn (Domino)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbColumn (ODBC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbCommand (ODBC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbExists</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbLookup (Domino)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbLookup (ODBC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbManager</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbTitle</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DDEExecute</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DDEInitiate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DDEPoke</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste</td>
<td>Agent modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>-----------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>@DDETerminate</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DEFAULT</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DeleteDocument</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DeleteField</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DialogBox</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Do</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DocChildren</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DocDescendants</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DocFields</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DocLength</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DocLevel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DocLock</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DocMark</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DocNumber</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DocOmittedLength</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DocParentNumber</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DocSiblings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DocumentUniqueID</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Domain</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@DoWhile</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Where does this @function work? (Part 1 E -- K)**

The following table lists each @function (E -- K) and indicates whether or not the @function works in the following contexts:

- Toolbar button formula
- Selection formula
- Column formula
• Agent (triggered by "Action menu or list selection")
• Agent (run "Before new mail arrives," "After new mail has arrived," "When documents are pasted," or "After documents are created or modified")
• Agent (run "On schedule")
• Hide-when formula
• Section editor formula

For a listing of the other @functions in Part 1, see the following:

"Where does this @function work? (Part 1 A -- D)"

"Where does this @function work? (Part 1 L-- R)"

"Where does this @function work? (Part 1 S -- Z)"

If you need to know whether an @function works in a different context, see "Where does this @function work? (Part 2)."

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.

<table>
<thead>
<tr>
<th>@functions</th>
<th>Toolbar button</th>
<th>Selection</th>
<th>Column</th>
<th>Agent manual</th>
<th>Agent mail paste modify</th>
<th>Agent scheduled</th>
<th>Hide-when</th>
<th>Section editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>@EditECL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@EditUserECL</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Elements</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@EnableAlarms</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Ends</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Environment</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Error</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Eval</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Exp</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Expload</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Failure</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
<td>Section editor</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>@False</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@FileDir</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@FloatEq</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@FontList</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@For</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@FormLanguage</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetAddressBooks</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetCurrentTimeZone</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetDocField</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetField</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetFocusTable</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetHTTPHeader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetIMContactListGroupNames</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetPortsList</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetProfileField</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetViewInfo</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@HardDeleteDocument</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@HashPassword</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Hour</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@If</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IfError</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Implode</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@InheritedDocumentUniqueID</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste</td>
<td>Agent modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>@Integer</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsAgentEnabled</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsAppInstalled</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsAvailable</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsCategory</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IsDB2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsDocBeingEdited</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingLoaded</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingMailed</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@IsDocBeingRecalculated</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingSaved</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@IsDocTruncated</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsEmbeddedInsideWCT</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IsError</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsExpandable</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@IsMember</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsModalHelp</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsNewDoc</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@IsNotMember</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsNull</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsNumber</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsResponseDoc</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsText</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsTime</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@function</td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
<td>Section editor</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>@IsUnavailable</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsValid</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsVirtualizedDirectory</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Keywords</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Where does this @function work? (Part 1 L -- R)**

The following table lists each @function (L -- R) and indicates whether or not the @function works in the following contexts:

- Toolbar button formula
- Selection formula
- Column formula
- Agent (triggered by "Action menu or list selection")
- Agent (run "Before new mail arrives," "After new mail has arrived," "When documents are pasted," or "After documents are created or modified")
- Agent (run "On schedule")
- Hide-when formula
- Section editor formula

For a listing of the other @functions in Part 1, see the following:

"Where does this @function work? (Part 1 A -- D)"

"Where does this @function work? (Part 1 E -- K)"

"Where does this @function work? (Part 1 S -- Z)"

If you need to know whether an @function works in a different context, see "Where does this @function work? (Part 2)."

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.
<p>| @LaunchApp          | X | X | X | X | X | X | X | X | X |
| @LDAPServer        | X | X | X | X | X | X | X | X | X |
| @Left              | X | X | X | X | X | X | X | X | X |
| @LeftBack          | X | X | X | X | X | X | X | X | X |
| @Length            | X | X | X | X | X | X | X | X | X |
| @Like              | X | X | X | X | X | X | X | X | X |
| @Ln                | X | X | X | X | X | X | X | X | X |
| @Locale            | X | X | X | X | X | X | X | X | X |
| @Log               | X | X | X | X | X | X | X | X | X |
| @LowerCase         | X | X | X | X | X | X | X | X | X |
| @MailDbName        | X | X | X | X | X | X | X | X | X |
| @MailEncryptSavedPreference | X | X | X | X | X | X | X | X | X |
| @MailEncryptSentPreference | X | X | X | X | X | X | X | X | X |
| @MailSavePreference | X | X | X | X | X | X | X | X | X |
| @MailSend          | X | X | X | X | X | X | X | X | X |
| @MailSignPreference | X | X | X | X | X | X | X | X | X |
| @Matches           | X | X | X | X | X | X | X | X | X |
| @Max               | X | X | X | X | X | X | X | X | X |
| @Member            | X | X | X | X | X | X | X | X | X |
| @Middle            | X | X | X | X | X | X | X | X | X |
| @MiddleBack        | X | X | X | X | X | X | X | X | X |
| @Min               | X | X | X | X | X | X | X | X | X |
| @Minute            | X | X | X | X | X | X | X | X | X |
| @Modified          | X | X | X | X | X | X | X | X | X |</p>
<table>
<thead>
<tr>
<th>Toolbar button</th>
<th>Selection</th>
<th>Column</th>
<th>Agent manual</th>
<th>Agent mail paste</th>
<th>Agent modify</th>
<th>Agent scheduled</th>
<th>Hide when</th>
<th>Section editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Module</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Month</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Name</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@NameLookup</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Narrow</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@NewLine</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@No</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@NoteID</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Nothing</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Now</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@OptimizeMailAddress</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@OrgDir</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Password</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@PasswordQuality</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Pi</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@PickList</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Platform</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@PolicyIsFieldLocked</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@PostedCommand</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Power</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Prompt</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@ProperCase</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Random</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@RefreshECL</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
<td>Section editor</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>@RegQueryValue</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>REM</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Repeat</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Replace</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ReplaceSubstring</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ReplicaID</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Responses</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Return</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Right</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@RightBack</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Round</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Where does this @function work? (Part 1 S -- Z)**

The following table lists each @function (S -- Z) and indicates whether or not the @function works in the following contexts:

- Toolbar button formula
- Selection formula
- Column formula
- Agent (triggered by "Action menu or list selection")
- Agent (run "Before new mail arrives," "After new mail has arrived," "When documents are pasted," or "After documents are created or modified")
- Agent (run "On schedule")
- Hide-when formula
- Section editor formula

For a listing of the other @functions in Part 1, see the following:

"Where does this @function work? (Part 1 A -- D)"

"Where does this @function work? (Part 1 E -- K)"

"Where does this @function work? (Part 1 L -- R)"

If you need to know whether an @function works in a different context, see "Where does this @function work? (Part 2)"
An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.

<table>
<thead>
<tr>
<th>@function</th>
<th>Toolbar button</th>
<th>Selection</th>
<th>Column</th>
<th>Agent manual</th>
<th>Agent mail paste</th>
<th>Agent scheduled</th>
<th>Hide-when</th>
<th>Section editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Second</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SELECT</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Select</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ServerAccess</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ServerName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Set</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@SetDocField</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetEnvironment</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@SetField</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetHTTPHeader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetProfileField</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetTargetFrame</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetViewInfo</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Sign</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Sin</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Sort</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Soundex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Sqrt</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@StatusBar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Subset</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Success</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Toolbar button</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste</td>
<td>Agent modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>@Sum</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Tan</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TemplateVersion</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Text</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TextToNumber</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TextToTime</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ThisName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ThisValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Time</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TimeMerge</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TimeToTextInZone</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TimeZoneToText</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Today</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Tomorrow</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ToNumber</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ToTime</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Transform</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Trim</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@True</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Unavailable</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UndeleteDocument</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Unique</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@UpdateFormulaContext</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UpperCase</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Toolbar</td>
<td>Selection</td>
<td>Column</td>
<td>Agent manual</td>
<td>Agent mail paste</td>
<td>Agent modify</td>
<td>Agent scheduled</td>
<td>Hide-when</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------</td>
<td>------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>@URLDecode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@URLEncode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@URLGetHeader</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLHistory</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLOpen</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@URLQueryString</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@UserAccess</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@UserName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@UserNameLanguage</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@UserNamesList</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@UserPrivileges</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UserRoles</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@V2If</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@V3UserName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@V4UserAccess</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@ValidateInternetAddress</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@VerifyPassword</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Version</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ViewShowThisUnread</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ViewTitle</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@WebDBName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Weekday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@While</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Wide</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Where does this @function work? (Part 2 A -- D)

The following table lists each @function (A -- D) and indicates whether or not the @function works in the following contexts:

- Window title formula
- Hotspot formula (actions & buttons)
- Hotspot formula (formula pop-ups)
- Field formula
- Form formula
- Form action formula
- View action formula
- Navigator
- Layout region

For a listing of the other @functions in Part 2, see the following:

"Where does this @function work? (Part 2 E -- K)"

"Where does this @function work? (Part 2 L -- R)"

"Where does this @function work? (Part 2 S -- Z)"

If you need to know whether an @function works in a different context, see "Where does this @function work? (Part 1)."

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.
Note that some @functions return different values when the formula runs on a server.

<table>
<thead>
<tr>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Abs</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Abstract</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Accessed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ACos</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@addToFolder</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Adjust</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AdminECLIIsLocked</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@All</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Ascii</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ASin</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ATan</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ATan2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AttachmentLengths</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AttachmentModifiedTimes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@AttachmentNames</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Attachments</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Author</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Begins</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@BrowserInfo</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@BusinessDays</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Certificate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Char</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@CheckAlarms</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@CheckFormulaSyntax</td>
<td>Window title</td>
<td>Hotspot action &amp; button</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>------</td>
<td>------------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ClientType</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Command</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Compare</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ConfigFile</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Contains</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Cos</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Count</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Created</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Date</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Day</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DB2Schema</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbColumn (Domino)</td>
<td>X</td>
<td>X</td>
<td>X*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbColumn (ODBC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbCommand (ODBC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>*@DbCommand (ODBC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbExists</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbLookup (Domino)</td>
<td>X</td>
<td>X</td>
<td>X*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbLookup (ODBC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbManager</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DbTitle</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@DDEExecute</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
*In a Web Application, @DbCommand acts on an embedded view in a document when called from an action in that document.

Where does this @function work? (Part 2 E -- K)

The following table lists each @function (E -- K) and indicates whether or not the @function works in the following contexts:

- Window title formula
- Hotspot formula (actions & buttons)
- Hotspot formula (formula pop-ups)
• Field formula
• Form formula
• Form action formula
• View action formula
• Navigator
• Layout region

For a listing of the other @functions in Part 2, see the following:

"Where does this @function work? (Part 2 A -- D)"

"Where does this @function work? (Part 2 L -- R)"

"Where does this @function work? (Part 2 S -- Z)"

If you need to know whether an @function works in a different context, see “Where does this @function work? (Part 1).”

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.

<table>
<thead>
<tr>
<th>@EditECL</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@EditUserECL</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Elements</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Ends</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENVIRONMENT</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Environment</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Error</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Eval</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Exp</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Explode</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@Failure</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@False</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Field</td>
<td>Window title</td>
<td>Hotspot action &amp; button</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
<td>Layout region</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>-----</td>
<td>------------</td>
<td>------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>FIELD</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@FileDir</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@FloatEq</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@FontList</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@For</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@FormLanguage</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@GetAddressBooks</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@GetCurrentTimeZone</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@GetDocField</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@GetField</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@GetFocusTable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetHTTPHeader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetIMContactListGroupNames</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@GetPortsList</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@GetProfileField</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@GetViewInfo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@HashPassword</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Hour</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@If</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IfError</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Implode</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@InheritedDocumentUniqueID</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Integer</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@IsAgentEnabled</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Window title</td>
<td>Hotspot action &amp; button</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
<td>Layout region</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>@IsAppInstalled</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsAvailable</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDB2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingEdited</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingLoaded</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingMailed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingRecalculated</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDocBeingSaved</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsDocTruncated</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsEmbeddedInsideWCT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsError</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsMember</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsModalHelp</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsNewDoc</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsNotMember</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsNull</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsNumber</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsResponseDoc</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsText</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsTime</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsUnavailable</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsValid</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@IsVirtualizedDirectory</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Keywords</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Where does this @function work? (Part 2 L -- R)

The following table lists each @function (L -- R) and indicates whether or not the @function works in the following contexts:

- Window title formula
- Hotspot formula (actions & buttons)
- Hotspot formula (formula pop-ups)
- Field formula
- Form formula
- Form action formula
- View action formula
- Navigator
- Layout region

For a listing of the other @functions in Part 2, see the following:

"Where does this @function work? (Part 2 A -- D)"

"Where does this @function work? (Part 2 E -- K)"

"Where does this @function work? (Part 2 S -- Z)"

If you need to know whether an @function works in a different context, see "Where does this @function work? (Part 1)."

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.

<table>
<thead>
<tr>
<th></th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td>@LanguagePreference</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@LaunchApp</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@LDAPServer</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Left</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@LeftBack</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Length</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Like</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Ln</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Locale</td>
<td>Window title</td>
<td>Hotspot action &amp; button</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
<td>Layout region</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>@Log</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@LowerCase</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MailDbName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MailEncryptSavedPreference</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MailEncryptSentPreference</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MailSavePreference</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MailSend</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MailSignPreference</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Matches</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Max</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Member</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Middle</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@MiddleBack</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Min</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Minute</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Modified</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Modulo</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Month</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Name</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@NameLookup</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Narrow</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@NewLine</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@No</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Window title</td>
<td>Hotspot action &amp; button</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
<td>Layout region</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>@NoteID</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Nothing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Now</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@OptimizeMailAddress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@OrgDir</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Password</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@PasswordQuality</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Pi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@PickList</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Platform</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@PolicyIsFieldLocked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@PostedCommand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Power</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Prompt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ProperCase</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Random</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@RefreshECL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@RegQueryValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Repeat</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Replace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ReplaceSubString</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ReplicaID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Responses</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
### Where does this @function work? (Part 2 S -- Z)

The following table lists each @function (S -- Z) and indicates whether or not the @function works in the following contexts:

- Window title formula
- Hotspot formula (actions & buttons)
- Hotspot formula (formula pop-ups)
- Field formula
- Form formula
- Form action formula
- View action formula
- Navigator
- Layout region

For a listing of the other @functions in Part 2, see the following:

"Where does this @function work? (Part 2 A -- D)"

"Where does this @function work? (Part 2 E -- K)"

"Where does this @function work? (Part 2 L -- R)"

If you need to know whether an @function works in a different context, see "Where does this @function work? (Part 1)."

An X indicates that the @function works in that context. A blank cell indicates that the @function does not work in that context. An asterisk (*) indicates that there is a caveat associated with using an @function in a particular context.

Note that some @functions return different values when the formula runs on a server.
<table>
<thead>
<tr>
<th>Function</th>
<th>Window title</th>
<th>Hotspot action &amp; button</th>
<th>Hotspot formula pop-up</th>
<th>Field</th>
<th>Form</th>
<th>Form action</th>
<th>View action</th>
<th>Navigator</th>
<th>Layout region</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Select</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ServerAccess</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@ServerName</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Set</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@SetDocField</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@SetEnvironment</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@SetField</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetHTTPHeader</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@SetProfileField</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetTargetFrame</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@SetViewInfo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Sign</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Sin</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Sort</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Sqrt</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@StatusBar</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Subset</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Success</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>@Sum</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Tan</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TemplateVersion</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@Text</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>@TextToNumber</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Window title &amp; button</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
<td>Layout region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------</td>
<td>------------</td>
<td>-----------</td>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@TextToTime</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ThisName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ThisValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Time</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@TimeMerge</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@TimeToTextInZone</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@TimeZoneToText</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Today</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Tomorrow</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ToNumber</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ToTime</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Transform</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Trim</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@True</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Unavailable</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Unique</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UndeleteDocument</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UpdateFormulaContext</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UpperCase</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLEncode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLDecode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLHeaders</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLHistory</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@URLOpen</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Window title</td>
<td>Hotspot action</td>
<td>Hotspot formula pop-up</td>
<td>Field</td>
<td>Form</td>
<td>Form action</td>
<td>View action</td>
<td>Navigator</td>
<td>Layout region</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-----------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>@URLQueryString</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@UserAccess</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@UserName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@UserNameLanguage</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@UserNamesList</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@UserPrivileges</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@UserRoles</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@V2If</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@V3UserName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@V4UserAccess</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@ValidateInternetAddress</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@VerifyPassword</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Version</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@ViewShowThisUnread</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ViewTitle</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@WebDBName</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Weekday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@While</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Wide</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Word</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Year</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Yesterday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>@Zone</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
@Functions with ECL security

The following table lists the @functions affected by an execute control list (ECL). Those @functions do not execute on the workstation unless the marked ECL privileges are granted to the formula’s signer.

The ECL flags listed in the table are:
- Access to current database (cur)
- Access to environment variables (env)
- Access to non-Notes databases (db)
- Access to external programs (prog)
- Ability to send mail (mail)
- Access to Workstation Security ECL (ecl)
- Ability to read other databases (read)
- Ability to modify other databases (mod)

<table>
<thead>
<tr>
<th>@function</th>
<th>cur</th>
<th>env</th>
<th>db</th>
<th>prog</th>
<th>mail</th>
<th>ecl</th>
<th>read</th>
<th>mod</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DbColumn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@DbColumn(ODBC)</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DbCommand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@DbLookup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>@DbLookup(ODBC)</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DDEExecute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@DDEInitiate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@DDEPoke</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@DDETerminate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@DeleteDocument</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@DeleteField</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@EditECL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@EditUserECL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@Environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>@GetProfileField</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@MailSend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
@Abs

Returns the absolute (unsigned) value of a number.

Syntax

@Abs( anyNumber )

Parameters

anyNumber

Number. Any number valid in Lotus Notes/Domino, whether positive or negative, whole or fractional, integer or real.

Return value

absoluteValue

Number. The absolute value of anyNumber.

Usage

When you use this function as the Input translation formula for a Number field, you do not have to supply the field with a default value.

You can enter a field name as the anyNumber parameter. If you do, be sure that the field you reference in an @Abs function:

- Is a number field
- Has a default value of zero

Language cross-reference

Abs function in LotusScript language
Examples: @Abs

1. This example returns 2.16.
   @Abs(–2.16)
2. This example returns 2.16 if the number in the field named Net is either 2.16 or –2.16.
   @Abs(Net)
3. This example returns 25 if Score1 = 50 and Score2 = 75, or if Score1 = 75 and Score2 = 50.
   @Abs(Score1 - Score2)
4. This formula, for a computed number field called numDays, uses @Abs to calculate the number of
days between two dates, which are stored in time fields dateA and dateB. @Integer(dateA-dateB)
returns the number of seconds between dateA and dateB, so the formula divides by 60*60*24 to get
days. For example, if dateA is 08/11/95 and dateB is 09/22/95, the formula returns: 42.
   @If( numDays = ""; 0; @Abs( @Integer( dateA - dateB ) / (60 * 60 * 24 ) ) )

@Abstract

Abbreviates the contents of one or more fields by:

- Selecting the most significant words in a body of text
- Abbreviating common words
- Dropping vowels from words
- Removing unnecessary text or characters, such as mail headers or white space

This function only works with single-byte character sets.

Syntax

@Abstract( [keywords]; size; beginText; bodyFields )

Parameters

[ [keywords] ]

Any number of keywords that tell Lotus Notes/Domino how you want to abbreviate and sort the text
(see list below). Keywords are executed in the order in which you list them. Enclose each keyword in
brackets and separate multiple commands with colons: [DROPVOWELS];[NOTRIMWHITE];[ABBREV].

size

Number. The maximum size of the abstracted text. Can be no larger than 64,994 bytes. The number of
bytes available for the abstracted text is size - 1; one byte is reserved for internal use.

beginText

Text. A comment to insert at the beginning of the returned text, no larger than 10 characters. The size of
beginText counts toward the total size of the abstracted text, but its contents are unaffected by @Abstract
commands. Specify an empty string ("") if you do not want a comment.

bodyFields

Text or text list. Any number of fields containing the text to abstract. May be text, rich text, or keyword
fields. The text within each field is concatenated with spaces in the order specified. If Notes/Domino
cannot locate a field by name, it uses the string literal instead. Enclose each field name in quotes and
separate multiple names with colons: "Sales";"Figures."
CAUTION:
Rich text fields are not part of a document until saved. If you want @Abstract to work on additions and changes to the current document, you must first save and then recalculate the document. @Abstract cannot convert rich text to text in a view column.

Return value

abstractedText

Text. The text contained in each of the body fields, abbreviated and sorted as specified by the commands.

Keywords

You can use the following keywords with @Abstract:

[TXTONLY]

Removes mail headers and punctuation chunks from the text.

[CNTWORDS]

Computes the significance of each word in the text. A word’s significance depends on the number of times it appears in the text. A word that appears in the Significant Word file (see ”Files,” below) gets its significance boosted. A word that appears in the Insignificant Word file (see ”Files,” below) has no significance.

[SAVE]

Saves the text in its current state. Saved text can be restored with the [RESTORE] keyword.

[RESTORE]

Discards the current text and restores the last-saved text. You can only restore saved text one time. If no text has been previously saved, this keyword has no effect.

[TRYFIT]

Takes the current text and determines if it has become small enough to fit in the specified size. If so, @Abstract returns the current text and stops, ignoring any remaining commands. If not, @Abstract continues with the next keyword.

[SORTCHUNKS]

Sorts the text according to significance. The text is divided into chunks, of which there are three types: text, mail header, and punctuation.
- Text chunks are usually sentences. They may be at the beginning, end, or middle of a paragraph.
- Mail header chunks are created according to the contents of the Mail Headers file (see ”Files” below).
- Punctuation chunks consist of any text with no letters or digits.

The significance of a chunk depends upon the significances of the words within it, the number of words in the chunk, and the type and position of the chunk. To use [SORTCHUNKS], you must also use [CNTWORDS].

[ABBREV]
Abbreviates the text. @Abstract uses an Abbreviation Dictionary to substitute abbreviations for words in the text (see “Files,” below). You can control other aspects of the abbreviation process with the following commands (which have no effect unless followed by the [ABBREV] keyword):

[USEDICT]

Specifies that the Abbreviation Dictionary should be used. This is the default.

[NODICT]

Specifies that the Abbreviation Dictionary should not be used.

[KEEPVOWELS]

Keeps vowels in words. This is the default.

[DROPVOWELS]

Removes vowels from words. The first vowel in a word that begins with a vowel isn’t affected. If you use [DROPVOWELS], you can optionally use one of the following subcommands.

[DROPFIRSTVOWEL]

Drops vowels from the beginning of words.

[KEEPFIRSTVOWEL]

Keeps vowels at the beginning of words. This is the default.

[TRIMWHITE]

Removes extra white space characters from the text. This is the default.

[NOTRIMWHITE]

Retains extra white space characters in the text.

[TRIMPUNCT]

Removes extra white space characters surrounding punctuation.

[NOTRIMPUNCT]

Retains extra white space characters surrounding punctuation.

[NOSTOPLIST]

Disables the insignificant word list (notestop.txt)

[NOSIGLIST]

Disables the significant word list (notesigl.txt).
Rules
There are three built-in programs you can use with @Abstract.

[RULE1] consists of the following commands, executed in this order:

[TEXTONLY]:[TRYFIT]
Removes all mail header and punctuation chunks. If the text fits, the function is done; otherwise, continues.

[TRIMPUNCT]:
Trims white space around punctuation marks.

[SAVE]:
Saves the current state of the text.

[ABBREV]:[TRYFIT]:
Abbreviates the text. If the text fits, the function is done; otherwise, continues.

[RESTORE]:
Restores the state of the text to what it was prior to abbreviating.

[SAVE]:
Saves the current state of the text.

[DROPVOWELS]:[ABBREV]:[TRYFIT]:
Abbreviates text by dropping vowels. If the text fits, the function is done; otherwise, continues.

[RESTORE]:
Restores the state of the text to what it was prior to abbreviating.

[COUNTWORDS]:[SORTCHUNKS]:[ABBREV]
Counts words and sorts the chunks. Abbreviates the text and returns it.

If the removal of mail headers and punctuation allowed the text to fit into the desired size, then text is returned as is. If the first abbreviation was enough to make the text fit, the returned text begins with a minus character ( - ). If the second abbreviation was enough to make the text fit, the returned text begins with a plus character ( + ). If the function counted the words and sorted the chunks, the text will start with an asterisk ( * ) and the sentences will be separated with a ( | ) to indicate that they were rearranged.

[RULE2] issues the following commands:

[TRIMPUNCT]:[ABBREV]

[RULE3] issues the following commands:
[TEXTONLY]:[TRYFIT]:
Removes all mail header and punctuation chunks. If the text fits, the function is done; otherwise, continue.

[TRIMPUNCT]:
Trims white space around punctuation marks.

[SAVE]:
Saves the current state of the text.

[ABBREV]:[TRYFIT]:
Abbreviates the text. If the text fits, the function is done; otherwise, continue.

[RESTORE]:
Restores the state of the text to what it was prior to abbreviating.

[DROPVOWELS]:
Abbreviates text by dropping vowels.

[SAVE]:
Saves the current state of the text.

[ABBREV]:[TRYFIT]:
If the text fits, the function is done; otherwise, continue.

[RESTORE]:
Restores the state of the text to what it was prior to abbreviating.

[COUNTWORDS]:[SORTCHUNKS]:[ABBREV]
Counts words and sorts the chunks. Abbreviates the text and returns it.

If the function counted the words and sorted the chunks, the returned text begins with an asterisk ( * ) and the sentences are separated with a ( | ) to indicate that they were rearranged.

Files
The following files are used by @Abstract. You can create all, some, or none of these text files, depending on how you want to use @Abstract. Any files you do create must be named as specified below and be present in your Notes/Data file path when you start running the software.

Abbreviation Dictionary (noteabbr.txt)
Each line of the file should contain two words: the first is the original word and the second is its abbreviation. An abbreviation must be shorter than the word it replaces. For example:

telephone ph
number no
Capitalization works as follows:
- If the abbreviation is specified in uppercase letters, then it always appears in uppercase letters.
- If the original word appears in lowercase letters, the abbreviation appears as specified in the abbreviation dictionary.
- If the original word appears in uppercase letters or in a mixture of uppercase and lowercase letters, the abbreviation appears in uppercase letters.
- A lowercase first letter in the abbreviation will be converted to uppercase if needed to match the first letter in the original word.
- The remaining letters in the abbreviation will be converted to uppercase if needed to match the case of the original word’s second letter.

The abbreviation is never converted to lowercase, but it may be converted to uppercase.

<table>
<thead>
<tr>
<th>Specified abbreviation</th>
<th>Word being replaced</th>
<th>Resulting abbreviation</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>telephone</td>
<td>Phone</td>
<td>The original word appears in lowercase, so the specified abbreviation’s case is used.</td>
</tr>
<tr>
<td>Phone</td>
<td>TElephone</td>
<td>PHONE</td>
<td>The abbreviation’s case is based upon the original word’s case.</td>
</tr>
<tr>
<td>Phone</td>
<td>Telephone</td>
<td>Phone</td>
<td>The abbreviation’s case is based upon the original word’s case.</td>
</tr>
<tr>
<td>PHONE</td>
<td>Telephone</td>
<td>PHONE</td>
<td>The abbreviation is specified as uppercase, so it always appears as uppercase.</td>
</tr>
<tr>
<td>Phone</td>
<td>tElephone</td>
<td>PHONE</td>
<td>The first letter of the abbreviation was already uppercase, so Notes/Domino leaves it alone. The remaining letters of the abbreviation are converted to uppercase to match the second letter of the original word.</td>
</tr>
</tbody>
</table>

Significant Words (notesig1.txt)

The file should be a free-form list of significant words, such as "urgent" or "immediately." When @Abstract computes the significance of text, it boosts the significance of any words included in the significant word list. For example:

client
boss
chocolate

Insignificant Words (notestop.txt)

The file should be a free-form list of words that are always insignificant, such as "the," "and," and "of." When @Abstract computes the significance of words, it ignores any words included in this file. For example:
	he
and
of

Mail Headers (notehead.txt)

The file should be a free-form list of words that indicate mail headers, such as Subject, From, and To. In order for @Abstract to consider a chunk a mail header, it must begin with one of the words specified in
this file and be followed immediately by a colon and a space. If you want a mail header to be considered significant, place an asterisk after the word. For example:

```
Subject*
From
```

**Language cross-reference**

Abstract method in LotusScript NotesItem class

abstractText method in Java Item class

GetFormattedText method in LotusScript NotesRichTextItem class

getFormattedText method in Java RichTextItem class

**Examples: @Abstract**

1. This formula abbreviates the contents of the description field by eliminating vowels.
   ```
   @Abstract( [DROPVOWELS]:[ABBREV]; 200; ""; "description" )
   
   If the description field contained this text: The kickoff meeting for our capital campaign is tomorrow.
   Then the formula returns: Th kckff mtng fr or cptl cmpgn is tmrrw.
   ```

2. This formula abbreviates the contents of the description field by using an Abbreviation Dictionary and eliminating vowels, including the vowels that appear as the first letter in a word.
   ```
   @Abstract([USEDICT]:[DROPVOWELS]:[DROPFIRSTVOWEL]:[ABBREV]; 200; ""; "description" )
   
   If the Abbreviation Dictionary contains the following:
capital cap meeting mtg tomorrow tom
   Then the formula returns:Th kckff mtg fr r cap cmpgn s tom.
   ```

3. This formula shows a misunderstanding in the use of @Abstract. It returns the contents of the description field unaltered, since the [ABBREV] keyword incorrectly precedes [DROPVOWELS].
   ```
   @Abstract( [ABBREV]:[DROPVOWELS]; 200; ""; "description" )
   ```

4. This formula removes the white spaces from around all punctuation and abbreviates the text in the “opinion” field according to the noteabbr.txt file, which contains the following:
   ```
   following flwg punctuation punc
   @Abstract([RULE2];300;"Result:";"opinion")
   
   If the opinion field contains the text: The FOllowing is a list of punctuation marks: ! , ; .
   Then the formula returns: Result:The FLWG is a list of punc marks!:<;:.
   ```

**@Accessed**

Indicates the time and date when the document was last accessed by a Lotus Notes client, whether for reading or editing.

**Syntax**

```
@Accessed
```

**Return value**

```
lastAccessed
```

Time-date. The time and date that the current document was last accessed.
Usage
@Accessed is most useful in field formulas, selection formulas, agents, and actions. Because @Accessed requires some time to compute, it should not be used in applications where efficiency is critical.

The value returned by @Accessed is exact only to the day, not the hour. If the document is edited, the property is always updated. If the document is read more than once during the same 24–hour period, the value is only updated the first time accessed.

The last–accessed value is not replicated; each replica copy of the document maintains its own value. The value returned by @Accessed represents the last time the document was accessed in that replica of the database.

If the database is stored on CD-ROM, @Accessed has no meaning because read/write access is not controlled by the Notes/Domino editor.

Usage in workflow applications
This function is useful for determining whether a document has been “stalled” in a workflow application; for example, you can run an agent that checks the last–accessed date on a series of documents and sends out reminders about documents that should have been read but have not.

@Accessed can also be used in an agent to determine which documents in a database have not been accessed within a certain period of time, and archive them.

Note: @Accessed is similar to @Modified, which records the date the document was last edited and saved.

Usage in column or selection formulas
Be careful when using @Accessed in views (in column or selection formulas) because it forces the view to be refreshed every time it’s opened. You can prevent this by selecting the Manual/Background option for the view refresh frequency. Using @Accessed in a view will also cause that view to perpetually appear to need refreshing -- the refresh mark will always display in the corner.

Language cross-reference
LastAccessed property in LotusScript NotesDocument class

LastAccessed property in Java Document class

Examples: @Accessed
This formula returns: 06/22/95 10:46:03 AM; if the document was last read or edited on June 22, 1995 at 10:46:03 AM.
@Accessed

@ACos
Calculates the arc (inverse) cosine, using the cosine of an angle.

Syntax
@ACos( cosine )
**Parameters**

*cosine*

Number. A cosine of an angle, from -1 through 1.

**Return value**

*angle*

Number. An angle, in radians, from 0 through pi. This represents an angle between 0 and 180 degrees.

**Language cross-reference**

ACos function in LotusScript language

**Examples: @ACos**

1. This example returns pi/2.
   ```plaintext
   @ACos( 0 )
   ```
2. This example returns 1.0472 radians (60 degrees).
   ```plaintext
   @ACos( 0.5 )
   ```

**@AddToFolder**

Adds current document to one folder while removing it from another. NULL string can be substituted for either argument to skip the action.

**Note:** This @function is new with Release 5.

**Syntax**

```plaintext
@AddToFolder(foldernameadd ; foldernameremove)
```

**Parameters**

*foldernameadd*

Text. Name of the folder the document will be added to.

*foldernameremove*

Text. Name of the folder the document will be removed from.

**Usage**

This formula can be used in toolbar button and agent formulas.

```plaintext
@Command([Folder]; Foldername; MoveOrCopy) works just like @AddToFolder except it moves a document from the current folder.
```

**Language cross-reference**

Folder method in LotusScript NotesUIWorkspace class

PutInFolder method in LotusScript NotesDocument class

putInFolder method in Java Document class
RemoveFromFolder method in LotusScript NotesDocument class

removeFromFolder method in Java Document class

**Examples: **@AddToFolder

1. This example adds the currently selected document to the folder named Work.
   @AddToFolder("Work";"")

2. This example adds the currently selected document to the folder named Work and removes it from the folder named Favorites.
   @AddToFolder("Work";"Favorites")

@Adjust

Adjusts the specified time-date value by the number of years, months, days, hours, minutes, and/or seconds you specify. The amount of adjustment may be positive or negative.

**Syntax**

@Adjust( dateToAdjust ; years ; months ; days ; hours ; minutes ; seconds ; [DST] )

**Parameters**

dateToAdjust

Time-date. The time-date value you want to increment. This should be a single value, not a range.

years

Number. The number of years to increment by.

months

Number. The number of months to increment by.

days

Number. The number of days to increment by.

hours

Number. The number of hours to increment by.

minutes

Number. The number of minutes to increment by.

seconds

Number. The number of seconds to increment by.

[DST]

Keyword. Optional. Specify [INLOCALTIME] to further adjust the time for daylight-saving time if the adjustment crosses the boundary and daylight-saving time is in effect. Specify [INGMT] or omit this parameter to not further adjust the time for daylight-saving time. The adjustment is such that adding or subtracting in day increments yields the same time in the new day.
Return value

adjustedDate

Time-date. The date, incremented by the amount of time you have specified.

Usage

You must include all arguments except the [DST] keyword; include a zero (0) for parameters you don’t want to adjust.

Tip: To find the difference between two dates, subtract them. The result is returned in seconds. To adjust the result to days, divide the result by 86,400 - which is the number of seconds in a day. For example, if you have two date fields, date1, which contains [07/01/01] and date2, which contains [07/05/01], use the following formula to return the number of days between the two dates:

\[(date2-date1)/86400\]

This code returns 4.

Calculating due dates

A typical use for @Adjust is calculating a due date from an entry date, by adjusting only one component of the time-date value, for example, the month component.

Language cross-reference

AdjustYear method in LotusScript NotesDateTime class

adjustYear method in Java DateTime class

AdjustMonth method in LotusScript NotesDateTime class

adjustMonth method in Java DateTime class

AdjustDay method in LotusScript NotesDateTime class

adjustDay method in Java DateTime class

AdjustHour method in LotusScript NotesDateTime class

adjustHour method in Java DateTime class

AdjustMinute method in LotusScript NotesDateTime class

adjustMinute method in Java DateTime class

AdjustSecond method in LotusScript NotesDateTime class

adjustSecond method in Java DateTime class

Examples: @Adjust

1. This example returns 09/2/97.

@Adjust([08/30/95];2;2;0;0;0)

Lotus Notes/Domino sees 30 in the days portion of the time-date value and adjusts it by 2, which increments the month value by 1. Lotus Notes/Domino then adjusts the month value by 2, and the year value by 2.

2. This example returns 03/20/94.
@Adjust([03/30/96];–2;0;–10;0;0;0)
Notes/Domino returns a date that is 2 years and 10 days before the supplied date.

3. This example returns the date one month from the date in the field named Date.
   @Adjust(Date;0;1;0;0;0;0)

4. This example returns the date one month and one day from the current time-date.
   @Adjust(@Now;0;1;1;0;0;0)

5. Given a date, this formula calculates the beginning of the week. It takes the date stored in the
dueDate field, and returns the date representing the previous Monday. For example, if dueDate is
06/02/95, this formula returns 05/29/95.
   @Adjust( dueDate; 0; 0; - ( @Weekday( dueDate ) - 2 ); 0; 0; 0 )

@AdminECLIsLocked

Note: This @function is new with Release 7.

Checks the current status of the Administration ECL in the name and address book and returns 1 (True)
if the Administration ECL is locked and editing is prevented; otherwise returns 0 (False).

Syntax
@AdminECLIsLocked

Return value
flag

Boolean
• 1 (True) indicates that the Administration ECL is locked and may not be edited
• 0 (False) indicates that the Administration ECL is not locked

Usage
You cannot use this function in Web applications. In Release 7, this function will be used in pubnames.ntf
to determine if AdminECL is locked or not.

@All

Returns the value True.

Syntax
@All

Return value
flag

Number. The number 1 (True).

Usage
Use @All in selection formulas, mail agents, paste agents, scheduled agents, or in any formula requiring a
SELECT statement. Lotus Notes/Domino appends SELECT @All to agents in contexts where @All is
needed. All views default to a selection formula of SELECT @All.
Examples: @All

1. This example selects all documents in the database when used as a view selection formula.
   \[
   \text{SELECT } \@\text{All}
   \]

2. This formula, when used in a mail or paste agent, selects all documents and sets the Status field to "Open."
   \[
   \text{FIELD Status:="Open";SELECT@All}
   \]

@AllChildren

Includes all response documents at all levels for parent documents that match selection criteria.

Syntax

\[
\text{SELECT } \text{selectionFormula I @AllChildren}
\]

Return value

Selects all the documents that match \text{selectionFormula} plus their immediate responses.

Usage

@AllChildren can only be used in a view selection or selective replication formula. It must be appended to the end of a selection formula using the Boolean OR operator (" | "). Don’t use it within complex expressions in a formula.

@AllChildren allows you to define a view as a set of documents that match a given formula plus the immediate responses to those documents. It also allows you to create a selective replication formula to replicate a set of documents along with the immediate responses.

Selection formulas that use @AllChildren may provide a significant advantage over formulas that use @IsResponseDoc. While @IsResponseDoc returns True for any response document in a database, @AllChildren returns only those responses that are immediate children of matching documents.

@AllDescendants

Includes all response and response-to-response documents for parents that match selection criteria.

Syntax

\[
\text{SELECT } \text{selectionFormula I @AllDescendants}
\]

Return value

Selects all the documents that match \text{selectionFormula} plus their responses and responses-to-responses, for as many levels of response documents as exist.

Usage

@AllDescendants can only be used in a view selection or selective replication formula. It must be appended to the end of a selection formula using the Boolean OR operator (" | "). Don’t use it within complex expressions in a formula.

@AllDescendants allows you to define a view as a set of documents that match a given formula plus all the responses to those documents, at any level. It also allows you to create a selective replication formula to replicate a set of documents along with all responses.
Selection formulas that use @AllDescendants may provide a significant advantage to formulas that use @IsResponseDoc. While @IsResponseDoc returns True for any response document in a database, @AllDescendants returns only those responses that are descendants of matching documents.

Examples: @AllChildren and @AllDescendants

1. A response hierarchy contains the following documents.

   1.0 What is your favorite color? (Esteban Garcia) 1.1 Blue (Mary Lu) 1.2 Aqua (Jim Thompson) 1.2.1 Why do you like aqua? (Mary Lu) 1.2.2 It reminds me of the ocean (Jim Thompson) 1.3 I like the color orange (Bill Jones)

   The first SELECT statement selects documents 1.2, 1.2.1, and 1.2.2; the second selects documents 1.0, 1.1, 1.2, and 1.3; the third selects documents 1.0, 1.1, 1.2, 1.2.1, 1.2.2, and 1.3; and the fourth selects documents 1.2.1 and 1.3.

   SELECT @Author = "Jim Thompson" | @AllChildren
   SELECT @Author = "Esteban Garcia" | @AllChildren
   SELECT @Author = "Esteban Garcia" | @AllDescendants
   SELECT @Contains( Subject; "like" ) | @AllChildren

2. You have a Flowers discussion database and you want to add a new view that will show only those documents having to do with orchids. You create an Orchid view, use the View InfoBox to indicate that it should show documents in a response hierarchy, and write the following selection formula for the view:

   SELECT @Contains( Subject; "orchid" ) | @IsResponseDoc
   
   You get this view:

   **Date Topic**04/08/95 The orchid family of flowers (Anne Davis, 2 responses) 04/08/95 Sighting of new variation (Brad Sullivan) 04/08/95 The "ghost" orchid (Rachel Greenbaum) 04/08/95 Local flower shops that carry orchids (Mary Tsen, 1 response) 04/08/95 Try the Blumenhaus (Anne Davis)

   The view, however, is selecting every response document in the entire database, whether or not it has to do with orchids. For example, here's what the view looks like when the response hierarchy is turned off:

   **Date Topic**04/08/95 The orchid family of flowers (Anne Davis) 04/08/95 Sighting of new variation (Brad Sullivan) 04/08/95 Special varieties of roses (Michael Bowling) 04/08/95 My roses bloomed late this year (Marcel DuBois) 04/08/95 Local flower shops that carry orchids (Mary Tsen) 04/08/95 Try the Blumenhaus (Anne Davis) 04/08/95 The "ghost" orchid (Rachel Greenbaum)

   The unneeded documents take up valuable space in the view index on the database server. (In addition, if you used this same formula for replication, the unneeded documents would be replicated).

   You use @AllChildren to rewrite the selection formula:

   SELECT @Contains( Subject; "orchid" ) | @AllChildren

   This formula selects and displays only those response documents whose parent contains "orchid" in the Subject field. The view does not contain any hidden response documents.

   **Date Topic**04/08/95 The orchid family of flowers (Anne Davis, 2 responses) 04/08/95 Sighting of new variation (Brad Sullivan) 04/08/95 The "ghost" orchid (Rachel Greenbaum) 04/08/95 Local flower shops that carry orchids (Mary Tsen, 1 response) 04/08/95 Try the Blumenhaus (Anne Davis)

3. Just as you’d hoped, the orchids generate a lively discussion. The Main View of the database, which selects all documents, now looks like this:

   **Date Topic**04/08/95 The orchid family of flowers (Anne Davis, 7 responses) 04/08/95 Sighting of new variation (Brad Sullivan, 2 responses) 04/08/95 What color? (Anne Davis) 04/08/95 Please post exact location (Mary Tsen) 04/08/95 The "ghost" orchid (Rachel Greenbaum, 3 responses) 04/08/95 Very difficult to see (Brad Sullivan, 1 response) 04/08/95 Only blooms for an hour or so! (Rachel Greenbaum) 04/08/95 Some sightings reported in Florida (Anne Davis) 04/08/95 Roses beginning to bloom (Peter Donovan, 2 responses) 04/08/95 Special varieties of roses (Michael Bowling) 04/08/95
My roses bloomed late this year (Marcel DuBois) 04/08/95 Local flower shops that carry orchids (Mary Tsen, 1 response) 04/08/95 Try the Blumenhaus (Anne Davis) 04/08/95 Tulip trips to Holland (Mary Tsen)

The Orchid view you just created, however, does not contain all the documents you want. @AllChildren only selects the immediate children of any parent document(s) that meet the selection criteria:

**Date Topic**04/08/95 The orchid family of flowers (Anne Davis, 4 responses) 04/08/95 Sighting of new variation (Brad Sullivan) 04/08/95 The "ghost" orchid (Rachel Greenbaum, 2 responses) 04/08/95 Very difficult to see (Brad Sullivan) 04/08/95 Some sightings reported in Florida (Anne Davis) 04/08/95 Local flower shops that carry orchids (Mary Tsen, 1 response) 04/08/95 Try the Blumenhaus (Anne Davis)

In this case, @AllDescendants might provide a better solution. You rewrite the selection formula:

```
SELECT @Contains( Subject; "orchid" ) | @AllDescendants
```

The Orchid view now contains entire threads of the orchid discussion:

**Date Topic**04/08/95 The orchid family of flowers (Anne Davis, 7 responses) 04/08/95 Sighting of new variation (Brad Sullivan, 2 responses) 04/08/95 What color? (Anne Davis) 04/08/95 Please post exact location (Mary Tsen) 04/08/95 The "ghost" orchid (Rachel Greenbaum, 3 responses) 04/08/95 Very difficult to see (Brad Sullivan, 1 response) 04/08/95 Only blooms for an hour or so! (Rachel Greenbaum) 04/08/95 Some sightings reported in Florida (Anne Davis) 04/08/95 Local flower shops that carry orchids (Mary Tsen, 1 response) 04/08/95 Try the Blumenhaus (Anne Davis)

---

**@Ascii**

Converts an LMBCS (Lotus Multi-Byte Character Set) string to an ASCII string.

**Syntax**

```
@Ascii( string ) @Ascii( string ; [ALLINRANGE] )
```

**Parameters**

`string`

Text or text-list. An LMBCS string.

`[ALLINRANGE]`

Keyword. Optional. Specifies that @Ascii should return a null string ("") if any characters in the original string cannot be represented by ASCII codes 32 to 127.

**Return value**

`newString`

Text or text-list. The original string, with each character converted to an ASCII-compliant character. Any character that can’t be represented by ASCII codes 32 to 127 is replaced with a question mark (?). If you specify [ALLINRANGE] and there are characters that can’t be represented by ASCII codes 32 to 127, returns a null string ("").

**Usage**

@Ascii first converts the string into ASCII-compliant characters, replacing any unrepresented characters with question marks, and then, if [ALLINRANGE] is True, checks for question marks within the string. This means that if the original string contains a question mark and [ALLINRANGE] is specified, a null string is returned even if the entire string can be represented by ASCII codes 32-127.
Examples: @Ascii

1. This example returns Cue.
   ```plaintext
   @Ascii( "Çüé" )
   ```

2. This example returns Cue??.
   ```plaintext
   @Ascii( "ÇüéY" )
   ```

3. This example returns a null string ("") since the last 2 characters can't be represented by ASCII codes 32 to 127.
   ```plaintext
   @Ascii("ÇüéY";[ALLINRANGE])
   ```

4. This example returns Cue??; cat if field1 is a field containing the text list "ÇüéY";"cat."
   ```plaintext
   @Ascii( field1 )
   ```

---

@ASin

Calculates the arc (inverse) sine using the sine of an angle.

Syntax

```plaintext
@ASin( sine )
``` 

Parameters

sine

Number. A sine of an angle, from -1 through 1.

Return value

angle

Number. An angle, in radians, from -pi/2 through pi/2. This represents an angle between -90 and 90 degrees.

Language cross-reference

ASin function in LotusScript language

Examples: @ASin

1. This example returns pi/2.
   ```plaintext
   @ASin( 1 )
   ```

2. This example returns 0.72082 radians (41.3 degrees).
   ```plaintext
   @ASin ( 0.66 )
   ```

---

@ATan

Calculates the arc (inverse) tangent using the tangent of an angle.

Syntax

```plaintext
@ATan( tangent )
``` 

Parameters

tangent

Number. The tangent of an angle.
Return value

angle

Number. An angle, in radians, from -pi/2 through pi/2. This represents an angle between -90 and 90 degrees.

Language cross-reference

ATn function in LotusScript language

Examples: @ATan

1. This example returns pi/4.
   
   @ATan( 1 )

2. This example returns -pi/4.
   
   @ATan( -1 )

3. This example returns 1.10715 radians (63.4 degrees).
   
   @ATan( 2 )

@ATan2

Calculates the arc tangent using the tangent y/x of an angle.

Syntax

@ATan2( x ; y )

Parameters

x

Number. The denominator of the tangent value y / x.

y

Number. The numerator of the tangent value y / x.

Return value

angle

Number. An angle, in radians, from -pi through pi. This represents an angle between -180 and 180 degrees, depending on the sign of x and y (see the list below).

<table>
<thead>
<tr>
<th>If</th>
<th>Then angle is in the range</th>
</tr>
</thead>
<tbody>
<tr>
<td>x is positive</td>
<td>0 to pi/2 (Quadrant I)</td>
</tr>
<tr>
<td>y is positive</td>
<td></td>
</tr>
<tr>
<td>x is negative</td>
<td>pi/2 to pi (Quadrant II)</td>
</tr>
<tr>
<td>y is positive</td>
<td></td>
</tr>
<tr>
<td>x is negative</td>
<td>-pi to -pi/2 (Quadrant III)</td>
</tr>
<tr>
<td>y is negative</td>
<td></td>
</tr>
<tr>
<td>x is positive</td>
<td>-pi/2 to 0 (Quadrant IV)</td>
</tr>
<tr>
<td>y is negative</td>
<td></td>
</tr>
</tbody>
</table>
Language cross-reference
ATn2 function in LotusScript language

Examples: @ATan2
1. This example returns pi/4.
   @ATan2( 1; 1 )
2. This example returns 3pi/4.
   @ATan2( -1; 1 )
3. This example returns 1.10715 radians (63.4 degrees).
   @ATan2( 1; 2 )

@AttachmentLengths
Returns a number or a number list containing the length of each attachment to the current document. The number(s) returned are only approximations; the actual size(s) of the attachment(s) may be slightly different.

Syntax
@AttachmentLengths( excludeMIMEBody )

Parameters
excludeMIMEBody

Boolean. Optional.
- Specify True (1) to exclude large MIME parts that are stored as attachments (but displayed in-line). This is the default.
- Specify False (0) to include large MIME parts that are stored as attachments (but displayed in-line).

Return value
sizeInBytes

Number or number list.
- If the current document contains one attachment, sizeInBytes is a number representing the size of that attachment in bytes.
- If the current document contains more than one attachment, sizeInBytes is a number list where each number in the number list is the size of one of the attachments, in bytes.

Usage
The attachment size is computed based on uncompressed file size (that is, the number of bytes the attachment would use if you extracted it); the actual disk storage space required for the file may be somewhat smaller.

@AttachmentLengths returns an empty list if there are no attachments. If there is one attachment of length 0, @AttachmentLengths returns 0.
Language cross-reference
FileSize property in LotusScript NotesEmbeddedObject class
FileSize property in Java EmbeddedObject class

Examples: @AttachmentLengths
1. This example returns 6102 if that is the approximate size of the single, attached file.
   @AttachmentLengths
2. This example, given a semicolon as the multi-value separator, returns
   AUTOEXEC.BAt:112;CONFIG.SYS:1549;Q4SALES.WK4:17636 if those are the names and lengths of the
   files attached.
   @AttachmentNames + ":" + @Text(@AttachmentLengths)
3. This example returns 0 if there is one attachment of length 0.
   @AttachmentLengths
4. This example returns an empty list (no value appears at all) if there are no attachments.
   @AttachmentLengths
5. This example sums the attachment lengths, checking first to make sure there are attachments.
   @Sum(@Attachments > 0; @AttachmentLengths; 0)

@AttachmentModifiedTimes
Returns a datetime that displays the date on which the file attachment associated with the current
document was last modified. If the document contains more than one file attachment, returns the
modification dates in a datetime list.

Note: This @function is new with Release 6.

Syntax
@AttachmentModifiedTimes( excludeMIMEBody )

Parameters
excludeMIMEBody

Boolean. Optional.
• Specify True (1) to exclude large MIME parts that are stored as attachments (but displayed in-line).
  This is the default.
• Specify False (0) to include large MIME parts that are stored as attachments (but displayed in-line).

Return value
modificationDate

Datetime or datetime list.
• If the current document contains one attachment, the modificationDate is a datetime value representing
  the date on which the attachment was last modified.
• If the current document contains more than one attachment, the modificationDate is a datetime list value
  representing the dates on which the attachments were last modified. The order in which the dates
  display in the list matches the order in which the file names display in the text list returned by
  @AttachmentNames.
• If the current document has no attachments, returns a null string ("").
Examples: @AttachmentModifiedTimes

1. For a document that contains a rich text field containing one attached file, this code, when added to a computed datetime field, returns 09/26/2001, the date on which the attached file was last modified.

   @AttachmentModifiedTimes

2. If the document contains a rich text field to which the domino.dtd, whitepaper.pdf, and myreport.wk1 files were attached on August 7th, the following code, when added to a computed datetime field, returns: 09/25/2001;05/10/2001;09/26/2001, which are the respective dates on which the attached files were last modified.

   @AttachmentModifiedTimes

3. If a form contains two rich text fields and you attach the domino.dtd file to the first field, save the document, and reopen it, this code, in a computed datetime field, displays 09/25/2001.

   @AttachmentModifiedTimes

   If you then attach the whitepaper.pdf file to the second rich text field and refresh the document, the computed field changes to display 09/25/2001;05/10/2001. Attaching the myreport.wk1 file to the first rich text field and refreshing the document causes the computed field to return 09/25/2001;05/10/2001;09/26/2001.

@AttachmentNames

Returns the operating system file names of any files attached to a document. If there are multiple files attached, the names are returned as a multiple–value text list.

Syntax

@AttachmentNames( excludeMIMEBody )

Parameters

excludeMIMEBody

Boolean. Optional.

- Specify True (1) to exclude large MIME parts that are stored as attachments (but displayed in-line). This is the default.
- Specify False (0) to include large MIME parts that are stored as attachments (but displayed in-line).

Return value

names

Text or text list.

- If the current document contains one attachment, names is text representing the file name of that attachment.
- If the current document contains more than one attachment, names is a text list where each item is the file name of one of the attachments.

Language cross-reference

Source property in LotusScript NotesEmbeddedObject class

Source property in Java Embedded Object class

Values property in LotusScript NotesItem class
Examples: @AttachmentNames
1. If a file named SALESQ1.WK4 is attached to the document, this example returns: SALESQ1.WK4.
2. Given a semicolon as the multivalue separator, if files named SALESQ1.WK4 and ADMIN.DOC are attached to the document, this example returns: SALESQ1.WK4; ADMIN.DOC.

@Attachments

Returns the number of files attached to a document.

Syntax

@Attachments( excludeMIMEBody )

Parameters

excludeMIMEBody

Boolean. Optional.
- Specify True (1) to exclude large MIME parts that are stored as attachments (but displayed in-line). This is the default.
- Specify False (0) to include large MIME parts that are stored as attachments (but displayed in-line).

Return value

numFiles

Number. The number of files attached to the current document.

Usage in a Column Formula

When used in a column formula in a view or folder, @If(@Attachments;5;0) can be used to display the paper clip icon if the current document has one or more attachments, or displays a blank if there are no attachments. This is the formula used to indicate attachments in the Notes/Domino mail template. For this formula to work, you must select Icon in the Column Definition dialog box for this column.

Examples: @Attachments
1. This example returns 3 if there are three files attached to a document.
2. This example returns 0 if there are no files attached to a document.

@Author

Returns a text list containing the names of the author(s) of the current document.

Syntax

@Author

Return value

authorList
Text list. All the authors of the current document. For authors with hierarchical names, Notes/Domino returns the abbreviated form of the name (as in Denise Lee/Research/Acme), rather than the canonical form (CN=Denise Lee/OU=Research/O=Acme).

@Author uses the following instructions (in the sequence outlined below) to find document author(s) and return the appropriate text list:

1. Search the document for a field of type Authors. If there is one, return the name(s) stored there. (If there are multiple Authors fields, returns the contents of the first Authors field found in the document.)
2. If there is no Authors field, look for a field called From. If there is a From field, look for the field FromDomain.
   • If both fields are found, combine the two fields, separating them by an @ sign (as in, Mary Tsen@AcmeWest).
   • Otherwise, return the contents of the From field only.
3. If there is no From field, look for a field named $UpdatedBy. If there is one, return the contents of the field.
4. If there is no $UpdatedBy field and this is a new document (not yet saved), return the current user’s name.
5. If none of the above can be found, return the null string (″″).

Usage
@Author is most useful for documents containing an Author Names or From field.

Language cross-reference
FieldGetText method in LotusScript NotesUIDocument class
IsAuthors property in LotusScript NotesItem class
IsAuthors property in Java Item class
Authors property in LotusScript NotesDocument class
Authors property in Java Document class

Examples: @Author
If a document has one Authors field that contains: Mary Tsen, David Smith, Denise Lee/Research/Acme. This example returns: Mary Tsen; David Smith; Denise Lee/Research/Acme.
@Author

@Begins
Determines whether a particular substring is stored at the beginning of another string.

Syntax
@Begins( string ; substring )

Parameters
string
Text. Any string.
substring

Text. The string you want to search for at the beginning of string.

Return value
flag

Boolean.
- Returns 1 (True) if substring is contained within string, beginning from the first letter
- Returns 0 (False) if not

Usage
This function is case-sensitive.

Examples: @Begins
1. This example returns 1.
   @Begins("Hi There","Hi")
2. This example returns 0.
   @Begins("Hi There","hi")
3. This example checks the field named Topic; if that field begins with the string "All desks memo", returns the string: Junk Mail. Otherwise, it returns the string: Read this first.
   @If(@Begins(Topic;"All desks memo");"Junk Mail"; "Read this first")

@BrowserInfo
Determines the capabilities of a Web client, that is you can determine the properties of the browser for the current request.

Note: This @function is new with Release 5.

Syntax
@BrowserInfo( "propertyname" )

Parameters
propertyname

Text. The name of the browser property to be retrieved.

Return value
The return value type is dependent on the capability. The table below shows the current set of Web browser and Notes client capabilities that Lotus Notes/Domino supports:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Return type</th>
<th>Return value for browsers</th>
<th>Return value for Notes client</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrowserType</td>
<td>Text</td>
<td>The type of the browser: &quot;Microsoft,&quot; &quot;Netscape,&quot; &quot;Compatible&quot; (for browsers that claim to be compatible with Netscape, including Notes Navigator 5.0), or &quot;Unknown.&quot;</td>
<td>&quot;Notes&quot;</td>
</tr>
<tr>
<td>Cookies</td>
<td>Boolean</td>
<td>1 (True) if the browser supports cookies; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>Property name</td>
<td>Return type</td>
<td>Return value for browsers</td>
<td>Return value for Notes client</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>---------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>DHTML</td>
<td>Boolean</td>
<td>1 (True) if the browser supports dynamic HTML; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>FileUpload</td>
<td>Boolean</td>
<td>1 (True) if the browser supports file upload; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>Frames</td>
<td>Boolean</td>
<td>1 (True) if the browser supports the HTML &lt;FRAME&gt; tag; otherwise 0 (False).</td>
<td>1 (True)</td>
</tr>
<tr>
<td>Java</td>
<td>Boolean</td>
<td>1 (True) if the browser supports Java applets; otherwise 0 (False).</td>
<td>1 (True)</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Boolean</td>
<td>1 (True) if the browser supports JavaScript; otherwise 0 (False).</td>
<td>1 (True)</td>
</tr>
<tr>
<td>Iframe</td>
<td>Boolean</td>
<td>1 (True) if the browser supports the Microsoft HTML &lt;IFRAME&gt; tag; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>Platform</td>
<td>Text</td>
<td>The operating system platform of the browser: &quot;Win95,&quot; &quot;Win98,&quot; &quot;WinNT,&quot; &quot;MacOS,&quot; or &quot;Unknown.&quot;</td>
<td>&quot;Unknown&quot;</td>
</tr>
<tr>
<td>Robot</td>
<td>Boolean</td>
<td>1 (True) if the browser is probably a Web robot; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>SSL</td>
<td>Boolean</td>
<td>1 (True) if the browser supports SSL; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>Tables</td>
<td>Boolean</td>
<td>1 (True) if the browser supports the HTML &lt;TABLE&gt; tag; otherwise 0 (False).</td>
<td>1 (True)</td>
</tr>
<tr>
<td>VBScript</td>
<td>Boolean</td>
<td>1 (True) if the browser supports VBScript; otherwise 0 (False).</td>
<td>0 (False)</td>
</tr>
<tr>
<td>Version</td>
<td>Number</td>
<td>The browser version number, or -1 for unrecognized browsers.</td>
<td>Notes client build number</td>
</tr>
</tbody>
</table>

**Usage**

@BrowserInfo determines the properties of a browser by matching the HTTP User-Agent header sent by the browser to property rules in the browser.cnf file in the Lotus Domino data directory. @BrowserInfo also contains hard-coded rules for the Notes client.

@BrowserInfo can be used in all types of formulas except view selection and view column formulas.

Pre-5.0 Notes clients will not be able to open forms that use @BrowserInfo. The client will display the error message "Invalid formula: unknown function/operator." To prevent this error, check the version number of the client in your formulas. Example:

@If(@TextToNumber(@Version) >= 160; @BrowserInfo("BrowserType");"Unknown")

**Language cross-reference**

CGI variables

The browser.cnf file

**Examples: @BrowserInfo**

This example displays the value in the field named KeyThought, if the current browser supports JavaScript; otherwise the value in the field Topic is displayed.
@BusinessDays

Returns the number of business days in one or more date ranges.

Syntax

@BusinessDays( startDates; endDates; daysToExclude; datesToExclude )

Note: This @function is new with Release 6.

Parameters

startDates

Time-date or time-date list. The start of each date range.

endDates

Time-date or time-date list. The end of each date range.

daysToExclude

Numer or number list. Optional. Days of the week not counted as business days, where 1 is Sunday and 7 is Saturday. Decimal numbers are rounded to integers. Numbers other than 1-7 are ignored.

datesToExclude

Time-date or time-date list. Optional. Dates not counted as business days.

Return value

numberOfDays

Number or number list. The number of days from startDates to endDates, inclusive, less daysToExclude and datesToExclude that fall within the date range.

Usage

The operation on startDates and endDates is a pair-wise list operation. If they are not the same length, the shorter list is filled out with the value of the last element.

@BusinessDays returns -1 if the calculation produces a negative number of days, an end date precedes a start date, or a time-date value contains only a time.

Examples: @BusinessDays

1. This agent displays the number of days in 2001 excluding Saturdays, Sundays, and 10 holidays.

   @Prompt([OK];
   @Text(
   @BusinessDays([01/01/2001]; [12/31/2001]; 1 : 7;
   );
   "Business days in 2001")

2. This agent displays the number of days in each quarter of 2001 excluding Saturdays, Sundays, and 10 holidays.
3. This field value formula returns the number of days from StartDate to EndDate, inclusive, less NonWorkDays and Holidays. StartDate and EndDate are time-date fields with scalar values. NonWorkDays is a keyword field with alias values of "1" and "7" for Sunday and Saturday. Holidays is a time-date field that allows multiple values.

4. This code, when added to a view action in a calendar view that contains a multiple-day event, displays a dialog box that shows the number of business days in the event. For instance, if, in your calendar view, you include a Vacation event that lasts for 32 days (startDT field is 08/02/2002 and endDT field is 09/02/2002), when a user selects the Vacation event from the calendar and clicks on the button, a dialog box appears entitled "Business days" that displays 22.

To account for a holiday on September 2, edit the formula as follows:

@Prompt([OK];"Business days";@Text(@BusinessDays(startDT;endDT;1:7;[09/02/2002])))

@Certificate
Extracts information from the Certified Public Key in the Domino Directory.

Syntax
@Certificate( [ dataToRetrieve ] ; Certificate )

Parameters
[ dataToRetrieve ]

Keyword. Must be enclosed in brackets as shown. Use one of the following keywords:

[SUBJECT]
The name of the certified user ID or server ID.

[ISSUER]
The name of the ID used to issue the certificate.

[EXPIRATION]
The date and time that the North American certificate expires.

[INTLEXPIRATION]
The date and time that the International certificate expires.
Required. This specifies the name of the field where the Certified Public Key information is stored.

**Return value**

*dataRetrieved*

Text for the Subject and Issuer names, and time-date values for the Expiration and IntlExpiration dates.

**Usage**

@Certificate is useful within a macro or view selection formula for selecting a list of users whose certificates are about to expire; it is used by several Domino Directory tools.

@Certificate only retrieves data; you cannot use it to change certificate information (use the appropriate Administration menus to update certificates). Certified Public Key information is stored only for users and servers with hierarchical IDs; @Certificate returns a null string for nonhierarchical IDs.

If you use incorrect syntax, @Certificate returns a null string and does not generate an error message.

@Certificate returns a null string (""), instead of the name of the server ID, when used in a Scheduled agent running on the server.

You cannot use this function in Web applications.

**Language cross-reference**

Signer property of LotusScript NotesDocument class

Signer property of Java Document class

**Examples:** @Certificate

1. This example returns CN=Michael Bowling/OU=R&D/O=WorkSavers/C=US for Michael Bowling’s hierarchical ID.
   @Certificate([SUBJECT];Certificate)

2. This example returns the name of the ID that certified the ID.
   @Certificate([ISSUER];Certificate)

3. This example returns the date and time the North American ID expires.
   @Certificate([EXPIRATION];Certificate)

4. This example returns the date and time the International ID expires.
   @Certificate([INTLEXPIRATION];Certificate)

**@Char**

Converts an IBM Code Page 850 code number into the corresponding single character string.

**Syntax**

@Char( *codeNumber* )

**Parameters**

*codeNumber*

Number. Any number between 0 and 255. Non-integer numbers are truncated to integers.
Return value

correspondingChar

Text. A single character that corresponds to codeNumber.

Usage

@Char(10) returns a carriage return.

@Char(9) returns a tab.

Note: In the Notes client, the codeNumber parameters 0 and 9 do not work in column formulas.

@Char(13) returns a carriage return when used in an @Prompt formula.

To add multiple lines to a single column row:
1. In the View Properties box:
   • Change the Lines per row to the number of carriage returns you want to include in the row.
   • Select Shrink rows to content.
2. In the Column Properties box:
   • Choose New Line as the Multi-value separator.
   • Deselect the Show multiple values as separate entries checkbox.
3. In the code for the column formula, specify each string or number that you want to display on a new line as a separate value. Since you set the Multi-value separator to New Line, this inserts a carriage return between each value. For example, the following column formula vertically lists the content of the FirstName field above the content of the LastName field in the column row:

   ```
   first := FirstName;
   last := LastName;
   @Trim(first : last)
   ```

Language cross-reference

Tab function of LotusScript language

AddNewLine method of NotesRichTextItem class

addNewLine method of Java RichTextParagraphStyle class

Examples: @Char

1. This example returns: A.
   ```
   @Char(65)
   ```
2. This example returns: a.
   ```
   @Char(97)
   ```
3. This example returns: 8.
   ```
   @Char(56)
   ```
4. This example returns the character in the field named QuestionnaireNumber if that field is currently filled in; otherwise, returns a null string.
   ```
   @If(@IsAvailable(QuestionnaireNumber);
       @Char(QuestionnaireNumber);"")
   ```
5. This example uses @Char(13) to insert a carriage return into the text of @Prompt.
   ```
   @Prompt([OK]; "Complete"; "The agent has finished." + @Char(13) + "Please exit this document without saving.")
   ```
@CheckAlarms
Triggers the alarm daemon to check for new alarms in the mail file.

Syntax
@CheckAlarms

Usage
You use @CheckAlarms whenever you make changes to any scheduling that involves alarms. This includes creating a new appointment or anniversary event with an alarm, changing an existing appointment that has an alarm (because the mailer daemon has to reread the information to find out when the new alarm should go off), or deleting an appointment that had an alarm.

Language cross-reference
CheckAlarms method of LotusScript NotesUIWorkspace class

@CheckFormulaSyntax
Checks a block of commented out formula language code for errors.

Note: This @function is new with Release 6.

Syntax
@CheckFormulaSyntax(formulaText)

Parameters
formulaText

Text. The formula code to test for errors, commented out. Enclose the formula code in braces ({}). To comment out the code.

Return value
errorInformation

Text or textlist.
• Returns “1” if the formula has no errors.
• Returns the text list "errorMessage" : "errorLine" : "errorColumn" : "errorOffset" : "errorLength" : "errorText"
where each list item is defined as follows:
  errorMessage: Message returned by the compiler.
  errorLine: Line where the error occurred, beginning with 1, not zero. New lines created by wrapped text are not counted.
  errorColumn: Number of character spaces from the first character in the line where the error occurred, beginning with 1.
  errorOffset: Number of character spaces from the first character in the formulaText block where the error occurred, beginning with 1.
  errorLength: Length of the text making up the error.
  errorText: Text or token that the compiler processes as the cause of the error.
Usage
This @function reports compile errors, not run-time errors. A run-time error is generated, for example, if a function has an insufficient number of arguments. This function is useful especially if you are using the @Eval function to execute a text expression at run-time, since you can use it to first check the syntax of any text you supply to @Eval.

Examples: @CheckFormulaSyntax
1. This example returns "Unknown @Function" when used as the default value for a text field.
   formula := {subject:"test";
   remark:"ok";
   SendTo:"Darrin Dogs/Star";
   @MailSend(SendTo;"";"";subject;remark;"ID";[Sign];[Encrypt])};
   @CheckFormulaSyntax(formula);
2. This code returns "1" when used as the default value for a text field.
   formula := {subject:"test";
   remark:"ok";
   SendTo:"Darrin Dogs/Star";
   @MailSend(SendTo;"";"";subject;remark;"ID";[Sign];[Encrypt])};
   @CheckFormulaSyntax(formula);

@ClientType
Returns a text string to differentiate Lotus Notes clients and World Wide Web browsers.

Syntax
@ClientType

Return value
client type
Text.
• Returns "Notes" if the client type is a Lotus Notes client
• Returns "Web" if the client type is a Web browser

Usage
@ClientType is useful within database formulas, form formulas, buttons in forms, and "hide-when" formulas. Do not use @ClientType in column formulas.

@ClientType always returns "None" when executed in a server background agent.

Examples: @ClientType
1. This example returns the client type.
   @Prompt([OK]; "Client type"; @ClientType)
2. This example, used in a button, opens a view called "By Category - Notes" if the client type is "Notes," or a view called "By Category - Web" otherwise.
   @If(@ClientType = "Notes"; @Command([OpenView]; "By Category - Notes");
   @Command([OpenView]; "By Category - Web"))
@Command

Executes a Lotus Notes/Domino command. Most of the standard menu commands can be executed using @Command. In addition, a number of specialized commands are available. In a formula, any command invoked using @Command runs in the order you specify in the formula. This means that any changes made by the command, such as inserting text into a field, affect the rest of the formula (see exceptions below).

Syntax

@Command( [ command ] ; parameters )

Usage

This function does not work in column, selection, hide-when, section editor, window title, field, or form formulas, or in agents that run on a server. It’s intended for use in toolbar button, hotspot, and action formulas.

Exceptions

The commands listed in the Evaluated after all @functions column in the table below always execute after all the functions present in a formula are executed, which means that the action performed by a command cannot be used by a function that follows it in a formula. The commands listed in the Evaluated immediately column have the equivalent functionality to the corresponding Evaluated after all @functions commands, except they execute as soon as they are encountered in the formula.

<table>
<thead>
<tr>
<th>Evaluated after all @functions</th>
<th>Evaluated immediately</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditClear</td>
<td>Clear</td>
</tr>
<tr>
<td>EditProfile</td>
<td>EditProfileDocument</td>
</tr>
<tr>
<td>FileCloseWindow</td>
<td>CloseWindow</td>
</tr>
<tr>
<td>FileDatabaseDelete</td>
<td>DatabaseDelete</td>
</tr>
<tr>
<td>FileExit</td>
<td>ExitNotes</td>
</tr>
<tr>
<td>Folder</td>
<td>FolderDocuments</td>
</tr>
<tr>
<td>NavigateNext</td>
<td>NavNext</td>
</tr>
<tr>
<td>NavigateNextMain</td>
<td>NavNextMain</td>
</tr>
<tr>
<td>NavigateNextSelected</td>
<td>NavNextSelected</td>
</tr>
<tr>
<td>NavigateNextUnread</td>
<td>NavNextUnread</td>
</tr>
<tr>
<td>NavigatePrev</td>
<td>NavPrev</td>
</tr>
<tr>
<td>NavigatePrevMain</td>
<td>NavPrevMain</td>
</tr>
<tr>
<td>NavigatePrevSelected</td>
<td>NavPrevSelected</td>
</tr>
<tr>
<td>NavigatePrevUnread</td>
<td>NavPrevUnread</td>
</tr>
<tr>
<td>ReloadWindow</td>
<td>RefreshWindow</td>
</tr>
<tr>
<td>ToolsRunBackgroundMacros</td>
<td>RunScheduledAgents</td>
</tr>
<tr>
<td>ToolsRunMacro</td>
<td>RunAgent</td>
</tr>
<tr>
<td>ViewChange</td>
<td>SwitchView</td>
</tr>
<tr>
<td>ViewSwitchForm</td>
<td>SwitchForm</td>
</tr>
</tbody>
</table>

Note: The Evaluated as encountered commands are new with Release 6.
@Compare

Compares the alphabetic order of the elements in two lists pair-wise.

Note: This @function is new with Release 6.

Syntax

@Compare( textlist ; textlist ; [ options ] )

Parameters

textlist

Text list. The first two parameters are text lists. If one list is shorter, the last element in the shorter list is repeated until it reaches the same length as the longer list. The corresponding elements of each list are compared.

[ options ]

Keyword list. The list can include any of the following keywords. Conflicting options result in the last specified.

[CASESENSITIVE] (default)

[CASEINSENSITIVE]

[ACCENTSENSITIVE] (default)

[ACCENTINSENSITIVE]

[PITCHSENSITIVE] (default)

[PITCHINSENSITIVE]

Return value

result

Number list. Each element is the result of comparing the corresponding elements in the text lists, and is one of three values:

• 0 if the elements in the two lists are equal
• -1 if the element in the first list is less than the element in the second list. For example, this is the result if the first list contains alice and the second list contains bobby.
• 1 if the element in the first list is greater than the element in the second list. For example, this is the result if the first list contains bobby and the second list contains alice.

Usage

The comparison sequence for the English character set is as follows: the apostrophe, the dash, the numbers 0-9, the alphabetic characters a-z and A-Z, and the remaining special characters. The sequence for the alphabetic characters is in order, lowercase character first: a, A, b, B, and so on through z, Z. This sequence can lead to some anomalies; for example, “new york” compares before “New Boston.” Use the [CaseInSensitive] option, or @UpperCase, @LowerCase, and @ProperCase to address this behavior.

If you set Unicode standard sorting as the sorting option, you cannot select the following keywords or combinations:
• [PITCHINSENSITIVE]
• [CASESENSITIVE];[ACCENTINSENSITIVE]

You specify Unicode standard sorting by setting the notes.ini variable $CollationType to @UCA, or by selecting the "Unicode standard sorting" checkbox that displays in the following dialog boxes:
• Sorting dialog box that displays when you choose File - Preferences - User Preferences - International - Sorting from the main menu
• Database Properties box*
• Design Document Properties box*

*The Unicode option is disabled in the Database and Design Document Properties boxes until you select a default sort order.

For more information on Unicode sorting, see http://oss.software.ibm.com/icu/

Language cross-reference
StrCompare function of LotusScript language

Examples: @Compare

1. This action compares a list to the value "N" and displays the result. Boston and Moscow result in -1 (less than N), Tokyo results in 1 (greater than N), and n and N result in 0.

   list := "Boston" : "Tokyo" : "Moscow" : "N" : "n";
   result := @text(@compare(list; "N"; [CaseInsensitive]));
   @Prompt([OKCANCEL]: [NOSORT];
   "Result"; ""; ""; list + " (" + result + ")")

2. This computed field formula compares the two multi-value fields Name1 and Name2 and posts the result as its value. Text is substituted for the numeric result values.

   @If(Name1 = "" | Name2 = ""; ""; @do{
   comp1 := @Compare(Name1; Name2;
   [CASEINSENSITIVE] ; [ACCENTINSENSITIVE]);
   comp2 := @Replace(@Text(comp1); "-1" : "0" : "1";
   " is less than " : " is equal to " : " is greater than ");
   Names1 + comp2 + Name2))

3. This computed field formula for a multi-value field named Column2 compares Column1 with A and Z to see if its values start in the alphabetic range. Text is posted to Column2 when the value in Column1 is out of range.

   @If(Column1 = ""; "$"; @do{
   Low1 := @Compare(Column1; "A"; [CASEINSENSITIVE]);
   High1 := @Compare(Column1; "Z"; [CASEINSENSITIVE]);
   Low2 := @Replace(@Text(Low1); "-1" : "0" : "1"; "Does not start with alpha" : "" : "");
   High2 := @Replace(@Text(High1); "-1" : "0" : "1"; "" : "" : "Does not start with alpha");
   Low2 + High2))

4. This formula retrieves all the elements that begin with a, b, or c from the text list in the sailboats field.

   @For(n:=1; n <= @Elements(sailboats); n := n+1;
   FIELD result := @If(n=1; @If(@Compare(sailboats[n]; "d"; [CASEINSENSITIVE])=-1;
   sailboats[n];""); @If(@Compare(sailboats[n]; "d"; [CASEINSENSITIVE])=-1;
   sailboats[n];result));
   result

   If the sailboats field contains "Hunter":"C&C";"Pearson":"Contessa":"Bristol," this formula returns "C&C;Contessa;Bristol."
@ConfigFile

Returns the file path for the initialization file for Lotus Notes (notes.ini).

Note: This function is new with Release 6.

Syntax
@ConfigFile

Return value
notes.ini path
String. Returns the file path to the notes.ini initialization file.

Usage
When the formula is executed on the Notes client, it returns the filename and path of the notes.ini initialization file for the Notes client. When the formula is executed on the server or Web server (when accessed in a Web page, for example), it returns the filename and path of the notes.ini initialization file for the server.

Examples: @ConfigFile
1. This formula, when added to a computed field on a form and previewed in Notes, returns: C:\Lotus\Notes\notes.ini if the current Notes client was installed with the standard file configuration.
   @ConfigFile
2. This formula, when added as computed text to a page, returns D:\webapp\notes.ini, when the page being previewed by a Web browser resides in a database hosted by the webapp server.
   @ConfigFile

@Contains

Determines whether a substring is stored within a string.

Syntax
@Contains( string ; substring )

Parameters
string
Text or text list. The string(s) you want to search.

substring
Text or text list. The string(s) you want to search for in string.

Return value
flag
Boolean.
• Returns true (1) if any substring is contained in one of the strings
• Returns false (0) if no substrings are contained in any of the strings
Note: If any element in the substrings is a null string (""), this function always returns true.

Usage
This function is case-sensitive.

You cannot use this function to test for substrings in a rich text field.

Language cross-reference
Instr function in the LotusScript language

Examples: @Contains
1. This example returns 1 to indicate that the substring, "Th," is contained in the string, "Hi There."
   @Contains("Hi There";"Th")
2. This example returns 1 to indicate that the items in one text list are contained in the other text list.
   @Contains("Tom";"Dick";"Harry";"Harry";"Tom")
3. This example returns 1 to indicate that the single text item in one parameter is present in the text list
   that makes up the other parameter.
   @Contains("Tom";"Tom";"Dick";"Harry")
4. This input validation formula for the "RequestShipDate" field checks if the date in the field is invalid
   or if the field named ProductLeadTime contains the phrases "weeks" or "month." If either condition is
   true, when the user saves the document a message box displays stating, "You must request a valid
   ship date."
   @If(@Contains(ProductLeadTime;"weeks";"month"); @If(!@IsTime(RequestedShipDate);
   @Failure("You must request a valid ship date.");@Success;@Success)
5. This view selection formula creates a new view that includes only documents that have a Subject field
   containing the text "Mary Lamb" (in any case).
   SELECT form = "Memo" & @Contains(@LowerCase(Subject); "mary lamb")
6. This action formula opens a WebForm instead of a NotesForm if the user executing the action is
   assigned the role of "[WebUser]" in the ACL.
   @Command(Compose; @If(@Contains(@UserRoles; "WebUser"); "WebForm"; "NotesForm"))

Note: For this example to work, the Enforce a consistent ACL across all replicas of this database
option must be selected.

@Cos
Given an angle in radians, returns the cosine of the angle. In a right triangle, the cosine of an acute angle
is the ratio of the length of its adjacent side to the length of the hypotenuse.

Syntax
@Cos( angle )

Parameters
angle

Number. An angle measured in radians.

Return value
cosine

Number. The cosine of angle, from -1 to 1.
Language cross-reference

Cos function of LotusScript language

Examples: @Cos

1. This formula returns 1.
   \[ @\cos( 2 \times \pi ) \]
2. This formula finds the length of side c in triangle ABC. You know the value of angle C in radians, and the lengths of sides a and b. This formula finds the length of side c.
   \[
   \sqrt{\sqrt{\text{sideA}^2} + \sqrt{\text{sideB}^2} - (2 \times \text{sideA} \times \text{sideB} \times (\cos(\text{angleC}))} \]
   This formula is a version of the law of cosines, which states that for any triangle ABC, \( c^2 = a^2 + b^2 - 2ab\cos(C) \).

@Count

Calculates the number of text, number, or time-date values in a list. This function always returns a number to indicate the number of entries in the list.

This function is similar to @Elements, except that it returns the number 1 instead of 0, when the value it is evaluating is not a list or is a null string.

Note: This @function is new with Release 6.

Syntax

@Count( list )

Parameters

list

Text list, number list, or time-date list.

Return value

numElements

Number. The number of elements in the list. If the value is not a list or is a null string, @Count(list) returns the number 1.

Language cross-reference

GetItemValue method of LotusScript NotesDocument class

Values property of LotusScript NotesItem class

gGetItemValue method of Java Document class

Values property of Java Item class

Examples: @Count

1. This formula returns the value 3 when the stooges field contains the text list: "Moe":"Larry":"Curly":
   \[ @\text{Count}(\text{stooges}) \]
2. This code returns the value 1, even if there are 4 entries under the Kevlar category in the Sails view, when the value of the third column in the view is determined by the simple function, # in View:
This formula returns 1 because the list produced by the # in View simple function is a list of special text, not a standard number list. If @Count were replaced by @Elements, the formula would return 0.

3. This formula returns a list of the largest numbers after performing a pair-wise comparison of the elements in the Asia_total and USA_total fields, which contain number lists representing the monthly sales totals for the year in these two regions:

\[
t_{Asia} := \text{@Count(Asia\_total)};
\]

\[
t_{USA} := \text{@Count(USA\_total)};
\]

\[
dif := (t_{Asia} - t_{USA});
\]

\[
\text{result} := \text{@If(@Sign(dif) = -1;@Subset(USA\_total; t_{USA} = @Abs(dif));}
\]

\[
\text{@Subset(Asia\_total; (t_{Asia} - dif)));
\]

\[
\text{@If(@Sign(dif) = -1;@Max(Asia\_total; result);@Max(result; USA\_total))}
\]

If the USA has not yet posted its fourth quarter sales totals, this formula displays only the results of comparing the figures posted for the first nine months. It does not follow the default behavior of repeating the September sales figure three times to even out the two list lengths.

### @Created

Returns the time–date when the document was created.

#### Syntax

@Created

#### Return value

dateCreated

Time-date. The date when the current document was created.

#### Usage

@Created differs from @Now, in that @Created returns a time–date value that remains constant, while @Now returns a dynamic time–date that changes with each formula evaluation when it is used in a computed field.

In a field formula, Lotus Notes/Domino takes the value for @Created from the server clock, unless the database is local.

### Language cross-reference

Created property of LotusScript NotesDocument class

Created property of Java Document class

#### Examples: @Created

1. This example returns 06/23/95 11:36:50 AM for a document created on June 23, 1995, at 11:36:50 A.M.

\[
@\text{Created}
\]

2. This example returns 8/4/93 3:10:00 PM for a document created on April 4, 1992 at 3:10 P.M.

\[
@\text{Adjust(@Created; 1; 4; 0; 0; 0; 0)}
\]

See @Adjust for an explanation of the parameters following @Created above.

3. This code, when added as the view selection formula, populates the view with only those documents created after July 23, 2001.

\[
\text{SELECT @Created > [07/23/2001]}
\]
4. If you add the following code as the form formula for a view, all documents created before June 1, 2001 display using the "oldFormat" form and those created on or after June 1 use the "newFormat" form.

```plaintext
@If(@Created >= [06/01/01];"newFormat";"oldFormat")
```

5. This view selection formula uses @Created to select only those documents created in the current month. To avoid having the view refresh indicator display, it uses @TextToTime("Today") instead of @Today. Date calculations in views may impact the performance of an application.

```plaintext
SELECT (( @Year( @Created ) = @Year( @TextToTime( "Today" ) ) ) & ( @Month( @Created ) = @Month( @TextToTime("Today") ) ) )
```

### @Date

Translates numbers for the various components of time and date, then returns the time–date value.

#### Syntax

```plaintext
@Date( year ; month ; day ) @Date( year ; month ; day ; hour ; minute ; second ) @Date( time-date )
```

#### Parameters

- **year**
  Number. The year that you want to appear in the resulting date. You must specify an entire four-digit year. (For example, use 1996, not 96).

- **month**
  Number. The month that you want to appear in the resulting date.

- **day**
  Number. The day that you want to appear in the resulting date.

- **hour**
  Number. The number of hours. This value will be truncated from the resulting date.

- **minute**
  Number. The number of minutes. This value will be truncated from the resulting date.

- **second**
  Number. The number of seconds. This value will be truncated from the resulting date.

- **time-date**
  Time-date. For a time–date value such as @Now or [10/31/93 12:00:00], @Date removes the time portion of the value, leaving only the date.

#### Return value

**truncatedTimeDate**

Time-date. The date corresponding to the parameters that you sent to @Date, minus any time components.
Language cross-reference
DateNumber function in LotusScript language
DateOnly property of LotusScript NotesDateTime class
DateOnly property of Java DateTime class
toJavaDate method of Java DateTime class

Examples: @Date
1. This example returns 06/23/95.
   @date(1995; 06; 23)
2. This example returns 06/23/0095.
   @date(95; 06; 23)
3. This example returns 06/23/2095.
   @date(2095; 06; 23)
4. This example returns 06/23/95 if the time–date value in the field named ResponseDate is 06/23/95 03:00:01 PM.
   @date(ResponseDate)
5. This example returns 1/20/93 08:58:12 AM.
   @date(1993; 01; 20; 8; 58; 12)
6. This example returns 11/20/95.
   @date([11/20/95 8:58:12])

@Day

Extracts the day of the month from the specified date.

Syntax
@Day( timeDateValue )

Parameters

timeDateValue

Time-date. The date containing the day value that you want to extract.

Return value
dayOfMonth

Number. The number corresponding to the day of the month indicated by timeDateValue. Returns -1 if the time-date provided contains only a time value and not a date.

Language cross-reference
Day function of LotusScript language

Examples: @Day
1. This example returns 15 if today is July 15, August 15, September 15, and so on.
   @Day(@Now)
2. This example returns the string "Payment received on or before the 15th" if the PaymentReceived field is filled in on or before the 15th of the month; otherwise, it returns the string "Payment received after the 15th."

```
@If(@Day(PaymentReceived)<16;"Payment received on or before the 15th";"Payment received after the 15th")
```

@DB2Schema

Given the name of a database as a text string, returns a text string containing the DB2 schema of that database if it is a db2nsf database or the empty string if it is not a db2nsf database.

**Note:** This @function is new with Release 7.

**Syntax**

@DB2Schema( server : file )

@DB2Schema( server ; replicaID )

**Parameters**

`server`

Text. The name of the server.

`file`

Text. The path and file name of the database. Specify the database path and file name using the appropriate format for the operating system.

`replicaID`

Text. The replica ID of the database.

**Return value**

`schema`

Text. The DB2 schema of the nsf database indicated by `server : file` or `server ; replicaID`. The empty string (""") is returned if the database is a non-DB2 database. Returns an error via @Error if:

- The server cannot be reached
- The database specified in `file` or `replicaID` cannot be found

**Usage**

@DB2Schema is intended to be used with Query Views, where a DB2 SQL query returns a result set for display. This SQL query is an evaluated formula, and may incorporate @functions in the query formula, the evaluation of which results in the text string of the SQL executed in DB2. To SELECT from a Domino Access View (DAV) within the current db2nsf's schema, the DAV table must be qualified by the schema name. Otherwise, DB2 uses the accessing user's name as the schema name. @DB2Schema allows the schema name to be dynamically specified within a query formula.

This function also works in all contexts where @function use is supported, including view selection formulas, and column formulas.
Examples: @Day

1. This example returns 15 if today is July 15, August 15, September 15, and so on.
   @Day(@Now)

2. This example returns the string "Payment received on or before the 15th" if the PaymentReceived field
   is filled in on or before the 15th of the month; otherwise, it returns the string "Payment received after
   the 15th."
   @If(@Day(PaymentReceived)<16;"Payment received on or before the 15th";"Payment received after the 15th")

@DbColumn (Domino data source)

Returns a column of values from a view or folder in a Domino database.

Syntax

@DbColumn( class : cache ; server : database ; view ; columnNumber )

Note: The separator between the class and cache arguments as well as the server and database
arguments is a colon; the rest of the separators are semi-colons.

Parameters

class

Text. Indicates what type of database you are accessing. You can indicate a Domino database with either
"Notes" or "" (null string).

cache

String argument. Optional. In the initial lookup, specify either "" or "NoCache." If the former case,
subsequent lookups to the same data source, you can specify "ReCache."

- "" (null string) caches the results of the lookup. Each subsequent lookup to the same location (within
the same Domino session and so long as the database executing this lookup remains open) reuses that
data until you specify "ReCache." Cached data improves performance and may be a good choice for
stable data.
- "ReCache" refreshes the cache with the latest data from the database. If you want to ensure that this
lookup gets the latest information, specify this option.
- "NoCache" gets the results of the lookup from the database; no cache is used. If you want to ensure
that Domino retrieves the latest information for every lookup, specify this option.

server : database

Text list. The server location and file name of the database. See "Specifying the server and database."

view

Text. The name of the view in which to search. The view name must exactly match the full name of the
view as specified in the View properties (you can omit the alias). If the view cascades from another name
on the menu, include that name, too. See "Specifying the view name."

columnNumber

Number. The column number within the view. Because Lotus Notes/Domino looks up information in the
view based on column numbers, you can only retrieve data that actually appears in the view. See
"Specifying the column number."
Return value

(valuesFound)

Text, numbers, date-time, or text list. The values found in the view column that you indicated. See "Accessing the values found," later in this chapter.

Specifying the server and database

There are several ways to specify the server: database parameter:

- To perform the lookup on the current database (the same database in which the formula is being evaluated), specify "" as the entire argument to the function. "" means the local Domino directory where you are executing.
- To perform a lookup on a local database, use "" for the server name and specify the database name explicitly, such as "":DATABASE.NSF."
- To perform a lookup (from the workstation) on a Domino database that resides on a server, include the server plus the path and file name as a text list, as in "SERVER":DATABASE.NSF."
- If there are multiple copies of the database located on various Domino servers, using the database replica ID in place of both the server and database name allows you to access a replica copy of that database without having to specify either the server name or the database name. For example, if you use "85255CEB:0032AC04" (a database replica ID, found in the database InfoBox) as the database name, Lotus Domino uses a replica of the database to retrieve the information.

Lotus Domino searches for replicas in this order, using the first replica it encounters:

- Workspace
- If there is one replica on your workspace, Lotus Domino uses it.
- If there are multiple, stacked replicas on your workspace, Lotus Domino uses the replica on top of the stack.
- If there are multiple, unstacked replicas on your workspace, Lotus Domino looks for an icon matching your current server and uses that. If none of the icons matches your current server, Lotus Domino uses the icon that was added to your workspace first.
- Current server
- Locally (your hard disk)

Once a replica is located, it’s added to your workspace to save time on future lookups.

Notes

- To avoid typing errors in the replica ID, choose File - Database - Design Synopsis and select Replication. You can then copy the replica ID from the synopsis and paste it into your formula.
- If your database is located in a DOS or OS/2 subdirectory, such as MAIL\MINE.NSF, put a double backslash between the directory and the database name, as in "MAIL\MINE.NSF"; because formulas treat backslashes as "quote" characters.

Specifying a view or folder

You can specify a view (or folder) parameter using either the full name of the view or its synonym. For example, if your Last Name view is cascaded from By Author in the View menu, and has the synonym |LName, the name looks like this in the view InfoBox:

By Author\Last Name|LName

When you reference this view with @DbColumn, you can simply use the LName synonym, enclosed in quotation marks:

"LName"
If the view name doesn’t have a synonym, you can use the By Author name plus the Last Name cascade, again enclosed in quotation marks (but without the synonym). And since the view name is used in a formula, the “\” must be preceded with an additional “\” to ensure that Lotus Domino interprets it correctly:

"By Author\Last Name"

**Specifying the column number**

To specify a columnNumber parameter, you count the view columns from left to right, with the leftmost column being column number 1. Because of the way that Lotus Domino indexes views, however, not every column is counted for the lookup.

Use the following method to calculate the column number for lookup purposes:

1. Count the columns in the view, from left to right. Look at the view in design mode to make sure that you see all the columns, including columns used for sorting or categorizing the view.
2. Discount all columns that display a constant value, such as "Submitted by:" or 32. If a column contains a formula that happens to return the same result for every document, it is not considered a "constant", so be sure to include it in your column count.
3. Discount all columns that consist solely of the following @functions: @DocChildren, @DocDescendants, @DocLevel, @DocNumber, @DocParentNumber, @DocSiblings, @IsCategory, @IsExpandable.
4. Now recount the columns, working from left to right.

This revised column number is the value to specify in the lookup formula.

If you specify a non–existent column, you don’t get an error, but rather a null value.

**Accessing the return values**

If multiple values are returned by @DbColumn, they are formatted as a list and are separated with the multivalue separator designated for the current field in the field InfoBox.

@DbColumn can return no more than 64K bytes of data. Use the following equations to determine how much of your data can be returned using @DbColumn.

For lookups that return text:

\[ 2 + (2 \times \text{number of entries returned}) + \text{total text size of all entries} \]

For lookups that return numbers or dates:

\[ (10 \times \text{number of entries returned}) + 6 \]

**Usage**

@DbColumn is intended mainly for use with keyword formulas. Instead of hard–coding a list of keywords and then periodically updating that list by re–editing the form containing the keyword field, @DbColumn allows you to dynamically retrieve a list of values from a database view or table.

This function does not work in column or selection formulas, or in mail agents.

**Server agents and security**

Consider the database containing @DbColumn as the *source* database, and the database being accessed as the *target* database.
When you use @DbColumn in an agent, it can access data in a target database that is running on either the same server as the one hosting the source database or another server. The agent signer must have at least Reader access to the target database.

**Note:** Agents running on R5 or earlier servers can only access target databases stored on the same server as the source database. In addition, the agent signer must have at least Reader access to the target database. The use of a replica id in the acl is still supported in Release 6 and later. If the agent signer is not available in the acl of a pre-Release 6 database and the replica id is, the replica id is used instead. (You grant access to the source database by adding the replica id of the source database, for example 85255CEB:0032AC04, to the ACL of the target database and assigning it Reader access or higher.)

### Other agents and security

When @DbColumn is used in any other type of formula or agent, it has unlimited access to any target database stored on the user's own workstation. If the target database is stored on another Domino server, the access for @DbColumn is determined by the user’s own access level (based on the user’s Notes ID).

@DbColumn is subject to the Read Access list for a view.

### Language cross-reference

- ColumnValues property of LotusScript NotesDocument class
- getColumn method of LotusScript NotesView class
- ColumnValues property of Java Document class
- getColumn method of Java View class

### Examples: @DbColumn (Domino data source)

This keyword formula uses @DbColumn. Whenever a document is composed using the form, Lotus Domino retrieves the list of product names stored in column 2 of the Inventory On Hand view of the Inventory database (INVENTORY.NSF). This lookup is used in a purchase requisitions application to retrieve a current list of products available in inventory.

```
@DbColumn("";"":INVENTORY.NSF";"Inventory On Hand";2)
```

### @DbColumn (ODBC data source)

Uses data source information to activate the appropriate ODBC driver. The driver then locates the specified DBMS, table, and column, and returns all values in that column. You can optionally specify whether the returned list of values is sorted, whether duplicate values are deleted, and how null values are handled.

**Note:** @DbColumn can only retrieve data; it can’t add, delete, or modify data.

### Syntax

```
@DbColumn( "ODBC" : cache ; data_source ; user_ID1 : user_ID2 ; password1 : password2 ; table ; column : null_handling ; "Distinct" : sort )
```

### Parameters

- **"ODBC"**
  
  String argument. Indicates that you are accessing an ODBC data source.

- **cache**
String argument. Optional. In the initial lookup, specify either "" or "NoCache." If the former case, subsequent lookups to the same data source, you can specify "ReCache."

- "" (null string) caches the results of the lookup. Each subsequent lookup to the same location (within the same Domino session and so long as the database executing this lookup remains open) reuses that data until you specify "ReCache." Cached data improves performance and may be a good choice for stable data.

- "ReCache" refreshes the cache with the latest data from the database. If you want to ensure that this lookup gets the latest information, specify this option.

Note: "ReCache" is new with Release 6.

- "NoCache" gets the results of the lookup from the database; no cache is used. If you want to ensure that Lotus Domino retrieves the latest information for every lookup, specify this option.

\[ \text{data}\_\text{source} \]

Text. The name of the external data source being accessed. A data source indicates the location of one or more database tables.

See "Specifying the data source."

\[ \text{user}\_\text{ID1} : \text{user}\_\text{ID2} \]

Text list. The user IDs needed to connect to the external database. You may need up to two IDs, depending on the DBMS being accessed.

See "Specifying IDs and passwords."

\[ \text{password1} : \text{password2} \]

Text list. The passwords required by the user IDs.

See "Specifying IDs and passwords."

\[ \text{table} \]

Text. The name of the database table being accessed.

\[ \text{column} \]

Text. The name of the column from which data is being retrieved.

\[ \text{null}\_\text{handling} \]

Text. Specifies how null values are treated when the data is retrieved.

See "Specifying null handling."

"Distinct"

String argument. Optional. Removes duplicate values from the list before returning data.

See "Specifying Distinct."

\[ \text{sort} \]
String argument. Specify "**Ascending**" to sort the list of values into ascending order before it is returned; specify "**Descending**" to sort the list in descending order.

See "Specifying sort."

**Return value**

**valuesFound**

Text, number, date-time, or a list of these types. The values found in the **column** you indicated.

See "Accessing the values found," later in this chapter.

**Note:** If you use the option button or the check box user interface for a keywords field, Lotus Domino updates the keyword list only when the document is composed or is loaded for editing. If you use the Standard user interface for the list, the keyword list is updated every time the document is recalculated.

**Specifying the data source**

The data source name can contain up to 32 alphanumeric characters.

@DbLookup can access data sources that have already been registered in the ODBC.INI file (or similar registry on platforms other than Windows™).

**Specifying IDs and passwords**

You only need these arguments if your DBMS requires them.

Instead of storing the IDs in the @DbColumn formula, you can replace them with null strings (""). If an ID is required, the user will be prompted for it. This is useful when you do not want other designers to see IDs, or when you want users to enter their own IDs when accessing external data. However, you must include IDs and passwords in formulas that run automatically (such as an agent) because these formulas don’t prompt for information.

The user IDs and passwords for accessing a data source are required only once per Domino database session, as long as that database remains open. If the user opens another Domino database and executes a formula that accesses the same data source, the user ID and password will be required again.

Password parameters are necessary only when ID parameters are specified. Like IDs, passwords can either be stored in the @DbColumn formula, or prompted for by substituting the null string. If the database password is null, you can omit it from the formula.

For example, for the full ID/password specification, enter:

- "","" (two null strings, separated by a semicolon) to specify no ID and password, or to prompt for both
- "user_ID1";"password1" to specify one user ID and password combination
- "user_ID1";"user_ID2";"password1";"password2" to specify two user ID and password combinations

**Specifying the table name**

You can optionally include the name of the table’s owner to remove ambiguity; use the format "**owner_name.table_name**", with a period to separate the owner name from the table name. For example:

*dbo.author*

**Table** can also refer to a database view in the DBMS being accessed.
Specifying null handling

Normally, null values are ignored and the resulting list is shortened (same as using the Discard option described below).

To control how null values are handled, specify one of the following, appended to the column parameter with a colon:

- "Fail" generates this error message if the column of data contains any null values:
  
  Null values found - canceling @Db function

  No data is returned with the message.

- "Discard" discards the null values, thus shortening the returned list of values. If one or more values are discarded when the @DbColumn formula is executed, you see this message on the status bar:
  
  Caution: NULL values discarded from @Db list.

- "Replacement value" specifies a replacement value for null values. The replacement value must be a quoted string, but if the column is numeric or date-time, the string must be convertible to that type. If your formula includes a sort string argument, the list of values to be returned is sorted before the replacement values are inserted. During sorting, all null values are placed at the beginning of the list for an ascending sort; and at the end for a descending sort. They are not replaced until sorting is complete. This can result in a list that has some values sorted incorrectly. For example, if you specify "zzz" as your replacement value, all the "zzz" values will appear at the beginning of the list, even if you sorted it in ascending order.

  If one or more values are replaced when the @DbColumn formula is executed, you see this message on the status bar:

  Caution: NULL value replaced with user-defined value in @Db list

  Generally, the replacement value should be one that is not likely to appear in the list as valid data; for example, if the column is text, your replacement value might be "***" so that you can easily find those values in Lotus Domino.

Specifying Distinct

The Distinct string argument is similar to @Unique in Lotus Domino, except that Distinct ensures that duplicate values are removed before the data is returned. Using Distinct instead of @Unique has two advantages:

- The formula operates more quickly because the additional work is performed outside of Lotus Domino.
- You can potentially retrieve a larger amount of useful data into Domino -- since the duplicate values are removed at the back-end, more unique values can be returned to Lotus Domino.

Note: Distinct is not supported by all ODBC drivers. If there are null values in the data and you specify Distinct, one null is usually returned.

Specifying sort

If you use the Distinct string argument, you can append the sort parameter to it with a colon. Use one of these keywords for the sort parameter to specify sorting of the return values:

- Ascending sorts the list in ascending order.
- Descending sorts the list in descending order.

If no sort string argument is specified, values are returned in arbitrary order.

Note: The sort keywords are not supported by all ODBC drivers. If you attempt to use both Ascending and Descending in your formula, you see an "Invalid argument" message.
If multiple values are returned, they are formatted as a list and are separated with the multi-value separator designated for the current field.

@DbColumn can return no more than 64KB of data. Use the following equations to determine how much of your data can be returned with @DbColumn.

- For lookups that return text:
  \[ 2 + (2 \times \text{number of entries returned}) + \text{total text size of all entries} \]
  - Each text string is limited to 511 bytes; if only one text string is returned, it is limited to 64KB.

- For lookups that return numbers or dates:
  \[ (10 \times \text{number of entries returned}) + 6 \]

If the user’s NOTES.INI file includes the statement

```
NoExternalApps=1
```

the @DbColumn formula is disabled. The user will not see an error message; the formula fails to execute. This applies to @DbColumn only when you use it with ODBC.

**Usage**

@DbColumn is intended mainly for keyword formulas. Instead of hard-coding a list of keywords and then periodically updating that list, @DbColumn allows you to dynamically retrieve a list of values from an external database table.

This function only works in Web applications if the remote server hosting the table from which data is being retrieved exists on the same machine as the Domino server, which is rarely the case.

**Language cross-reference**

GetValue method of Lotus Connectors ODBCResultSet class

**Examples: @DbColumn (ODBC data source)**

1. This formula retrieves from the inventory database the complete list of colors in which your company’s uniforms are available. The data is stored like this:

<table>
<thead>
<tr>
<th>Item</th>
<th>Size</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>Small</td>
<td>Red</td>
</tr>
<tr>
<td>Skirt</td>
<td>Small</td>
<td>Green</td>
</tr>
<tr>
<td>Sweater</td>
<td>Medium</td>
<td>Red</td>
</tr>
<tr>
<td>Trousers</td>
<td>Medium</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

Use @DbColumn to retrieve the entire contents of the Color column (column 3):

```
@DbColumn("ODBC";"INVENTORY";"";"";"UNIFORMS";"Color")
```

Values in the resulting list appear just as they were encountered in the database; they are not sorted and duplicate values are retained:

```
Red:Green:Red:Yellow
```

2. This example uses the sample "pubs" database that is included with Microsoft SQL Server. The formula uses the ODBC SQL Server driver to access the database, locate the table called "authors" that is owned by user "dbo," and then retrieve the list of names in the "au_lname" column. The author names are sorted in ascending order; null values are discarded.

```
@DbColumn("ODBC";"PUBLISHERS";"dbo";"vanilla";
"dbo.authors"; "au_lname":"Discard";"Ascending")
```
@DbCommand (Domino data source)

Accesses view and folder information from a Domino database in Web applications.

Note: This @function is new with Release 6.

Syntax
@DbCommand("Domino" ; "ViewNextPage")
@DbCommand("Domino" ; "ViewPreviousPage")
@DbCommand("Domino" ; "FolderList" ; promptString ; foldersToExclude)

Parameters
"Domino"
String argument. Indicates that you want to access a Domino data source.

"ViewNextPage"
String argument. Displays the next chunk of documents in an embedded view.

"ViewPreviousPage"
String argument. Displays the previous chunk of documents in an embedded view.

"FolderList"
String argument. Indicates that you want to display a list of the names of folders in the database that are accessible from the Web.

promptString
String. Optional. Use only if including "FolderList" string argument. String to display as the first choice in a Listbox field. If you want the first choice in the list to be "Select a folder," specify:
@DbCommand("Domino" ; "FolderList" ; "Select a folder")

foldersToExclude
Textlist. Optional. Use only if including "FolderList" string argument. Names of the folders you do not want to display in the listbox field. If you do not want the "MyStuff" and "Problems" folders to be included in the list, specify:
@DbCommand("Domino" ; "FolderList" ; "Select a folder" ; "MyStuff" : "Problems")

Usage
You cannot use this function to access a Domino data source in the Notes client.

Use the FolderList string argument with the @DbCommand in a selection formula for a Listbox field that is set to Use formula for choices to display a list of available folders in a Web application. If no folders exist, the Listbox field is empty when it displays and the promptString does not display in it either.

You can use the @FolderDocuments command with the FolderList string argument to copy or move a selected document in an embedded view that has HTML selection enabled into the folder selected from the Listbox field. To do so, complete these steps:
1. Give the Listbox field that uses the @DbCommand the reserved name $$SelectDestFolder.
2. Set the view that is being embedded into the form to Allow selection of documents on the Advanced tab of its View Properties box.
4. Add an action button to the form with the following formula: @Command([FolderDocuments];"","0"). When clicked, the document currently selected in the embedded view is copied to the folder currently selected in the $$SelectDestFolder Listbox field. Replace "0" with "1" to move the selected document instead of copying it.

The "ViewNextPage" and "ViewPreviousPage" string arguments are useful when your form has an embedded view that contains several documents. By adding Next and Previous actions to the form that contains @DbCommand functions with these keywords, you can display the documents in manageable chunks. Set the Embedded View Properties box options as follows:
1. Set the Web access Display setting to Using HTML.
2. Deselect Use default.
3. Select a number in the Lines to display field.

Examples: @DbCommand (Domino data source)
1. This code, when added as the selection formula to a Listbox field that is set to Use formula for choices on the Control tab of the Field Properties box, displays a list of all the folders in a database.
   - Select a folder- appears as the first option in the resulting Listbox.
   @DbCommand("Domino:";"FolderList":"-Select a folder-")
   If you name the Listbox field $$SelectDestFolder, the following code, when added to the "Move the Folder" action button, moves the document selected in the embedded HTML view into the folder selected from the Listbox field.
   @Command([FolderDocuments];"","1")
2. This code, when added as the selection formula to a Listbox field that is set to Use formula for choices on the Control tab of the Field Properties box, displays a list of the folders in the database. However, it prevents the "Private" and "Manager" folders from displaying in the resulting listbox.
   @DbCommand("Domino:";"FolderList":";"Private:";"Manager")
3. On a form that has an embedded view that contains 50 documents, this formula, when added as the code for the Next action button, displays documents 11-20 if the Lines to display field on the Info tab of the Embedded View Properties box is set to 10.
   @DbCommand("Domino:";"ViewNextPage")
   You can also add an action button called Previous that contains the following code. When a user clicks this button, the previous block of pages displays in the embedded view.
   @DbCommand("Domino:";"ViewPreviousPage")

@DbCommand (ODBC data source)
Given data source information from the ODBC.INI file (or equivalent), uses this information to activate the appropriate ODBC driver. The driver then locates the specified DBMS, passes the specified command to it for processing, and returns the data retrieved by that command.

Note: @DbCommand only works with ODBC data sources and only with SELECT statements. If used with statements that don’t retrieve a result set, @DbCommand simply transmits the statement. Use the ODBC capabilities of LotusScript for more extensive interaction.
Syntax
@DbCommand( "ODBC" : cache ; data_source ; user_ID1 : user_ID2 ; password1 : password2 ; command_string : null_handling )

Parameters
"ODBC"

String argument. Indicates that you are accessing an ODBC data source.

cache

String argument. Optional. In the initial lookup, specify either "" or "NoCache." If the former case, subsequent lookups to the same data source, you can specify "ReCache.

- "" (null string) caches the results of the lookup. Each subsequent lookup to the same location (within the same Domino session and so long as the database executing this lookup remains open) reuses that data until you specify "ReCache." Cached data improves performance and may be a good choice for stable data.
- "ReCache" refreshes the cache with the latest data from the database. If you want to ensure that this lookup gets the latest information, specify this option.

Note: "ReCache" is new with Release 6.

- "NoCache" gets the results of the lookup from the database; no cache is used. If you want to ensure that Lotus Domino retrieves the latest information for every lookup, specify this option.

data_source

Text. The name of the external data source being accessed. A data source indicates the location of one or more database tables. See "Specifying the data source."

user_ID1 : user_ID2

Text list. The user IDs needed to connect to the external database. You may need up to two IDs, depending on the DBMS being accessed. See "Specifying IDs and passwords."

password1 : password2

Text list. The passwords required by the user ID(s). See "Specifying IDs and passwords."

command_string

Text. An SQL statement, command statement, or name of a procedure to be executed. See "Specifying a command string."

null_handling

Text. Specifies how null values are treated when the data is retrieved. See "Specifying null handling."

Return value
valuesFound

Text, number, date-time, or a list of these types. The values returned by the command_string. See "Accessing values found."
Note: If you use the option button or the check box user interface for a keywords field, Lotus Domino updates the keyword list only when the document is composed or is loaded for editing. If you use the Standard user interface for the list, the keyword list is updated every time the document is recalculated.

Specifying the data source
The data source name can contain up to 32 alphanumeric characters.

@DbCommand can access data sources that have already been registered in the ODBC.INI file (or similar registry on platforms other than Windows).

Specifying IDs and passwords
You only need these arguments if your DBMS requires them.

Instead of storing the IDs in the @DbCommand formula, you can replace them with null strings (""). If an ID is required, the user will be prompted for it. This is useful when you do not want other designers to see IDs, or when you want users to enter their own IDs when accessing external data. However, you must include IDs and passwords in formulas that will run automatically (such as an agent) because these formulas don’t prompt for information.

The user IDs and passwords for accessing a data source are required only once per Domino database session as long as that database remains open. If the user opens another Domino database and executes a formula that accesses the same data source, the user ID and password will be required again.

Password parameters are necessary only when ID parameters are specified. Like IDs, passwords can either be stored in the @DbColumn formula, or prompted for by substituting the null string. If the database password is null, you can omit it from the formula.

For example, for the full ID/password specification, enter:
• "";"" (two null strings, separated by a semicolon) to specify no ID and password, or to prompt for both
• “user_ID1”;“password1” to specify one user ID and password combination
• "user_ID1";"user_ID2";"password1";"password2" to specify two user ID and password combinations

Note: For complex connections, additional ID and password parameters may be required to connect to the data source.

Specifying the command string
The command_string can be any of the following:
• An SQL statement (it must use the SQL syntax accepted by the back-end DBMS).
• A command statement using the back-end DBMS command language.
• The name of a procedure stored within the back-end DBMS (the procedure contains one or more command strings that are activated when the procedure is called by @DbCommand).

A date-time value must be entered in the format of the database, not that of Lotus Domino; for example, use 1996-01-31-12.00.00 for DB2/2, not 1996-01-31-12:00:00.

Specifying null handling
To control how null values are handled, specify one of the following, appended to the command_string parameter with a colon:
• "Fail" generates this error message if the column of data contains any null values:
  Null values found - canceling @Db function
  No data is returned with the message.
• "Discard" discards the null values, thus shortening the returned list of values. If one or more values are discarded when the @DbCommand formula is executed, you see this message on the status bar:
  Caution: NULL values discarded from @Db list.
• "Replacement value" specifies a replacement value for null values. The replacement value must be a quoted string, but if the column is numeric or date-time, the string must be convertible to that type.
• If your command string includes a sort string argument, the list of values to be returned is sorted before the replacement values are inserted. During sorting, all null values are placed at the beginning or end of the list, depending on the driver. They are not replaced until sorting is complete. This can result in a list that has some values sorted incorrectly.
  If one or more values are replaced when the @DbCommand formula is executed, you see this message on the status bar:
  Caution: NULL value replaced with user-defined value in @Db list.

Generally, the replacement value should be one that is not likely to appear in the list as valid data; for example, if the column is text, your replacement value might be "***" so that you can easily find those values.

Accessing values found
@DbCommand can return no more than 64KB of data. Use the following equations to determine how much of your data can be returned with @DbCommand.
• For lookups that return text:
  \[2 + (2 \times \text{number of entries returned}) + \text{total text size of all entries}\]
  Each text string is limited to 511 bytes; if only one text string is returned, it is limited to 64KB.
• For lookups that return numbers or dates:
  \[(10 \times \text{number of entries returned}) + 6\]

If the user’s NOTES.INI file includes the statement

NoExternalApps=1

the @DbCommand formula is disabled. The user will not see an error message; the formula fails to execute.

Usage
@DbCommand is useful for testing a non-equal relationship (such as less-than), or for testing multiple conditions at the same time. To use @DbCommands, pass a command to the back-end database for processing.

For example, to return data from records where:

\[\text{BALANCE} \geq 1000.00 \text{ and } \text{DAYS\_OVERDUE} > 30\]

Write the selection statement in SQL, and then use @DbCommand to pass that statement to the DBMS for processing; @DbCommand then returns the requested data.

For Web applications, you can use this function only with the syntax:

@DbCommand("Domino";"ViewNextPage")

or

@DbCommand("Domino";"ViewPreviousPage")

to create a link to the next or previous page in a view. You cannot use @DbCommand in any other context with Web applications.
Note: In a Web application, this command acts on an embedded view when it is called from an action on a page or document.

Examples: @DbCommand (ODBC data source)
This formula uses the sample “pubs” database that is included with Microsoft SQL Server. The formula uses an ODBC driver to access the data source called PUBLISHERS, locate the table called “authors” that is owned by user “dbo,” and then retrieve the list of names in the “au_lname” column for those authors who live in California and have a contract. The string CA is enclosed in single quotation marks, since it is already embedded within a quoted command string.

```plaintext
@DbCommand("ODBC";"PUBLISHERS";"dbo";"vanilla"; "SELECT au_lname FROM dbo.authors WHERE contract=1 AND state='CA' ")
```

@DbExists
Given a server and file name, or replica ID, indicates whether the specified database exists.

Syntax
@DbExists( server : file )
@DbExists( server ; replicaID )

Parameters
server
Text. The name of the server. Use an empty string (""") to indicate the local computer.

file
Text. The path and file name of the database. Specify the database path and file name using the appropriate format for the operating system.

replicationID
Text. The replica ID of the database.

Return value
flag
Number.
• Returns 1 (True) if the database exists.
• Returns 0 (False) if it does not exist.

Usage
This function does not work in column or selection formulas, or in agents that run on a server (mail and scheduled agents).

Language cross-reference
Open method of LotusScript NotesDatabase class
open method of Java Database class
Examples: @DbExists

1. This formula returns 1 if FRITES.NSF is in the MAIL directory on the server Belgium. Otherwise it returns 0.
   @DbExists( "Belgium" : "mail\frites.nsf" )

2. This formula checks if a database exists before opening it on the workspace.
   server := @Subset( @MailDbName; 1 );
   file := "mail\blah.nsf";
   @If( @DbExists( server : file ); @PostedCommand([FileOpenDatabase];
     server : file ); @Prompt([OK]; "Sorry"; "The database cannot be located
     on your home server." ) )

3. This formula uses a database’s replica ID instead of its file name:
   Exists := @DbExists("Cheshire";"85255600:00576146");

@DbLookup (Domino data source)

Given a key value, looks in the specified view (or folder) and finds all documents containing the key value in the first sorted column within the view. For each selected document, @DbLookup returns either the contents of a specified column in the view, or the contents of a specified field.

Syntax

@DbLookup( class : cache ; server : database ; view ; key ; fieldName ; keywords ) or @DbLookup( class : cache ; server : database ; view ; key ; columnNumber ; keywords )

Note: The separator between the class and the cache string arguments as well as the server and database are colons; the rest of the separators are semicolons.

Parameters

class

Text. Indicates what type of database you are accessing. You can indicate a Domino database with either "" or "Notes."

cache

String argument. Optional. In the initial lookup, specify either "" or "NoCache." If the former case, subsequent lookups to the same data source, you can specify "ReCache."

• "" (null string) caches the results of the lookup. Each subsequent lookup to the same location (within the same Domino session and so long as the database executing this lookup remains open) reuses that data until you specify "ReCache." Cached data improves performance and may be a good choice for stable data.

• "ReCache" refreshes the cache with the latest data from the database. If you want to ensure that this lookup gets the latest information, specify this option.

Note: "ReCache" is new with Release 6.

• "NoCache" gets the results of the lookup from the database; no cache is used. If you want to ensure that Lotus Domino retrieves the latest information for every lookup, specify this option.

server : database

Text list. The server location and file name of the database. See “Specifying the server and database.”

view
Text. The name of the view or folder in which to search. The view name must exactly match the view’s full name as specified in the view InfoBox (you can omit any synonyms). If the view cascades from another name on the menu, include that name too. See "Specifying the view."

**key**

Text. Determines which document is actually read in order to retrieve a value. A document’s key is the value displayed in the first sorted column within the view. See "Specifying a key."

**fieldName**

Text. The name of the field from which the data will be retrieved, once the correct document(s) has been identified. See "Specifying a field name."

**columnNumber**

Number. When you use a column number, Domino finds all documents in the view that match the specified key, and returns whatever value is displayed in the indicated column for each of those documents, regardless of the formula used to display the data. See "Specifying the column number."

**keywords**

**Note:** This parameter is new with Release 6.

Keyword. Optional. Keywords can be concatenated.

- [FAILSILENT] returns "" (null string) instead of an error if the key cannot be found.
- [PARTIALMATCH] returns a match if the key matches the beginning characters of the column value.
- [RETURNDOCUMENTUNIQUEID] returns the UNID of the document instead of a field or column value.

**Return value**

**valuesFound**

Text, numbers, date-time, or text-list. The values found in the fieldName or column you indicated, or the UNID of the document. See "Accessing the return values" below.

**Specifying the server and database**

There are several ways to specify the server: database parameter:

- To perform the lookup on the current database (the same database in which the formula is being evaluated), specify "" as the entire argument to the function. "" means the local Domino directory where you are executing.
- To perform a lookup on a local database, use "" for the server name and specify the database name explicitly, such as "":"DATABASE.NSF."
- To perform a lookup (from the workstation) on a Domino database that resides on a server, include the server plus the path and file name as a text list, as in ""SERVER":"DATABASE.NSF."
- If there are multiple copies of the database located on various Domino servers, using the database replica ID in place of both the server and database name lets you access a replica copy of that database without having to specify either the server name or the database name. For example, if you use ""85255CEB:0032AC04"" (a database replica ID, found in the database InfoBox) as the database name, Lotus Domino uses a replica of the database to retrieve the information.

Lotus Domino searches for replicas in this order, using the first replica it encounters:

- Workspace
If there is one replica on your workspace, Lotus Domino uses it.
If there are multiple, stacked replicas on your workspace, Lotus Domino uses the replica on top of the stack.
If there are multiple, unstacked replicas on your workspace, Lotus Domino looks for an icon matching your current server and uses that. If none of the icons matches your current server, Lotus Domino uses the icon that was added to your workspace first.
- Current server
- Locally (your hard disk)
Once a replica is located, it’s added to your workspace to save time on future lookups.

Notes
• To avoid typing errors in the replica ID, choose File - Database - Design Synopsis and select Replication. Then copy the replica ID from the synopsis and paste it into your formula.
• If your database is located in a DOS or OS/2 subdirectory, such as MAIL\MINE.NSF, put a double backslash between the directory and the database name, as in "MAIL\\MINE.NSF" because formulas treat single backslashes as escape characters.

Specifying a view
You can specify a view parameter using either the full name of the view (or folder) or its synonym. For example, if your Last Name view is cascaded from By Author in the View menu, and has the synonym \LName, it looks like this in the view InfoBox:

By Author\Last Name \LName

When you reference this view with @DbLookup, you can just use the LName synonym, enclosed in quotation marks:

"LName"

If the view name doesn’t have a synonym, you use the By Author name plus the Last Name cascade, again enclosed in quotation marks (but without the synonym). And since the view name is used in a formula, the \ must be preceded with an additional \ to ensure that Lotus Domino interprets it correctly:

"By Author\Last Name"

Specifying a key
You can only test for values that match the key (equality); there is no way to specify a different operator such as < (less-than).

In addition to specifying a constant as the key to be matched, you can also use the value of an editable field. For example, you could create a ContactInfo form that contains two fields: a contactName field and a lookupComments field. You want a user to be able to enter a contact name in the contactName field and have the lookupComments field display a list of comments associated with the contact that the user supplied. To do so, you could make the contactName field an editable text field (or a choice list field such as a Dialog list field). The lookupComments field could contain the following code as its Input validation formula:

@DbLookup("";"NoCache";"Sales";"Customers.nsf";"ContactList";contactName;"Comments")

When a user enters or chooses the customer name, "Susie Queue," for instance, in the contactName field of the ContactInfo form and presses F9 to refresh the document, the formula in the lookupComments field performs these tasks:
- Finds the ContactList view of the Notes database Customers.nsf on the Sales server.
- Locates the first sorted column containing the key "Susie Queue."
- Extracts the text strings displayed in the Comments column for each document containing the "Susie Queue" key.
- Returns the extracted list of comments to the lookupComments field. If more than one document was accessed, the strings returned are separated by a semicolon.

By specifying the field contactName as the key, whenever the @DbLookup formula is executed, the current value of the contactName field is used as the lookup criterion.

The match between the lookup key and the value in the sort column must be exact -- capitalization doesn’t matter, but spacing and punctuation must be precise. The match must be complete unless you specify the [PARTIALMATCH] keyword.

The view must contain a sorted column in order for the lookup to work; otherwise a null value is returned. Results are not accurate for a multi-value field that is sorted but not categorized.

**Specifying a field name**

When you use a fieldName to perform a lookup, the value returned is the value that is actually stored in the field; it may be different from what displays in the view. Lotus Domino can retrieve data from any field in any document displayed in the specified view, but if the field isn’t displayed as a view column, Lotus Domino must search the entire document to find the field, which may result in a slower lookup. You cannot retrieve data from a rich text field using @DbLookup.

Some of the documents matching the key may not even contain the specified field if they were created using different forms.

**Specifying the column number**

Lookups based on view columns are more efficient than those based on fields not included in the view. For best results, you should include the desired field in the view.

For example, if your view is categorized by product ID and you specify "01776" as the lookup key and 2 as the column, Lotus Domino returns whatever is displayed in column 2 for all documents categorized under product ID 01776.

To specify a columnNumber parameter, you count the view’s columns from left to right, with the leftmost column being number 1. Because of the way Lotus Domino indexes views, however, not every column is counted for the lookup.

Use this method to calculate the column number for lookup purposes:

1. Count the columns in the view, from left to right.
   - Be sure you don’t miss any columns, for example, a column used for sorting or categorizing the view may not show up. Look at the view in design mode to make sure you see all its columns.
2. Discount all columns that display a constant value, such as 32 or "Submitted by: ." If a column contains a formula that happens to return the same result for every document, it is not considered a "constant" so be sure to include it in your column count.
3. Discount all columns that consist solely of the following @functions: @DocChildren, @DocDescendants, @DocLevel, @DocNumber, @DocParentNumber, @DocSiblings, @IsCategory, @IsExpandable.
4. Now recount the columns, working from left to right.
   - This revised column number is the value to specify in the lookup formula.
Note: If you choose to use a column number instead of a field name in an @DbLookup formula, you can only retrieve data that actually appears in the view.

Accessing the return values
If multiple values are returned by @DbLookup, they are formatted as a list and are separated with the multi–value separator designated in the current field’s InfoBox.

@DbLookup can return no more than 64KB of data. Use the following equations to determine how much of your data can be returned with @DbLookup.

For lookups that return text:

2 + (2 * number of entries returned) + total text size of all entries

For lookups that return numbers or dates:

(10 * number of entries returned) + 6

Usage
This function does not work in column or selection formulas, or in mail agents.

Server agents and security
Consider the database containing @DbLookup the source database, and the database being accessed the target database.

When you use @DbLookup in an agent, it can access data in a target database that is running on either the same server as the one hosting the source database or another server. The agent signer must have at least Reader access to the target database.

Note: Agents running on R5 or earlier servers can only access target databases stored on the same server as the source database. In addition, the agent signer must have at least Reader access to the target database. The use of a replica id in the acl is still supported in Release 6. If the agent signer is not available in the acl of a pre-Release 6 database and the replica id is, the replica id is used instead. (You grant access to the source database by adding the replica id of the source database, for example 85255CEB:0032AC04, to the ACL of the target database and assigning it Reader access or higher.)

Other agents and security
When @DbLookup is used in any other type of formula or agent, it has unlimited access to any target database stored on the user’s own workstation. If the target database is stored on another Domino server, @DbLookup’s access is determined by the agent signer’s access level (based on the user’s Notes ID).

@DbLookup is subject to the Read Access list for a view.

Examples: @DbLookup (Domino data source)
1. Your organization maintains employee office location and department information in the Person documents in the public Name & Address Book. You might have a Purchasing application where employees fill out Purchase Requests for office supplies. You can have your Notes application look up this information and automatically insert it into documents.

Mary Tsen composes a Purchase Order. The P.O. Number, Date, and Requested By fields are filled in automatically by Notes. Mary fills in the details of the purchase order: quantity, part number, and so on.
When Mary saves the Purchase Order, the delivery information in the lower half of the document is calculated using a series of @DbLookup formulas to retrieve information about that user from the public Name & Address Book:

This is accomplished by using computed fields and writing a lookup formula for each field to be retrieved (Location and Telephone). For example, the formula for the Location field would be:

@DbLookup("";"Purchasing":"Names.NSF";"People"; @Right(RequestedBy; "");"Location")

This formula instructs Lotus Domino to open the Name & Address Book (Names.NSF) on the Purchasing server, locate the People view, and then locate the person whose last name matches the last name in the purchase order’s RequestedBy field. Once the correct document has been located, Lotus Domino copies the information from the Person document’s Location field into the purchase order Location field.

A similar formula then copies Mary’s telephone number from the Person record OfficePhoneNumber field into the purchase order Phone field.

Note: For the DeliverTo field, Mary’s name is determined when the document is composed, using @UserName.

2. Using the Name & Address Book again, you want to retrieve a list of office telephone numbers for everyone in the Purchasing department.

You could use @DbLookup with the key "Purchasing" to retrieve the OfficePhoneNumber field, and Notes would return the telephone number for every employee with "Purchasing" entered in the Department field of their Person record. The phone numbers are returned as a text list, using the selected multivalue separator for the field.

3. This formula returns the value stored in the Status field of the Virus Check document, which is accessed via the In Progress view of the PROJECTS.NSF database stored in the SMITH subdirectory on the RESEARCH server. The information will not be cached, so if this formula is evaluated again during the same Notes session, a new lookup will be performed to ensure that the status retrieved is up to date.

@DbLookup("";"NoCache";"RESEARCH":"SMITH\PROJECTS.NSF"; "In Progress";"Virus Check";"Status")

@DbLookup (ODBC data source)

Uses data source information from the ODBC.INI file to activate the appropriate ODBC driver. The driver then locates the specified DBMS, table, and column, and returns only the values in that column belonging to records whose value in the key column matches the specified key. You can optionally specify whether the returned list of values is sorted, whether duplicate values are deleted, and how null values are handled.

Note: @DbLookup can only retrieve data; it can’t add, delete, or modify data.

Syntax

@DbLookup( "ODBC" : cache ; " data_source " ; " user_ID1 " : " user_ID2 " ; " password1 " : " password2 " ; " table " ; " column " : " null_handling " ; " key_column " ; " key " ; "Distinct" : "sort")

Parameters

"ODBC"

String argument. Indicates that you are accessing an ODBC data source.

cache

String argument. Optional. In the initial lookup, specify either "" or "NoCache." If the former case, subsequent lookups to the same data source, you can specify "ReCache."
• "" (null string) caches the results of the lookup. Each subsequent lookup to the same location (within the same Domino session and so long as the database executing this lookup remains open) re-uses that data until you specify "ReCache." Cached data improves performance and may be a good choice for stable data.
• "ReCache" refreshes the cache with the latest data from the database. If you want to ensure that this lookup gets the latest information, specify this option.

  Note: "ReCache" is new with Release 6.
• "NoCache" gets the results of the lookup from the database; no cache is used. If you want to ensure that Lotus Domino retrieves the latest information for every lookup, specify this option.

"data_source"

Text. The name of the external data source being accessed. This name is specified as the dsn (data source name) in the Data Source Administrator or the odbc.ini file. A data source indicates the location of one or more database tables. See "Specifying the data source."

"user_ID1" : "user_ID2"

Text-list. The user IDs needed to connect to the external database. You may need up to two IDs, depending on the DBMS being accessed. See "Specifying IDs and passwords."

"password1" : "password2"

Text list. The passwords required by the user IDs. See "Specifying IDs and passwords."

"table"

Text. The name of the database table being accessed.

"column"

Text. The name of the column from which data is being retrieved.

"null_handling"

Text. Specifies how null values are treated when the data is retrieved. See "Specifying null handling."

"key_column"

Text. The name of the column used for key matching.

"key"

Text, number, or date-time, or a list. The value to be looked up in key_column. Use the Notes type that agrees with the type of the key column in the data source.

"Distinct"

String argument. Optional. Removes duplicate values from the list before returning data. See "Specifying Distinct."

"sort"
String argument. Sorts the list of values into either ascending or descending order before it is returned. See "Specifying sort."

**Return value**

`valuesFound`

Text, number, date-time, or a list of these types. The values found in the column you indicated. See "Accessing the values found," later in this chapter.

**Note:** If you use the option button or the check box user interface for a keywords field, Lotus Domino updates the keyword list only when the document is composed or opened for editing. If you use the Standard user interface for the list, the keyword list is updated every time the document is recalculated.

**Specifying the data source**

The data source name can contain up to 32 alphanumeric characters.

**Specifying IDs and passwords**

You only need these arguments if your DBMS requires them.

Instead of storing the IDs in the @DbLookup formula, you can replace them with null strings (""). If an ID is required, the user will be prompted for it. This is useful when you do not want other designers to see IDs, or when you want users to enter their own IDs when accessing external data. However, you must include IDs and passwords in formulas that will run automatically (such as an agent) because those formulas don't prompt for information.

The user IDs and passwords for accessing a data source are required only once per Domino database session as long as the database remains open. If the user opens another Domino database and executes a formula that accesses the same data source, the user ID and password will be required again.

Password parameters are necessary only when ID parameters are specified. Like IDs, passwords can either be stored in the @DbLookup formula, or prompted for by the ODBC driver by substituting the null string. If the database password is null, you can omit it from the formula.

For example, for the full ID/password specification, enter:

- ""," (two null strings, separated by a semicolon) to specify no ID and password, or to prompt for both
- "user_ID1","password1" to specify one user ID and password combination
- "user_ID1","user_ID2","password1","password2" to specify two user ID and password combinations

**Specifying the table name**

If the DBMS supports it, you can optionally include the name of the table’s owner to remove ambiguity. Use the format "owner_name.table_name", with a period separating the owner name and the table name.

For example:

"dbo.author"

*Table* can also refer to a database view in the DBMS being accessed.

**Specifying null handling**

To control how null values are handled, specify one of the following, appended to the column parameter with a colon:

- "Fail" generates this error message if the column of data contains any null values:

  Null values found - @0b function
No data is returned with the message.

- "Discard" discards the null values, thus shortening the returned list of values. If one or more values are discarded when the @DbLookup formula is executed, Lotus Domino displays this message on the status bar:
  Caution: NULL values discarded from @Db list.

- "Replacement value" specifies a replacement value for null values. The replacement value must be a quoted string, but if the column is numeric or date-time, the string must be convertible to that type. If your formula includes a sort string argument, the list of values to be returned is sorted before the replacement values are inserted. During sorting, all null values are placed at the beginning of the list for an ascending sort, and at the end for a descending sort. They are not replaced until sorting is complete. This can result in a list that has some values sorted incorrectly. For example, if you specify "zzz" as your replacement value, all those "zzz" values will appear at the beginning of the list even though you sorted it by ascending order.

If one or more values are replaced when the @DbLookup formula is executed, Domino displays this message on the status bar:
  Caution: NULL value replaced with user-defined value in @Db list.

Generally, the replacement value should be one that is not likely to appear in the list as valid data; for example, if the column is text, your replacement value might be "****" so that you can easily find those values.

**Specifying key_column and key**

Use "key_column" to indicate which column to search for the specified "key"; enclose the column name in quotation marks. If the DBMS product uses case–sensitive column names, be sure to use the correct capitalization. The values in the key column do not have to be sorted before you retrieve data with @DbLookup.

Specify a value using the Notes type that agrees with the key column in the data source. For example, specify a number or a number-valued expression when the key column is of any numeric type, such as integer, real, float, or double. If the key is a string (text) value, enclose it in quotation marks. A date-time value must be entered in the format of the database, not that of Lotus Domino; for example, use 1996-01-31-12:00:00 for DB2/2, not 1996-01-31-12:00:00.

Together, the key column and the key form the "where" clause of a selection statement:
"SELECT column WHERE key_column = key"

The ODBC Application Interface always tests for equality and only returns data from records where the value in the key column exactly matches the key. To test whether the value in the key column matches one of several possible values, format the key value as a list, separating items with colons as in "Red:"Blue:"Green." This acts like an OR operation, returning data from all records where the value in the key column matches "Red" OR "Blue" OR "Green." To perform an AND operation or to test for inequality, use @DbCommand to pass the appropriate command string to the DBMS. Also use @DbCommand to pass the appropriate command string if the key is a time-date value, because @DbLookup does not always convert the time-date value to the correct format for time-dates in the DBMS command language.

If you cannot get @DbLookup to return the correct values due to typing or other problems, try using a SELECT statement in @DbCommand.

**Specifying Distinct**

The Distinct string argument is similar to @Unique in Lotus Domino, except that Distinct ensures that duplicate values are removed before the data is returned to Lotus Domino. Using Distinct instead of @Unique has two advantages:
• The formula operates more quickly because the additional work is performed outside of Lotus Domino.
• You can potentially retrieve a larger amount of useful data into Lotus Domino -- since the duplicate values are removed at the back-end, more unique values can be returned to Lotus Domino.

Note: Distinct is not supported by all ODBC drivers. If there are null values in the data and you specify Distinct, one null is usually returned.

Specifying sort
If you use the Distinct string argument, you can append the sort parameter to it with a colon. Use one of these keywords for the sort parameter to specify sorting of the return values:
• "Ascending" sorts the list in ascending order.
• "Descending" sorts the list in descending order.

If no sort string argument is specified, values are returned in arbitrary order.

Note: The sort keywords are not supported by all ODBC drivers. If you attempt to use both Ascending and Descending in your formula, Lotus Domino displays an "Invalid argument" message.

Accessing the values found
If multiple values are returned, they are formatted as a list and are separated with the multi-value separator designated for the current field.

@DbLookup can return no more than 64KB of data. Use the following equations to determine how much of your data can be returned with @DbLookup.
• For lookups that return text:
  \[ 2 + (2 * \text{number of entries returned}) + \text{total text size of all entries} \]
  Each text string is limited to 511 bytes; if only one text string is returned, it is limited to 64KB.
• For lookups that return numbers or dates:
  \[ (10 * \text{number of entries returned}) + 6 \]

If the user’s NOTES.INI file includes the statement:
NoExternalApps=1

the @DbLookup formula is disabled. The user will not see an error message; the formula fails to execute. This applies to @DbLookup only when you use it with ODBC.

Usage
@DbLookup is intended mainly for keyword formulas. Instead of hard-coding a list of keywords and then periodically updating that list, @DbLookup lets you dynamically retrieve a list of values from an external database table.

@DbLookup can’t be used in mail agents, although it does work in paste agents. This function only works in Web applications if the remote server hosting the table from which data is being retrieved exists on the same machine as the Domino server, which is rarely the case.

Examples: @DbLookup (ODBC data source)
1. This formula retrieves from the inventory database the complete list of colors in which company uniforms are available. The data is stored like this:
<table>
<thead>
<tr>
<th>Item</th>
<th>Size</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shirt</td>
<td>Small</td>
<td>Red</td>
</tr>
<tr>
<td>Skirt</td>
<td>Small</td>
<td>Green</td>
</tr>
<tr>
<td>Sweater</td>
<td>Medium</td>
<td>Red</td>
</tr>
<tr>
<td>Trousers</td>
<td>Medium</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

To retrieve the entire contents of the Color column (column 3) for all records where the first sorted column (column 1, Item) contains "Shirt" or "Trousers":

```plaintext
@DbLookup("ODBC"; "INVENTORY"; ""; ""; "UNIFORMS"; "Color"; "Item"; "Shirt" : "Trousers")
```

Since multiple records contain at least one of the keys, the result is a list:

```
Red:Yellow
```

Values in the resulting list appear just as they were encountered in the database; they are not sorted and duplicate values are retained.

2. This example uses the sample “pubs” database that is included with Microsoft SQL Server. The formula uses an ODBC driver to access the data source called PUBLISHERS and locate the table called "authors" that is owned by user "dbo." In this table, the values in the "state" column are compared with the values "CA" and "TN." For every record whose state field contains either "CA" or "TN", the values stored in the "au_lname" field are returned. The author names are sorted in ascending order; null values are discarded.

```plaintext
@DbLookup("ODBC"; "PUBLISHERS"; "dbo"; "vanilla"; "dbo.authors"; "au_lname": "Discard"; "state"; "CA": "TN"; "Ascending")
```

@DbManager

Returns a list of users, groups, and servers who currently have Manager access to the database. In a window title formula, only the name of the first manager listed in the ACL is displayed.

@DbManager does not work in selection formulas or column formulas.

**Syntax**

@DbManager

**Return value**

*managers*

Text or text list. The users, groups, and servers that have manager access.

**Language cross-reference**

Managers property of LotusScript NotesDatabase class

Managers property of Java Database class

**Examples: @DbManager**

1. This example returns "Gerald Brown" if Gerald Brown is the only user with Manager access to the current database.

   ```plaintext
   @DbManager
   ```

2. This example returns "Gerald Brown;Supervisors" if Gerald Brown and a group called Supervisors have Manager access to the current database.

   ```plaintext
   @DbManager
   ```
3. This example returns "GERALD BROWN;LOIS BOYD" if Gerald Brown and Lois Boyd are the two users with Manager access to the current database.

@UpperCase(@DbManager)

@DbName

Returns the name of the current Domino server and database.

Syntax

@DbName

Return value

server ; path

Text list with two elements:

- server is the hierarchical name of the server on which the current database resides.
  This @function returns an empty string ("") if:
  - the database is local
  - the formula is used in a Scheduled agent running on the server
  - the formula is used in a view column
  Use @Name to extract a part of the name; for example, [CN] to extract the common name.

- path is the path and file name of the database.
  This @function returns:
  - the path relative to the Notes or Domino data directory if the database is in the data directory
  - the absolute path if the database is outside the data directory
  If the database is accessed through a directory or database link, this @function returns the location of the:
  - link if the @function is running locally (even if the database is on a server), so that the database appears to be where the link is
  - actual database if the @function is running on a server (for example, a scheduled agent)

Usage

Be careful when using @DbName in a column formula. If you build a view, then move the database within the file directory, thus changing its path, you must force a rebuild of the view (Crtl+Shift+F9) for the function to display the updated database information.

Language cross-reference

FileName property of LotusScript NotesDatabase class

FilePath property of LotusScript NotesDatabase class

Server property of LotusScript NotesDatabase class

FileName property of Java Database class

FilePath property of Java Database class

Server property of Java Database class
Examples: @DbName

These examples assume the semicolon is the selected separator.

1. This example returns ";PERSONAL.NSF" if the current document is in the PERSONAL database stored in the data directory of the user's own computer.
   @DbName

2. This example returns "SALES1;ADMIN\STATUS.NSF" if the current document is stored in a Domino database named STATUS.NSF in the ADMIN directory on the SALES1 server. If the database is stored at the server's root directory (that is, it is not stored in a subdirectory), the result would be "SALES1;STATUS.NSF." You can extract just the file name of the list by combining @DbName with @Subset, as shown in the example below.
   @DbName

3. This example returns "STATUS.NSF", the file name, since this is the last element in the returned list.
   @Subset(@DbName;–1)

4. This example returns the path of the current database, without the file name. For example, if the current database is SENSES\SOUNDS\SIGH.NSF, this formula returns "SENSES\SOUNDS."
   @LeftBack(@Subset(@DbName;-1);"\")

5. This example displays the server, path, and file name of the current database, substituting the common name for the hierarchical name of the server.
   database := @Subset(@DbName; -1)
   server := @Name([CN]; @Subset(@DbName; 1))
   @Prompt([OK]; "Database name"; @Implode(server) + " " + @Implode(database))

@DbTitle

Returns the title of the current database.

Syntax

@DbTitle

Return value

*title*

Text. The title of the current database.

Language cross-reference

Title property of the LotusScript NotesDatabase class

Title property of the Java Database class

Examples: @DbTitle

This form action formula uses @DbTitle to let the user create and send an e-mail memo to the author of the current document. @DbTitle is used in the memo’s Subject.

```plaintext
return:=@Char(13);
memobody:=@Prompt([OKCANCELEDIT]; "Mail message";
   "Enter the contents of your mail message below." + return + "It will be sent to " + From + "."; "");
@MailSend(From; ""; ""; "Your posting in " + @DbTitle; "
   memobody: return; [IncludeDoclink])
```
@DDEExecute

Passes the specified command string to the DDE application, which is identified by the conversation ID. @DDEExecute is always used in conjunction with @DDEInitiate and @DDETerminate.

Note: DDEExecute is not supported by UNIX or on the Macintosh.

Syntax

@DDEExecute( conversationID ; command )

Parameters

conversationID

The conversationID is returned by the @DDEInitiate function, which must precede the use of @DDEExecute. Use your own variable name; that’s how you pass the conversation ID between Lotus Domino and the other application. If the conversation ID is invalid, Lotus Domino returns an error. See @IsError.

command

Text. The command must be a text string that adheres to the syntax rules of the receiving application (see that application’s documentation). Enclose the command in quotation marks so it can be passed intact to the DDE application; that application will then interpret it as a DDE command.

Return value

acknowledgment

Number.

• Returns @True (1) if the DDE command is successfully executed
• Returns @False(0) if not
• Returns an error if the conversation ID is invalid

Usage

This function is intended for use primarily in field formulas, agents, and toolbar buttons. It does not work in column or selection formulas, and is not intended for use in window title or form formulas. Since the Macintosh does not support DDE, these commands will not work on Macintosh workstations. In addition, the format of the DDE commands may vary somewhat with each platform or application.

If the user’s NOTES.INI file includes the statement:

NoExternalApps=1

then any formula involving @DDE functions is disabled. The user doesn’t see an error message; the formula fails to execute.

You can have up to 10 DDE conversations running concurrently, although under normal circumstances you should only have one conversation running at a time. Be sure to terminate all DDE conversations once they’re completed, or you may run out of sessions and be unable to initiate more conversations when needed.

You cannot use this function in Web applications.

For details on how to access data in external applications, see, "Including OLE Objects in Applications" in the Application Development with Domino Designer book.
Examples: @DDEExecute

This formula conducts a DDE conversation between 1–2–3 for Windows and Domino.

Conv_ID := @DDEInitiate("123W";"Budget95.wk3");
@if (@IsError(Conv_ID); @Do(@Prompt([Ok];"Error"; "Unable to initiate conversation"); @Return(""));
@Do(@DDEPoke(Conv_ID;"A:B6";@Text(Amount));
@DDEExecute(Conv_ID;"[RUN("{Goto}A:B6~")]");
@DDEExecute(Conv_ID;"[RUN("/rfc~~")]");
@DDEExecute(Conv_ID;"[RUN("{Goto}A:B10";[Edit–Copy]")]");
@DDETerminate(Conv_ID);
@Command([EditNextField]);
@Command([EditPaste]));

The line-by-line explanations:

Conv_ID := @DDEInitiate("123W";"Budget95.wk3");

Initiates a conversation between Domino and 1–2–3. This statement specifies which worksheet to use (BUDGET95.WK3) and stores the conversation ID in the variable Conv_ID. Note that the specified file must be open before the @DDEInitiate is executed.
@if (@IsError(Conv_ID); @Do(@Prompt([Ok];"Error"; "Unable to initiate conversation"); @Return(""));

Determines whether the DDE conversation was successfully initiated. If it was, the formula continues; if it wasn’t, a message appears, and the formula stops executing.
@Do(@DDEPoke(Conv_ID;"A:B6";@Text(Amount));

Converts the contents of the numeric Amount field to text, and then passes that value to cell A:B6 in the 1–2–3 worksheet. The value must be converted to text because only text can be passed via DDEPoke.
@DDEExecute(Conv_ID;"[RUN("{Goto}A:B6~")]");

Makes cell A:B6 the current location in the worksheet.
@DDEExecute(Conv_ID;"[RUN("/rfc~~")]");

Passes the Range, Format, Currency command to 1–2–3; cell A:B6 is now formatted for currency values.
@DDEExecute(Conv_ID;"[RUN("{Goto}A:B10";[Edit–Copy]")]");

Passes the Goto and Edit Copy commands to 1–2–3; cursor is moved to cell A:B10 within the worksheet and the value stored in that cell is copied to the Windows Clipboard.
@DDETerminate(Conv_ID);

Terminates the DDE conversation.
@Command([EditNextField]);

Navigates to the next field within the current Domino document.
@Command([EditPaste]));

The contents of the Clipboard (the value from cell A:B10) are pasted into that field.

@DDEInitiate

Initiates a conversation with a DDE server, and returns the conversation ID.

Note: DDEInitiate is not supported by UNIX or on the Macintosh.
Syntax
@DDEInitiate( application ; topic )

Parameters

application
Text. The name of the application you want to initiate a DDE conversation with. This application must be launched before you call @DDEInitiate. The values for application and topic vary from one application to another; the appropriate values can usually be found in the index for the application’s documentation, under "DDE."

topic
Text. The data file you want to use. This file must be opened before you call @DDEInitiate.

Return value

conversationID
This ID identifies the particular DDE conversation so you can pass commands and data to it with @DDEExecute and @DDEPoke, and eventually terminate the conversation with @DDETerminate. Returns an error if the conversation cannot be initiated. See @IsError.

Usage

It is intended for use primarily in field formulas, agents, and toolbar buttons. Since the Macintosh does not support DDE, these commands will not work on Macintosh workstations. This function does not work in column or selection formulas, and is not intended for use in window title or form formulas.

If the user’s notes.ini file includes the statement
NoExternalApps=1

then any formula involving @DDE functions is disabled. The user doesn’t see an error message; the formula fails to execute.

You can have up to 10 DDE conversations running concurrently, although under normal circumstances you should only have one conversation running at a time. Be sure to terminate all DDE conversations once they’re completed, or you may run out of sessions and be unable to initiate more conversations when needed.

You cannot use this function in Web applications.

Initiation failures

If the conversation cannot be initiated, @DDEInitiate will return an error. See @IsError. Below are some reasons why the initiation could fail:

• The workstation operating system does not support DDE (Macintosh).
• The DDE application you’re trying to set up a conversation with is not running. The specified file is not open.
• The DDE application is running, but the specified topic is not open or the application does not support the topic specified with @DDEInitiate.
• The DDE application is running, but the specified file does not open.
• The maximum number of concurrent DDE conversations has been reached (currently, the maximum is 10).
@DDEPoke

Deposits unsolicited data into the specified location within the DDE server application. If the data was successfully inserted into the target location, @DDEPoke returns an ACK (acknowledgement) with the value @True(1); if the attempt was not successful, the call returns a NACK (negative acknowledgment) with the value @False(0). If the conversation ID is invalid, an error is returned (see @IsError).

Note: DDEPoke is not supported by UNIX or on the Macintosh.

Syntax
@DDEPoke( conversationID ; location ; data )

Parameters

conversationID
The conversationID returned by @DDEInitiate.

location
Text. The name of the location where you want to place the data. The location must be a cell, range, or field name; enclose it in quotation marks.

data
Text. Optional. The data you want to place at location. If you want to pass the contents of a non-text field, use @Text to convert it to text first. If data is a text list, only the first value in the list gets passed. If you omit data, Domino passes the contents of the Windows Clipboard to the receiving application. If you supply the data as a parameter, either enclose it in quotation marks or specify a Domino field name.

Usage
It is intended for use primarily in field formulas, agents, and toolbar buttons. Since the Macintosh does not support DDE, these commands will not work on Macintosh workstations. This function does not work in column or selection formulas, and is not intended for use in window title or form formulas.

If the user’s NOTES.INi file includes the statement
NoExternalApps=1
then any formula involving @DDE functions is disabled. The user doesn’t see an error message; the formula fails to execute.

You cannot use this function in Web applications.

@DDETerminate
Terminates the conversation with a DDE application.

Note: DDETerminate is not supported by UNIX or on the Macintosh.

Syntax
@DDETerminate( conversationID )

Parameters

conversationID
The conversationID is returned by @DDEInitiate. Use the same conversationID you used with the @DDEInitiate and @DDEExecute commands.

**Return value**

`status`

- Returns an error if the conversationID is invalid
- Returns nothing if the conversationID is valid

See @IsError.

**Usage**

It is intended for use primarily in field formulas, agents, and toolbar buttons. Since the Macintosh does not support DDE, these commands will not work on Macintosh workstations. This function does not work in column or selection formulas, and is not intended for use in window title or form formulas.

If the user’s NOTES.INI file includes the statement

```
NoExternalApps=1
```

then any formula involving @DDE functions is disabled. The user doesn’t see an error message, the formula fails to execute. Be sure to terminate all DDE conversations once they’re completed, or you may run out of sessions and be unable to initiate more conversations when needed.

You cannot use this function in Web applications.

---

**DEFAULT**

A reserved word that does one of the following:

- Assigns a default value to a field.
- Says that for the duration of the computation of this formula, if a document does not have this field, act as though it does with this as its value.
- Allows you to assign values that provide dynamic defaults to fields.

**Syntax**

```
DEFAULT variableName := value ;
```

**Usage**

This reserved word works in any formula.

Use DEFAULT once in a formula. If this keyword occurs multiple times in a formula, the first occurrence is effective and subsequent occurrences have no effect.

**Note:** Pre-R6 behavior differs: if a formula contains multiple occurrences of DEFAULT, the last occurrence is effective.

**Language cross-reference**

Let statement of LotusScript language

**Examples:** DEFAULT

These two formulas display the value in the field named KeyThought, if that field exists; otherwise, the value in the field Topic is displayed. Using DEFAULT lets you write a simpler formula that is less prone to error, and easier for others to understand.
@If(@IsAvailable(KeyThought);KeyThought;Topic);

and
DEFAULT KeyThought := Topic;
KeyThought;

@DeleteDocument
Deletes the current document.

Syntax
@DeleteDocument

Usage
This function works only in agents.

To mark a document for deletion in a view, see @Command[EditClear]

You cannot use this function in Web applications.

Language cross-reference
DeleteDocument URL command

Remove method of LotusScript NotesDocument class

DeleteDocument method of LotusScript NotesUIDocument class

remove method of Java Document class

Examples: @DeleteDocument
This example deletes those documents where the Status field equals "Closed."
FIELD Status:=@If(Status="Closed";@DeleteDocument;Status)

@DeleteField
Deletes the value of an editable field.

Syntax
FIELD fieldName := @DeleteField

Usage
This function works in agent, view action, and toolbar button formulas.

If the field has a default value, the default value is reinstated after this function deletes the current value.

This function is the same as @Unavailable.

Use this function to delete invisible fields from documents, such as fields created using the @SetField function.
Language cross-reference
Clear method of LotusScript NotesUIDocument class
RemoveItem method of LotusScript NotesDocument class
Remove method of LotusScript NotesItem class
removeItem method of Java Document class
remove method of Java Item class

Examples: @DeleteField
This formula creates a field named NewDate and sets it to today’s date, then removes the field named OldDate from the document.
FIELD NewDate:=@Today
FIELD OldDate:=@DeleteField;

@DialogBox
Brings up a dialog box that displays the current document (either open or selected in a view). The dialog box shares fields with the underlying document. The user interacts with the dialog box as usual, clicking OK or Cancel when finished.

This function can be used with any form, but it’s particularly useful with forms that contain a single table or layout region, because the user can interact with the table or layout region as if it were a dialog box.

Syntax
@DialogBox(form ; [AUTOHORZFIT] ; [AUTOVERTFIT] ; [NOCANCEL] ; [NONEWFIELDS] ; [NOFIELDUPDATE] ; [READONLY] ; [SIZETOTABLE] ; [NOOKCANCEL] ; [OKCANCELATBOTTOM] ; [NONOTE] ; title )

Parameters
form
Text. The name of the form.

[AUTOHORZFIT]
Keyword. Optional. Scales the dialog box horizontally to fit the first layout region or table on the form. Otherwise, the dialog box is not scaled horizontally.

[AUTOVERTFIT]
Keyword. Optional. Scales the dialog box vertically to fit the first layout region or table on the form. Otherwise, the dialog box is not scaled vertically.

[NOCANCEL]
Keyword. Optional. Does not display the Cancel button. Otherwise, the dialog box contains the Cancel button.

[NONEWFIELDS]
Keyword. Optional. Does not add fields to the underlying document that are in the dialog box form but not the underlying form. Otherwise, all dialog box fields are passed to the underlying document.

**[NOFIELDUPDATE]**

Keyword. Optional. Does not pass edits from the fields in the dialog box to the underlying document (for example, if you’re passing the edits somewhere else in a QueryClose script for the dialog box form). Otherwise, the edits are passed to the underlying form.

**[READONLY]**

Keyword. Optional. Prohibits writing to the dialog box (for example, if you are using the dialog box to display a Help screen). Otherwise, the dialog box is read-write. Use of this keyword implies [NoCancel].

**[SIZETOTABLE]**

*Note:* This parameter is new with Release 5.

Keyword. Optional. Applies [AUTOHORZFIT] and [AUTOVERTFIT] to the first table on the form. Otherwise, they are applied to the first layout region on the form.

**[NOOKCANCEL]**

*Note:* This parameter is new with Release 5.

Keyword. Optional. Does not display the OK button. Otherwise, the dialog box contains the OK button. Using this keyword prevents any changes made in the dialog box from being reflected in the current document. If you want the current document to be updated, but don’t want an OK button to display, include a hotspot button that executes the RefreshParentNote @command.

**[OKCANCELATBOTTOM]**

*Note:* This parameter is new with Release 6.

Keyword. Optional. Displays the OK and/or Cancel buttons side-by-side at the bottom right of the dialog box. Otherwise, the buttons appear stacked at the top right.

**[NONOTE]**

*Note:* This parameter is new with Release 7.

Keyword. Optional. Allows the @Dialogbox function to be used without requiring a current document. If a current document exists, it will be ignored. Using this keyword implies [NOFIELDUPDATE] and [NONEWFIELDS] even if they are not explicitly specified. If the [NONOTE] parameter is not specified, a current document is required.

*title*

Text. The title of the dialog box. Defaults to "Lotus Notes."

**Return value**

*none*

Only updates changes made to like-named fields on the current form if the dialog box is saved and closed. See "Sharing of field values" below.
Usage
This function is useful in buttons for actions. It does not work in column or selection formulas, or in agents that run on a server (mail and scheduled agents). It is not intended for use in window title or form formulas. It can be used with any form, but it’s particularly useful with forms that contain a single table or layout region, because the user can interact with the table or layout region as if it were a dialog box.

@DialogBox cannot return data from a rich text field.

You cannot use this function in Web applications.

If the form contains actions on the action bar, they are not displayed in the dialog box.

[AUTOHORZFIT] and [AUTOVERTFIT] allow you to display an entire layout region (no [SIZETOTABLE]) or table ([SIZETOTABLE]) in a dialog box without displaying the rest of the form. If the form has more than one layout region or table, the first is used. For best results:

• Use both [AUTOHORZFIT] and [AUTOVERTFIT].
• In the Layout properties box, deselect “Show border” and select “3D style.”

If [AUTOHORZFIT] and [AUTOVERTFIT] are both omitted, the entire form is used and no sizing takes place. If the form contains no layout region (no [SIZETOTABLE]) or table ([SIZETOTABLE]), the entire form is used and no sizing takes place.

Sharing of field values
@DialogBox displays the current document using a different form. This means:

• If the form has field names in common with the current document, the field values of the current document are displayed in the dialog box. Rich text fields will not be displayed in the form, even if field names are the same as in the current document.

• If the user changes the value of any fields in the dialog box and selects OK, the changes are reflected in the same fields on the current document.

• If the user enters a value for a field in the dialog box, and the current document does not contain a field by that name, the value is added to the document, even if it is not displayed in the form.

• If you do not want to include OK or Cancel buttons on the dialog box, but do want any changes made in it to be reflected in the current document, use @Command([RefreshParentNote]). If you add this command to a hotspot button on the dialog box form, for example, when a user clicks the button the field values in the current document are updated to reflect any changes made to fields having the same names in the dialog box.

Language cross-reference
@DialogBox method of LotusScript NotesUIWorkspace class

Examples: @DialogBox
1. A form called “Profile” contains a button whose formula is
   @DialogBox("Profile Options"; [AUTOHORZFIT] : [AUTOVERTFIT] )
   Both Profile and Profile Options have a field named “Comments.” When the user clicks the button, the document is displayed in a dialog box, using the Profile Options form. The dialog box is scaled to fit the layout region on Profile Options.
   The user can interact with the dialog box, for example, by editing the Comments field.
   The user clicks OK. The changes made to the Comments field are reflected in the document, if it is in Edit mode.

2. This formula displays a form named “Help screen” for reading only.
3. This formula sizes the table in the form to the dialog box and does not display the OK button.
@DialogBox("Table Test"; [AUTOHORZFIT] : [AUTOVERTFIT] : [SIZETOTABLE] : [NOOKCANCEL])

4. This formula displays the “info” form in a dialog box entitled, “Provide information,” that has no dialog box buttons. The only way the user can close this form to return to the current document is to click the Close icon in the upper right corner of the dialog box.
@DialogBox("info"; [NOCANCEL] : [NOOKCANCEL] ; "Provide information")

@Do
Evaluates expressions from left to right, and returns the value of the last expression in the list.

Syntax
@Do( expressions )

Parameters
expressions

Any number of expressions that you want @Do to evaluate. Separate multiple expressions with semicolons: expression1 ; expression2 ; expression3

Return value
lastExpression

The value of the last expression.

Usage
This function is useful in agents, hotspot buttons, and toolbar buttons and when you want to execute multiple expressions from within a single @function. It does not work in column or selection formulas.

Examples: @Do

This formula displays a dialog box asking whether the user wants to edit the current document. If the user selects Yes, another dialog box appears, prompting for the user’s name. If the user now selects Cancel, the formula stops execution; if the user enters a name and selects OK, the document is opened in Edit mode.

If the user selects No at the first dialog box, a different one follows it. This time, a message appears noting that the user chose not to edit the document, and Lotus Domino navigates to the next document in the view.
@if(@Prompt([YESNO]; "Question";
"Edit this document?");
@if(@Prompt([[OKCANCELEDIT]]; "Positive Response";
"You have chosen to edit this document. Select OK if the name below is correct.");
@UserName)!="ERR_CANCEL";
@Command([EditDocument]);@Return(""));
@Do(@Prompt([OK]; "Negative Response";
"You have chosen not to edit this document. Select OK to continue to the next document.");
@Command([NavNext]))

@DocChildren

In a column or window title formula, returns the number of child documents or categories belonging to the current document or category. Only immediate responses count as children. For example, the responses to a main document are its children, but the responses to a response document are not.
Syntax
@DocChildren @DocChildren( defaultString ) @DocChildren( zero-string ; defaultString ) @DocChildren( zero-string ; one-string ; defaultString )

Parameters
defaultString
Text. Optional. The text to return. If a "%" is used in the string, it will be replaced with the number of children documents or categories. Example: "% Responses."

zero-string
Text. Optional. The text to return if the document or category has no children, such as "No Responses."

one-string
Text. Optional. The text to return if the document or category has just one child, such as "One Response."

Return value
The return value depends on how you call @DocChildren:

numChildren
Special text. If @DocChildren is called with no parameters, then the number of child documents belonging to the current document or category is returned. You cannot convert special text to a number.

childString
Special text. If @DocChildren is called with one or more parameters, it returns the appropriate string, based on the number of child documents belonging to the current document or category. You cannot convert special text to a number.

Usage
Use @DocChildren in window title and column formulas, when you want to indicate how many top-level responses a particular document has, or how many main documents are within a particular category. This function does not work in any other formula.

This @function is calculated when the document is opened. Results are undefined in cases where the document is not opened, such as printing from a view.

You cannot use this function in Web applications, except in column formulas.

Language cross-reference
ChildCount property of LotusScript NotesViewEntry class
ChildCount property of Java ViewEntry class

Examples: @DocChildren
If there are three direct descendants to a document:
1. This example returns 3.
   @DocChildren
2. This example returns 3 Responses. Lotus Domino substitutes the appropriate number for %. If the
document doesn’t have any responses, this formula returns 0 Responses.
@DocChildren("% Responses")

3. This example returns 3 Responses. This time, if the document doesn’t have any responses, the formula
returns the message No Responses.
@DocChildren("No Responses";"% Responses")

4. This example returns There are 3 Responses. If the document has one response, the message is 1
Response; if the document has no responses, the message is No Responses.
@DocChildren("No Responses";"1 Response";
"There are % Responses.")

@DocDescendants

In a column or window title formula, returns the number of descendant documents or subcategories
belonging to the current document or category. Where @DocChildren only counts direct descendants,
@DocDescendants counts all descendants, regardless of level.

Syntax
@DocDescendants @DocDescendants( defaultString ) @DocDescendants( zero-string ; defaultString )
@DocDescendants( zero-string ; one-string ; defaultString )

Parameters

defaultString
Text. Optional. The text to return. If a "%" is used in the string, it will be replaced with the number of
descendant documents or categories. Example: "% Responses."

zero-string
Text. Optional. The text to return if the document or category has no descendants, such as "No
Responses."

one-string
Text. Optional. The text to return if the document or category has just one descendant, such as "One
Response."

Return value
The return value can be either special text or text:

numChildren
Special text. If @DocDescendants is called with no parameters, then the number of descendant documents
belonging to the current document or category is returned. You cannot convert special text to a number.

childString
Special text. If @DocChildren is called with one or more parameters, it returns the appropriate string,
based on the number of descendant documents belonging to the current document or category. You
cannot convert special text to a number.
Usage
Use @DocDescendants in window title and column formulas, when you want to indicate the total number of responses (at all levels) to a particular document, or the total number of documents within a particular category. This function does not work in any other formula.

This @function is calculated when the document is opened. Results are undefined in cases where the document is not opened, such as printing from a view.

You cannot use this function in Web applications, except in column formulas.

Language cross-reference
DescendantCount property of LotusScript NotesView class
DescendantCount property of Java View class

Examples: @DocDescendants
If there are three descendants to a document:
1. This example returns 3.
   @DocDescendants
2. This example returns 3 Response(s).
   @DocDescendants("% Response(s)")
3. This example returns 3 Responses. If there are no responses to the document, the formula returns No Responses.
   @DocDescendants("No Responses";"% Responses")
4. This example returns There are 3 Responses. If the document has one response, the message is 1 Response; if the document has no responses, the message is No Responses.
   @DocDescendants("No Responses";"1 Response";
   "There are % Responses.")

@DocFields
Returns a list of all the fields in a document.

Syntax
@DocFields

Return value
fields
Text list. Each item in the list is the name of a field on the document.

Usage
This function works in any formula that runs in the context of one or more documents. It does not work in column and view selection formulas.

After a document is saved, the returned list includes some of the internal Notes fields, such as the Form field, that is added by Lotus Notes to a form when it is saved or the $Links field, that indicates that the form contains a link to another document or database.
Language cross-reference
Fields property of LotusScript NotesForm class

Examples: @DocFields
1. This example returns Form; result; name; phone if those are the names of the fields in a document.
   @DocFields
2. This example returns Yes if used in a field on a form that contains a rich text field containing a link to a database or document.
   @If(@Contains(@DocFields; "$Links"); "Yes"; "No")
3. This example, when used in the postopen event of a form, enables the user to choose a field to alter from a list of the fields on the form then provide a value to put into the chosen field.
   Field fieldtochange := @Prompt([OkCancelEditCombo]; "Edit Fields"; "Please select the field you want to edit."; ""; @DocFields);
   @SetField(fieldtochange; @Prompt([OkCancelEdit]; "New Value"; "Please enter a new value."); "")
4. This example, when used in a field on a form, returns the number of fields contained by that form when it is saved:
   @Elements(@DocFields)-2
   Subtract 2 from the total number of elements to account for the current field and the Form field, which is an internal Notes field.

@DocLength
Returns the approximate size of a document in bytes.

Syntax
@DocLength

Return value
length

Number. The size of the document.

Usage
This function works in any formula that runs in the context of one or more documents.

The number returned is only an approximation. The actual size of the document may differ for the following reasons:
• The number accounts for user data only; it does not take into account per document or per field constants such as static text or formulas.
• The database allocates storage in 64-byte increments; a document may not use all of the 64 bytes allotted to it.

Documents that are open typically use more storage than documents that are closed. The value returned for @DocLength may vary depending on whether it is running in an open document or a closed document; for example, a document selected at the view level.
**Language cross-reference**
LOF function of LotusScript language

Size property of LotusScript NotesDocument class

Size property of Java Document class

**Examples: @DocLength**
This example returns 1808 if that is the approximate number of bytes in the document (one-page document, no enhanced text).

```plaintext
@DocLength
```

---

**@DocLevel**

Returns a text string that represents the level of the document or category.

**Syntax**
```plaintext
@DocLevel
```

**Return value**
```plaintext
level
```

Special text. The level of the document or category. You cannot convert special text to a number.

**Usage**

Use `@DocLevel` in column and window title formulas. If you use it in a window title or field formula, it will always evaluate to "1" until the document has been saved and reopened. This function does not work in any other formula.

This `@function` is calculated when the document is opened. Results are undefined in cases where the document is not opened, such as printing from a view.

You cannot use this function in Web applications.

**Language cross-reference**
ColumnIndentLevel property of LotusScript NotesViewEntry class

ColumnIndentLevel property of Java ViewEntry class

**Examples: @DocLevel**
1. This example of a category returns 1.
   ```plaintext
   @DocLevel
   ```
2. This example of a main document in a category returns 2.
   ```plaintext
   @DocLevel
   ```
3. This example of a response document in a category returns 3.
   ```plaintext
   @DocLevel
   ```
4. This example of a main document that is not in a category returns 1.
   ```plaintext
   @DocLevel
   ```
@DocLock
Locks, unlocks, returns the locked status of the current document, or indicates if a database has document locking enabled.

Note: This @function is new with Release 6.

Syntax
@DocLock ( [ options ] )

Parameters
[ options ]
Keyword. Choose one of the following actions:

[LOCK]
Locks the current document.

[UNLOCK]
Unlocks the current document.

[STATUS]
Indicates the locked status of the current document. Returns null if the document is not locked or a textlist of the users who have locked the document if it is locked.

[LOCKINGENABLED]
Indicates if the current database has document locking enabled. Returns 1 (@True) if locking is enabled and 0 if it is not.

Usage
The current document has to have been saved previously for this function to work properly. The document must be in Read mode when this function is triggered. Additionally, document locking must be enabled for the database or you will get the error, "Attempted a lock operation on a DB that doesn’t support locking" when you try to use the [LOCK], [STATUS], or [UNLOCK] keywords.

To enable document locking:
1. Specify a Lotus Domino server as the Administration Server (Master Lock Server) on the Advanced panel of the Access Control List dialog box for the database.

You cannot use this function in Web applications.

Language cross-reference
Lock method of LotusScript NotesDocument class

LockHolders property of LotusScript NotesDocument class

Unlock method of LotusScript NotesDocument class
Examples: @DocLock
If a user wants to lock a document opened in edit mode, the following four hotspot buttons entitled LockingEnabled, Status, Lock, and Unlock will enable her to do so.

First, to determine if the current document can be locked, the user checks if document locking is enabled for the database. When the user clicks the LockingEnabled button, which contains the following code, it returns 1 to indicate that locking is enabled.

@Prompt([OK];"Checking if document locking is enabled";@DocLock([LOCKINGENABLED]))

If locking is enabled, the user next clicks the Status button, which contains the following code. If it returns an empty message box, the current document is not locked.

@Prompt([OK];"Checking document status";@DocLock([STATUS]))

Once the user clicks the Lock hotspot button which contains the following code, the administrative server locks the document. If the user clicks the Status button again, a message box appears that displays the current user's hierarchical name.

@DocLock([LOCK])

If the user then clicks the Unlock hotspot button that contains the following code, the administrative server unlocks the document. When the user clicks the Status button again, an empty message box appears.

@DocLock([UNLOCK])

@DocMark
In an agent that runs a formula, indicates whether or not you want to save the changes to a document.

Syntax
@DocMark( [Update] ) @DocMark( [NoUpdate] )

Parameters
[UPDATE]
Keyword. Marks a document so that changes made to it are saved to disk.

[NOUPDATE]
Keyword. Marks a document so that changes made to it will not be saved to disk.

Usage
Use @DocMark in any type of agent to indicate if the changes made to a document by the agent should be saved. This function has no effect in any other formula.

You cannot use this function in Web applications.

Language cross-reference
Save method of LotusScript NotesDocument class

SaveToDisk property of LotusScript NotesItem class

save method of Java Document class

IsSaveToDisk property of Java Item class

280 Programming Guide, Volume 1: Overview and Formula Language
@DocNumber

In a column or window title formula, returns a string representing the entry number of the current document or category. For example, 2.3 indicates that the document is the third entry below the second entry.

Syntax

@DocNumber @DocNumber( separator ) @DocNumber( "" )

Parameters

separator

Text. Optional. Indicates a separator to be used in the document number instead of "." (period); must be one character.

""

Empty string argument. Optional. Tells the function to return the least significant item of the document number (in other words, its rightmost component).

Return value

docNum

Special text. The value that represents the document number of the document or category in the view. You cannot convert special text to a number.

Usage

Use @DocNumber in column or window title formulas. In window title or field formulas, it will evaluate to "0" until the document has been saved and reopened. This function does not work in any other formula.

This @function is calculated when the document is opened. Results are undefined in cases where the document is not opened, such as printing from a view.

You cannot use this function in Web applications, except in column formulas.

Language cross-reference

IndentLevel property of LotusScript NotesViewEntry class

GetPosition method of LotusScript NotesViewEntry class

IndentLevel property of Java ViewEntry class

getPosition method of Java ViewEntry class

Examples: @DocNumber

1. This example returns 37.1.3 for entry 37.1.3.
   @DocNumber
2. This example returns 37–1–3 for entry 37.1.3.
   @DocNumber("–")
3. This example returns 3 for entry 37.1.3.
   @DocNumber(""")
@DocOmittedLength

Returns the approximate number of bytes a truncated document lost during replication. The bytes are the total number of bytes per attachment, OLE object, large rich text field, or non-summary items that were too large, according to the replication settings for the database, to be replicated.

Note: This @function is new with Release 6.

Syntax
@DocOmittedLength

Return value

length

Number. The bytes of data that were not replicated. Returns zero if the document has not been truncated, was truncated previously, or was truncated by a pre-Release 6 server.

Usage

This function works only in databases that are running on and were replicated by a Lotus Domino 6 or later server.

Documents can be truncated during database replication to save space. One replication setting option, for instance, enables you to replicate summary data and only 40KB of rich text for each document. In the resulting replica, you can retrieve the rest of a truncated document by choosing Actions - Retrieve Entire Document from the menu. @DocOmittedLength enables you to determine how much information (in bytes) was removed from the document during replication to help you determine if you want to retrieve it.

This function works in any formula that runs in the context of one or more documents.

The number returned is only an approximation. The actual size of the document may differ for the following reasons:
• The number accounts for user data only; it does not take into account per document or per field constants such as static text or formulas.
• The database allocates storage in 64-byte increments; a document may not use all of the 64 bytes allotted to it.

Examples: @DocOmittedLength

1. This example, when used as a column formula, returns the total size of the document:
   @DocLength + @DocOmittedLength

@DocParentNumber

In a column or window title formula, returns a string that represents the entry number of the parent view entry. Both the current view entry and the parent can be either documents or categories.

Syntax
@DocParentNumber @DocParentNumber( separator ) @DocParentNumber( "" )
Parameters

separator

Text. Optional. Indicates a separator to be used in the parent document number instead of ".".

""

Empty string argument. Optional. Tells the function to return the least significant item of the parent document number (in other words, its rightmost component).

Return value

docNum

Special text. The value that represents the document number of the document or category in the view. You cannot convert special text to a number.

Usage

Use @DocParentNumber in column and window title formulas. If you use it in a field formula or window title formula, no result is displayed until the document has been saved and reopened. This function does not work in any other formula.

To determine the number for the current entry, use @DocNumber instead.

You cannot use this function in Web applications, except in column formulas.

Language cross-reference

GetPosition method of LotusScript NotesViewEntry class

getPosition method of Java ViewEntry class

Examples: @DocParentNumber

1. This example returns 37.1.3 for the document or category for which the parent is entry 37.1.3.
   @DocParentNumber
2. This example returns 37–1–3 for the document or category for which the parent is entry 37.1.3.
   @DocParentNumber("–")
3. This example returns 3 for the document or category for which the parent is entry 37.1.3.
   @DocParentNumber(""")

@DocSiblings

In a column or window title formula, returns a string that represents the total number of entries at the same level as a view entry (document or category). The returned total includes the document itself. For example, if the document is entry 8.2, and entries 8.1, 8.3, and 8.4 also exist, then there are four document siblings.

Syntax

@DocSiblings

Return value

numSiblings
Special text. The number of entries at the same level as the document or category. You cannot convert special text to a number.

**Usage**

Use @DocSiblings in column and window title formulas. If you use it in a field or window title formula, it evaluates to 0 until the document has been saved and reopened. This function does not work in any other formula.

This @function is calculated when the document is opened. Results are undefined in cases where the document is not opened, such as printing from a view.

You cannot use this function in Web applications, except in column formulas.

**Language cross-reference**

SiblingCount property of LotusScript NotesViewEntry class

SiblingCount property of Java ViewEntry class

**Examples: @DocSiblings**

This example returns **Response 1 of 4 to Current Vacation Policy** if the document is one of four responses to a document with the string Current Vacation Policy in the Topic field.

```plaintext
@if(@isNewDoc;"New Document";"Response" + @docNumber(" ") +
    " of " + @DocSiblings + " to " + Topic)
```

**@DocumentUniqueID**

The universal ID, which uniquely identifies a document across all replicas of a database. In text format, the universal ID is a 32-character combination of hexadecimal digits (0-9, A-F).

The universal ID is also known as the unique ID or UNID.

**Syntax**

@DocumentUniqueID

**Usage**

If two documents in replica databases share the same universal ID, the documents are replicas.

This function works in any formula that runs in the context of one or more documents.

To display the UNID, you must convert the result of this function to text, that is, you must specify @Text(@DocumentUniqueID).

The unique ID is one part of a document’s entire ID number. To see a document ID, click the Document IDs tab of the document properties box. The UNID is on the first two lines following OF (top line) and ON (second line) in two 8-character segments.

Once created, a document’s UNID never changes. If a document is copied and pasted, the pasted document gets a new UNID.

Every response document has a special field called $Ref that contains the UNID of the parent document.

In a field formula, @DocumentUniqueID (not converted to text) is a link to the document.
Language cross-reference

UniversalID property of LotusScript NotesDocument class

UniversalID property of Java Document class

DocUNID property of LotusScript NotesRichTextDocLink class

Examples: **@DocumentUniqueID**

1. This column formula displays the UNID of each document in the view.
   @Text(@DocumentUniqueID)

2. This computed field formula creates a doclink to the current document.
   @DocumentUniqueID

3. This "Computed when composed" field formula in a "Response" document creates a doclink to the parent document. In the properties box for the "Response" form, "Formulas inherit values from selected document" must be checked.
   @InheritedDocumentUniqueID

4. You want the Project field of a new "Response" document to match the Project field of the parent "Main Topic" document. In the properties box for the "Response" form, check "Formulas inherit values from selected document." Make Project on the "Response" form a computed field and give it this formula:

   **Project**

5. Field inheritance only happens once when the Response is created. However, you want to access the "Main Topic" after the "Response" is created. Create an agent that runs on a schedule, selects all documents in the database that use the form "Response," and runs the following formula:

   FIELD Project := @GetDocField($Ref; "Project");

6. This is a long solution to the above problem. Create a hidden view called, for example, "By doc ID" with the following selection formula:

   SELECT Form = "Main Topic"
   The first column is sorted and its formula is:
   @Text(@DocumentUniqueID)

   Create an agent that runs on a schedule, selects all documents in the database that use the form "Response," and runs the following formula:

   FIELD Project := @DbLookup("":""; "By doc ID"; @Text($Ref); "Project");

   Each time the agent runs, it performs a lookup in the "By doc ID" view to find the "Main Topic" that is the parent of the current "Response" (that is, the document with a @DocumentUniqueID that matches the current document's $Ref field). It then copies the contents of the Project field from the parent to the child.

**@Domain**

Returns the name of the current user's Domino mail domain listed in the current location document of the Personal Address Book.

**Syntax**

@Domain
Return value

domain

Text. The current user’s domain.

Usage

This function works in any formula and is useful in formulas that manipulate mail addresses. When a formula runs on a server, the server is considered the current user, so @Domain returns the name of the server’s domain.

You cannot use this function in Web applications.

Language cross-reference

GetUserInfo method of LotusScript NotesRegistration class

getUserInfo method of Java Registration class

Examples: @Domain

1. This example returns WorkSavers if the current user belongs to the WorkSavers domain.
   @Domain

2. This formula replaces any occurrences of the user’s mail address with a null string, thus removing the current user’s name from CopyTo.
   FIELD CopyTo:=@Replace(CopyTo;@UserName+"@"+@Domain;;);

Note: The above example works only with non–hierarchical names (those IDs certified by a non–hierarchical certifier).

@DoWhile

Executes one or more statements iteratively while a condition remains true. Checks the condition after executing the statements.

Note: This @function is new with Release 6.

Syntax

@DoWhile( statement ; ... ; condition )

Parameters

statement

A formula language statement. The maximum number of statements you can include is 254.

c Condition

Expression that returns a value of True (1) or False (0).

Return value

true

True (1) unless an error occurs during execution of the condition. An "unexpected data type" error occurs if the conditional expression results in a non-numeric value.
Usage

@DoWhile executes the statements then evaluates the condition. If the condition is True (1), @DoWhile executes the statements and evaluates the condition again. If the condition is False (0), @DoWhile terminates.

Tip: If you are looping through a field containing a list, be sure the Allow multiple values check box is selected in the Field Properties box for the list field.

For other iterative statements, see @For and @While

Language cross-reference

Do statement of LotusScript language

Examples: @DoWhile

This agent displays the elements of the Categories field one at a time.

@If(@Elements(Categories) = 0; @Return(0); "");

n := 1;
@DoWhile(
  @Prompt(OK; "Category " + @Text(n); Categories[n]);
  n := n + 1;
  n <= @Elements(Categories)
)

@EditECL

Displays the administration "Workstation Security: Execution Control List" dialog box for a specified address book and name, which lets you change that administration ECL. Administrators can name Administration ECLs. The name is not usually a user name, but whatever name the administrator chooses; for example, Manager, Developer, or LimitedAccess.

Syntax

@EditECL( server : database ; name )

Parameters

server : database

Text list. The server location and file name of the address book. Omit server or specify it as "" (null) for the local Notes/Domino directory.

name

Text. The name of the ECL. Specify "" (null) for the unnamed ECL.

Usage

You cannot use this function in Web applications.

Language cross-reference

See LotusScript NotesRegistration class

See Java Registration class
Examples: @EditECL
This formula edits the administration ECL named "Developers" in the address book on the server Marketing.
@EditECL("Marketing" : "names.nsf"; "Developers")

@EditUserECL
Displays the "Workstation Security: Execution Control List" dialog box, which allows you to change your personal ECL for the current workstation.

Syntax
@EditUserECL

Usage
You cannot use this function in Web applications.

Language cross-reference
See LotusScript NotesRegistration class
See Java Registration class

@Elements
Calculates the number of text, number, or time–date values in a list. This function always returns a number to indicate the number of entries in the list.

Syntax
@Elements( list )

Parameters
list
Text list, number list, or time-date list.

Return value
numElements
Number. The number of elements in the list. If the field value is a null string, @Elements(list) returns the number 0. @Count returns 1 if the field value is a null string or not a list value.

Usage
You can use @Elements in the condition statement of @For functions to set the loop count equal to the number of elements in the list:
@For(n := 1; n <= @Elements(list); n := n + 1; formula)

Examples: @Elements
1. This example returns 4 if the list in the SalesForce field is "Rogers";"Binney";"Harris";"Larson."
   @Elements(SalesForce)
2. This example returns 2.
@Elements("Jones":"Portsmore")

3. This example returns 5.
   3 + @Elements("Liston":"Reed")

4. This example, when added to the concat field, concatenates each element in the dogs field, containing, "Poodles";"Huskies";"Corgis" with each element in the love field, containing: "I love ";"I love ";"I love ".
   @For(n := 1;n <= @Elements(dogs); n := n+1;
   FIELD concat := @If(n = 1;love[n] + dogs[n];concat : (love[n] + dogs[n]))
   concat
   The result of this formula is: I love Poodles; I love Huskies; I love Corgis.

@EnableAlarms
Starts or stops the alarm daemon.

Syntax
@EnableAlarms( enableAlarms )

enableAlarms
Flag. The text "0" or "1". Specify "0" to disable and "1" to enable.

Usage
@EnableAlarms brings up the alarm daemon and sets the $EnableAlarms notes.ini variable. Once the variable is set, re-entering Lotus Notes/Domino brings up the alarm daemon. The "0" option stops the alarm daemon if it is running.

Language cross-reference
EnableAlarms method of LotusScript NotesUIWorkspace class

Enabled property of LotusScript NotesTimer class

@Ends
Determines if a substring is at the end of a string. @Ends is case-sensitive.

Syntax
@Ends( string ; substring )

Parameters

string
Text. The string to search.

substring
Text. The string to search for at the end of string.
Return value

flag

Boolean

- 1 (True) indicates that the substring is at the end of string
- 0 (False) indicates that the substring is not at the end of string

Examples: @Ends

1. This example returns 1.
   @Ends("Hi There";"re")
2. This example returns 0.
   @Ends("Hi There";"The")
3. This formula checks to see if the end of the Signature field contains the strings "Owens" or "Irons" or "Baker." If it does, the string Verify Signature is returned; otherwise, the string Don't Verify Signature is returned.
   @If(@Ends(Signature;"Owens":"Irons":"Baker");"Verify signature"; "Don't Verify Signature")

ENVIRONMENT

A reserved word that sets or gets an environment variable stored in the user’s notes.ini file (Windows, OS/2, and UNIX) or Notes Preferences file (Macintosh).

Syntax

ENVIRONMENT variable := textValue ;

Usage

For information on how to use environment variables, see "Getting and setting environment variables."

To get the value of an environment variable, use @Environment. To set the value of an environment variable, you can also use @Environment, or you can use @SetEnvironment.

For Web applications, use predefined field names to gather information about the Web user’s environment by requesting Common Gateway Interface (CGI) environment variables.

Language cross-reference

Environ function of LotusScript language

GetEnvironmentValue method of LotusScript NotesSession class

GetEnvironmentString method of LotusScript NotesSession class

SetEnvironmentVar method of LotusScript NotesSession class

getEnvironmentValue method of Java Session class

getEnvironmentString method of Java Session class

setEnvironmentVar method of Java Session class

@Environment

Sets or returns an environment variable stored in a formula.
Syntax

@Environment( variable ) @Environment( variable ; value )

Parameters

variable

Text or text list. The name of the environment variable you want to retrieve. To retrieve multiple environment variables, use a text list.

value

Text. Optional. The value you want to assign to the environment variable. Since users have their own notes.ini or Notes Preferences file, this value can be customized for each user. Omit this parameter if you just want to retrieve the value, not set it.

- If variable is a text list, every environment variable in the list will be assigned value.
- If value is a text list, only the first value in the list is used; the rest are ignored.

Return value

environmentVariable

Text. The value of the environment variable you specified. To use the return value in arithmetic operations, use @TextToNumber to convert it to a number.

Usage

For information on how to use environment variables, see “Getting and setting environment variables” in “Formula Language Coding Guidelines.”

Use @Environment when you want to set an environment variable within a formula. If it’s to be nested within another @function (such as @If or @Do), use @SetEnvironment instead.

The ENVIRONMENT keyword works the same as @Environment.

@Environment cannot be used in column or selection formulas; it’s only intended for use in field formulas, toolbar buttons, and agents. Some formulas, such as scheduled agents, are run on the server instead of on the user’s workstation. In this case, the environment variables affected are the server environment variables, not the workstation variables. You can use a computed text formula to retrieve variables, but not to set variables.

You can also use @Environment to get the value of an environment variable stored the user’s notes.ini file (Windows, OS/2, and UNIX) or Notes Preferences file (Macintosh). You can only set and retrieve the values of variables that begin with a dollar sign ($) symbol. Do not include the dollar sign in the variable parameter. For instance, to change the value of the $EnableAlarms INI variable from 1 to 0, enter:

@Environment("EnableAlarms";"0")

For Web applications, use predefined field names to gather information about the Web user’s environment by requesting Common Gateway Interface (CGI) environment variables.

Language cross-reference

Environ function of LotusScript language

GetEnvironmentString method of LotusScript NotesSession class

SetEnvironmentVar method of LotusScript NotesSession class
getEnvironmentString method of Java Session class

setEnvironmentVar method of Java Session class

Examples: @Environment, @SetEnvironment, and ENVIRONMENT

1. This example returns 5, if that is the value of the variable $IEVersionMajor stored in the current user’s notes.ini or Notes Preferences file.
   @Environment("IEVersionMajor")

2. This example places a variable called OrderNumber in the current user’s notes.ini or Notes Preferences file, and assigns it a value of zero.
   @Environment("OrderNumber";"0")

3. To save users time while completing Profile documents, you might want to automatically fill in an office location for them. You can create an editable text field called OfficeLocation. Its default formula is:
   @Environment("ENVOfficeLocation")
   Its input-translation formula is:
   @Environment("ENVOfficeLocation"; OfficeLocation); OfficeLocation
   The first time the user creates a Profile document, the OfficeLocation field is blank, so the user types in the office location. When the document is saved, the contents of the OfficeLocation field are saved in the notes.ini or Notes Preferences file. The next time the user creates a Profile document, the office location is retrieved from the environment variable ENVOfficeLocation, and the user doesn’t have to type it in again (unless the office location changes, in which case the user edits the field).
   You could also write the input-translation formula using either @SetEnvironment or the ENVIRONMENT keyword, both of which achieve the same result:
   @SetEnvironment("ENVOfficeLocation"; OfficeLocation); OfficeLocation
   or
   ENVIRONMENT ENVOfficeLocation:= OfficeLocation;
   OfficeLocation

4. In addition to the OfficeLocation, you might want to use an environment variable to store a user’s birthday. You can create an editable time field called Birthday. Its default formula is similar to the one used for OfficeLocation:
   @Environment("ENVBirthday")
   Its input-translation formula uses @Text to convert the time value into text:
   @SetEnvironment("ENVBirthday"; @Text(Birthday)); Birthday
   Use @Text to write a similar input-translation formula for a number field.

5. You want to generate sequential numbers on a per user basis, and you want to store the number in a field called OrderNumber. Define the field OrderNumber to be a Text data type; it must be some form of computed field. You can then write the following formula for the field.
   Temporary := @Environment("OrderNumber");
   Temporary2 := @If(Temporary="";"0";Temporary);
   CurrentOrderNumber := @TextToNumber(Temporary2);
   NextOrderNumber := CurrentOrderNumber + 1;
   ENVIRONMENT OrderNumber := @Text(NextOrderNumber);
   @Text(CurrentOrderNumber)

6. This formula tests whether an environment variable called OrderNumber has been stored in the user’s notes.ini or Notes Preferences file. If there is no such variable stored, @SetEnvironment initializes it to zero. If a value has already been stored, @Return returns it and stops the formula from executing.
   @If(@Environment(OrderNumber)==""; @SetEnvironment("OrderNumber";"0"); @Return(@Environment("OrderNumber")))
7. Two agents are used to look up a list of possible group names that users might belong to, prompt the user to select one, and then enter that name in the Group field for all selected documents (which, in this case, pertain to the current user).

The Set Group agent looks up the list of group names stored in column 1 of the Service Requests – By Group view, prompts the user to select a group name, and then stores the selected name in the TmpName environment variable before running the "(Set Group Helper)" agent. The "(Set Group Helper)" agent then retrieves the group name from the user’s notes.ini or Notes Preferences file and stores it in the Group name field for all selected documents.

Set Group agent executes once:
```
GroupList:=@DbColumn("":"NoCache";"";
"Service Requests\By Group";1);
Group:=@Prompt([OKCancelEditCombo];"Choose a group";"Choose a group";"Marketing";GroupList);
Tmp1:=@Environment("TmpName";Group);
@Command([RunAgent];"(Set Group Helper)"

(Set Group Helper) agent runs on each selected document:
FIELD Group:=@Environment("TmpName")
```

@Error

Allows you to generate an error condition within an expression. This is useful if you want to evaluate the current values in several fields and need to know if an error has occurred in the entry of any of them.

Syntax
```
@Error
```

Return value
```
@Error
```

Usage

Use @IsError to test for a data entry error.

When an error has occurred, @Error is returned. The function cannot return any other value.

@Error always results in an error condition when it tests a single value. If you use @Error alone as a formula, you will always generate an error.

You cannot test for an @Error value with any operator or @function other than @IsError. If you use an error value as an argument to an operator or @function, the return is always@Error.

Language cross-reference

On Error statement of LotusScript language

Error function of LotusScript language

See Java NotesException and NotesError classes

Examples: @Error

Read the following examples closely to understand the difference between @Error and @IsError.
1. This example returns the value in the Price field if it is greater than 100, otherwise it returns @Error.
```
@if(Price>100;Price;@Error)
```
2. This example checks to see if there is an @Error in the field named Price. If there is an @Error, the string There is an error in the price field is returned. If the contents of the field are anything other than @Error, Price Field Okay is returned.

@If(@IsError(Price);"There is an error in the price field";"Price Field Okay")

@Eval

At run-time, compiles and runs each element in a text expression as a formula. Returns the result of the last formula expression in the list or an error if an error is generated by any of the formula expressions in the list.

Note: This @function is new with Release 6.

Syntax

@Eval( textExpressions )

Parameters

textExpressions

Any text expressions that you want @Eval to evaluate. Surround the text expression to be evaluated with quotation marks (" ") and escape the quotes around individual text expressions within the formula with backslashes (\). Use the plus sign (+) to concatenate text expressions.

Return value

lastExpression

The value of the last expression.

Usage

This function is useful in agents, hotspot buttons, and toolbar buttons and when you want to evaluate multiple text expressions at run-time from within a single @function.

Use of @Eval in view columns and selection formulas may produce unexpected results. Because this function is evaluated at run-time, the view engine is unable to follow its standard procedure of analyzing the formulas ahead of time to discover what types of @functions it will encounter and prepare for them.

Language cross-reference

Examples: @Eval

1. This formula concatenates the value of the temporary variable x and the text expression "bar." It returns "foobar."

   x := "foo";
   @Eval("x + \"bar\"");

2. The following code, when added to an action button, creates the field "comment" and adds the user's input to it on the fly.

   input := {FIELD comment := @Prompt([OKCANCELEdit];"Input";"Input a value"; "Default");};
   @Eval(input);

   To view the content of the comment field, use the following code in a hotspot or action button.
   @Prompt([OK];"Value of comment field";@GetField("comment"))
@Exp

Calculates the number e (approximately 2.718282) raised to the specified power (this value can contain up to 14 decimal places).

Syntax

@Exp( power )

Parameters

power

Number. The power to which you want to raise e. Lotus Notes/Domino can only calculate this function when the number is between −11355.1371 and 11356.5234. Values outside this range will return the value @ERROR.

Usage

Natural logs use the constant e as their base. Use @Exp in formulas requiring exponential functions.

Language cross-reference

Exp function of LotusScript language

Examples: @Exp

1. This example returns 3.49034295746184 (e raised to the power of 1.25).
   @Exp(1.25)
2. This example returns 0.28650479686019 (e raised to the power of −1.25).
   @Exp(−1.25)

@Explode

Returns a text list composed of the elements of a text string or date range.

• If you specify a text string, elements are defined as sequences of characters separated by separator characters and newlines.
• If you specify a time-date range, elements are defined as individual days within the range.

Syntax

@Explode( dateRange ) @Explode( string ) @Explode( string ; separators ) @Explode( string ; separators ; includeEmpties ) @Explode( string ; separators ; includeEmpties ; newlineAsSeparator )

Parameters

dateRange

Time-date range or time-date range list. The range of dates that you want to make into a text list. Specify a valid date-time range, not a string representation of one. For example, @Explode( "05/01/96 - 05/02/96" ) is invalid because the parameter is a string. Use @Explode( [05/01/96 - 05/02/96] ).

string

Text. The string that you want to make into a text list.

separators
Text. Optional. One or more characters that define the end of an element in string. The default separators are "," (space, comma, semicolon), which means that Lotus Domino adds a new element to the text list each time a space, comma, or semicolon occurs in the original string. When you use more than one character to specify separators, each character defines one separator. For example, the specification "and" breaks the string at each occurrence of "a," "n," and "d"; it does not break the string at each occurrence of the word "and." The newline is a separator, regardless of the specification of this parameter, unless newlineAsSeparator is specified as False.

includeEmpties

Boolean. Optional. Specify True (1) to place an empty string (" ") in the returned list when a separator appears at the beginning or end of the string, or two separators appear consecutively in the string. Specify False (0) to not include empty list elements for leading, trailing, and consecutive separators. Defaults to False.

newlineAsSeparator

Note: This parameter is new with Release 6.

Boolean. Optional. Specify True (1) to treat the newline as a separator. Specify False (0) to not treat the newline as a separator. Defaults to True.

Return value

explodedString

Text list. A list containing each element found in string, or each date found in dateRange.

Language cross-reference

Split function of LotusScript language

Examples: @Explode

Given a semicolon (;) as the default display separator:

1. This example returns a list containing "Weekly," "Status," and "Report" if the content of the Topic field is "Weekly Status Report"; "Weekly,Status,Report"; "Weekly;Status;Report"; or "Weekly," "Status," and "Report" separated by newlines.
   @Explode(Topic)
2. This example returns a list containing "Weekly," "Status," and "Report" if the content of the Topic field is "Weekly+Status+Report"; or "Weekly," "Status," and "Report" separated by newlines.
   @Explode(Topic; "+&")
3. This example specifies the default separators but inserts empty elements for leading, trailing, and consecutive separators.
   @Explode(Topic; ",;\n\n"; @True)
4. This example specifies the defaults for parameters 2 and 3, but does not treat newlines as separators.
   @Explode(Topic; ",;\n\n"; @False, @False)
5. This example returns "Please send resume + references" if the content of the entry field is: "Please send resume & references".
   @Implode( @Explode( entry; ";\n\n" ); ";" )
6. This example returns Attendance grows at UCLA; Pomona Colleges; and USC if the content of the Headline field is "Attendance grows at UCLA, Pomona Colleges, and USC."
   @Explode(Headline; ","
7. This example returns 4 if the content of the Country field is "Mexico, Guatemala, Costa Rica, El Salvador".
8. This example returns 07/02/96; 07/03/96; 07/04/96; 07/05/96.

\[\text{@Explode([07/02/96 - 07/05/96])}\]

9. This example returns 07/01/94; 05/01/94; 10/01/94; 10/02/94; 10/03/94; 04/01/94; 04/02/94; 04/03/94. Note the order in which the dates are returned: single date-time values are returned first, followed by exploded date-time ranges. The return value is a text list.

\[\text{@Explode([07/01/94]:[10/01/94 - 10/03/94]:[05/01/94]:[04/01/94 - 04/03/94]}\]

10. You might want users to be able to enter a range of dates into an editable, multi-value, time-date field called Duration and display them in a computed, multi-value, text field called Days. Give the Duration field the following input-translation formula: \text{@Date(Duration)}. Give the Days field the following formula: \text{@Explode(Duration)}. Users can enter dates into the Duration field in this format: 04/16/71-04/18/71.

---

**@Failure**

Returns a message that you supply; when used in an input validation formula, @Failure displays its message whenever the entered value does not meet the validation criteria.

**Syntax**

@Failure( string )

**Parameters**

string

Text. The error message you want returned.

**Return value**

string

Text. The error message.

**Usage**

@Failure is mainly used in input validation formulas for editable fields, although you can also use it in agents and form formulas. When @Failure is used in formulas other than input validation formulas, the result is the input string; Lotus Notes/Domino displays no prompts or messages.

**Language cross-reference**

On Error statement of LotusScript language

GetErrorMessage method of LotusScript ODBC Connection, Query, and ResultSet classes

Java NotesError and NotesException classes

**Examples: @Failure**

This example show an input validation formula. It returns the error message "Area codes have only 3 digits" if the user enters a number greater than 999 in the field named AreaCode.

\[\text{@If(AreaCode<1000;@Success;@Failure("Area codes have only 3 digits"))}\]

---

**@False**

Returns the number 0.
Syntax
@False

Return value
Returns the number 0.

Usage
This function is equivalent to @No.

Language cross-reference
Built-in constants of LotusScript language

Examples: @False
1. This example returns 0.
   @False
2. This example returns 0 if the value in the field named Cost is 100 or less.
   @If(Cost>100;@True;@False)

FIELD
A reserved word that is necessary when you are assigning values to fields that are stored in a document (as opposed to temporary fields). You can use FIELD to change the contents of an existing field or to create new fields.

You cannot use the FIELD reserved word within an @function. Use @SetField instead.

Syntax
FIELD fieldName := value ;

CAUTION:
When you use FIELD to create a new field in existing documents, make sure that you do not duplicate the name of a field that already exists.

In some cases, action formulas that don’t evaluate to a result (for example, a button formula) return a "No Main or Selection expression in formula" error message. You can supply a value such as an empty string (""), or you could provide an expression at the end of the formula, as shown below:

SELECT @All

Usage
This reserved word is most useful in agent, button, hotspot, and action formulas. It does not work in column, selection, hide-when, window title, or form formulas.

Language cross-reference
ReplaceItemValue method of LotusScript NotesDocument class
replaceItemValue method of Java Document class
Examples: FIELD

1. There is a field named Company on a form. When users compose documents with this form, they enter the name of the company in this field. You can write the following filter, which adds "Inc." to the contents of the Company field:
   \[ \text{FIELD Company} := \text{Company} + "\text{, Inc."}; \]
2. Alternatively, you can create a new field called CompanyName in the form to hold the name of the company plus "Inc.", by assigning it the following formula:
   \[ \text{FIELD CompanyName} := \text{Company} + "\text{, Inc."}; \]
3. To delete the field CompanyName from an existing set of documents, you can use the following formula:
   \[ \text{FIELD CompanyName} := @\text{DeleteField}; \]
4. To assign a value to a field and use it in an @function:
   \[ \text{FIELD fullname} := @\text{If(fullname} = ""; \text{firstname} + "" + \text{lastname};\text{fullname}) \]

@FileDir

Returns the directory portion of a path name, that is, the path name minus the file name.

Note: This @function is new with Release 6.

Syntax

@FileDir( pathname )

Parameters

pathname

Text. Path name of a file.

Return value

directory

Text. The directory part of the path name.

Usage

The directory part of a file name is everything to the left of the file name as demonstrated below:

<table>
<thead>
<tr>
<th>Path name</th>
<th>Directory portion</th>
</tr>
</thead>
<tbody>
<tr>
<td>europe.dat</td>
<td>Null</td>
</tr>
<tr>
<td>c:\europe.dat</td>
<td>c:\</td>
</tr>
<tr>
<td>c:\market\data\europe.dat</td>
<td>c:\market\data\</td>
</tr>
<tr>
<td>\europe.dat</td>
<td>\</td>
</tr>
</tbody>
</table>

Use \[ @\text{Right} \] with the path name and @FileDir to extract the file name.

Examples: @FileDir

1. This computed field formula returns the directory part of the file named by Pathname.
   \[ @\text{FileDir(Pathname)} \]
2. This computed field formula returns the file name part of the file named by Pathname.
   \[ @\text{Right(Pathname}; @\text{FileDir(Pathname)} \]
3. This agent formula displays the directory part of the current database name.
   @Prompt([OK];
   "File directory";
   @FileDir(@Subset(@DbName; -1)))

4. This agent formula displays the file name part of the current database name.
   pathname := @Subset(@DbName; -1);
   @Prompt([OK];
   "File name";
   @Right(pathname; @FileDir(pathname)))

@FloatEq

Compares two numbers for equality within a confidence range.

Note: This @function is new with Release 6.

Syntax
@FloatEq( number ; number ; confidenceRange )

Parameters
number

Number. Any numeric value.

confidenceRange

Number. Optional. The amount within which the numbers must be equal. Defaults to 0.0001.

Return value
flag

Boolean.

• Returns 1 (True) if the difference of the numbers is less than the confidence range.
• Returns 0 (False) if the difference of the numbers exceeds or is equal to the confidence range.

Usage
@FloatEq is helpful in dealing with the inexactness of floating point operations.

Examples: @FloatEq
This action formula compares the fields SpecifiedLength and MeasuredLength, and displays a message if the fields are not within 0.01.
   @If(@FloatEq(SpecifiedLength; MeasuredLength; 0.01); "";
      @Prompt([OK]; "Length is out of spec";
             @Text(MeasuredLength)))

@FontList

Provides a list of available fonts on the Lotus Notes client where this @function is executed.

Note: This @function is new with Release 5.

Syntax
@FontList
Return value

availablefont

Text list. All the available font names. For "Default Serif", "Default Sans Serif", and "Default Monospace" fonts, @FontList returns alias values as follows:

- Default Serif, 0
- Default Sans Serif, 1
- Default Monospace, 4

Usage

Use @FontList as the keyword formula for a list field to display a list of fonts that are available on to the users.

This function does not work in Web applications.

Examples: @FontList

1. The following formula returns a list of font names such as "Arial" : "Courier" : "Default Sans Serif 1" : "Default Serif 0" : "Default Monospace 4" : "Times New Roman........"

   @FontList

2. The following code, when added to the "Change font" hotspot button on a form enables the user to apply the font they select from the "fontList" listbox field to the text in the "Body" rich text field.

   @Command([EditGoToField];"Body");
   @Command([EditSelectAll]);
   @Command([TextSetFontFace];fontList)

   To display the fonts available to the user in the fontList field, set the field Type to a list field, by choosing Listbox or Dialog list, for example. On the control tab of the Field Properties box, select the Use formula for choices option and enter the following formula:

   @FontList

@For

Executes one or more statements iteratively while a condition remains true. Executes an initialization statement. Checks the condition before executing the statements and executes an increment statement after executing the statements.

Note: This @function is new with Release 6.

Syntax

@For(initialize ; condition ; increment ; statement ; ... )

Parameters

initialize

A statement that assigns an initial value to a variable in condition.

condition

Expression that returns a value of True (1) or False (0).

increment

A statement that changes the initialized variable, typically incrementing it.
A formula language statement. The maximum number of statements you can include is 252.

**Return value**

*true*

True (1) unless an error occurs during execution of the condition. An "unexpected data type" error occurs if the conditional expression results in a non-numeric value.

**Usage**

@For executes the initialize statement once. Next @For evaluates the condition. If the condition is True (1), @For executes the statements, executes the increment statement, and evaluates the condition again. If the condition is False (0), @For terminates.

**Tip:** If you are looping through a field containing a list, be sure the “Allow multiple values” check box is selected in the Field Properties box for the list field.

The formula engine exits a formula or breaks an infinite loop if the time spent performing the iterations exceeds the standard timeout value allowed for an operation.

For other iterative statements, see @DoWhile and @While.

**Language cross-reference**

While statement of LotusScript language

For statement of LotusScript language

**Examples: @For**

1. This agent displays the elements of the Categories field one at a time.
   ```
   @For(n := 1;
       n <= @Elements(Categories);
       n := n + 1;
       @Prompt([OK]; "Category " + @Text(n); Categories[n]))
   ```

2. This computed field formula concatenates the list elements in the fname and lname fields:
   ```
   @For(n := 1; n <= @Elements(fname); n := n + 1;
       full := @If(@Length(fname[n]) > @Length(temp); fname[n] + " " + lname[n];
                   temp := (fname[n] + " " + lname[n])))
   ```
   If fname contains: "Catherine";"Patricia";"Maureen" and lname contains: "Rolling";"Kearns";"Legacy", the result is: "Catherine Rolling;Patricia Kearns;Maureen Legacy." If fname and lname each contain a different number of elements, be sure to include the field that has fewer elements in the @Elements function or an "Array index out of bounds" error results.

3. This computed field formula displays the longest name in a text list of poets names stored in the poets field. If the poets field contains "T.S. Eliot";"Dorothy Parker";"Edna St. Vincent Millay";"e.e. cummings": this field displays Edna St. Vincent Millay.
   ```
   temp := "";
   @For(n := 1; n <= @Elements(poets); n := n + 1;
       @If(@Length(poets[n]) > @Length(temp);
           temp := poets[n];temp))
   ```
@FormLanguage
Returns the language of the current form.

Note: This @function is new with Release 5.

Syntax
@FormLanguage

Return value
language

Text. The language specified for the current form. Format of this information is based on RFC1766.

Usage
If the database contains multilingual forms, you can specify the language for each form.

Language cross-reference
Language property of LotusScript NotesName class

Language property of Java Name class

Examples: @FormLanguage
The following formula returns "en-US" when used on a form designed in English(United States).
@FormLanguage

@GetAddressBooks
Returns a list of the address books associated with a client (if the current database is local) or server.

Note: This @function is new with Release 6.

Syntax
@GetAddressBooks( [ options ] )

Parameters
[ options ]

Keyword. You must select one of the following keywords as an argument for this @function:

[TITLES]

Returns the file names of the address books associated with the current database.

[FIRSTONLY]

Displays only the first database in the returned text list of address book names.

Return value
address books
Text or text list. When the current database is hosted by a server, returns the address books that exist on that server. When the current database is hosted locally, returns the address books listed in the NAMES= line of the notes.ini file for that client.

Examples: @GetAddressBooks
1. This code populates the chooseAddress listbox field options with "names.nsf" and "AcmeNorthServer!names.nsf" when added to the Use formula for choices textbox on the Control tab of the field properties box if the database containing the chooseAddress field is running on the Acme server:
   @GetAddressBooks([TITLES])
2. This code populates the chooseAddress listbox field options with "names.nsf" when added to the Use formula for choices textbox on the Control tab of the field properties box if the database containing the chooseAddress field is running on the Acme server:
   @GetAddressBooks([FIRSTONLY])

@GetCurrentTimeZone

Returns the current operating system’s time zone settings in canonical time zone format.

Note: This function is new with Release 6.

Syntax
@GetCurrentTimeZone

Return value
fieldValue

Canonical time zone representing the time zone settings of the operating system.

Usage
Use with the @TimeZoneToText function to translate the time zone value returned into a readable time zone value.

Examples: @GetCurrentTimeZone
1. This code, when added as the default value for a field, returns Z=5$DO=1$DL=4 1 1 10 - 1$ZX=10$ZN=Eastern if the current operating system’s time zone setting is GMT-05:00 Eastern Time.
   @GetCurrentTimeZone
2. This code, when added as the default value for a field, returns (GMT-5:00) Eastern Time (US & Canada).
   @TimeZoneToText(@GetCurrentTimeZone)

@GetDocField

Given the unique ID of a document, returns the contents of a specific field on that document. The document must reside in the current database.

Syntax
@GetDocField( documentUNID ; fieldName )

Parameters
documentUNID
Text. The unique ID of a document. \texttt{\@DocumentUniqueID} specifies the unique id of the current document. To specify the unique id of the parent document, you can use \$Ref as the parameter. \$Ref is the name of the special field on a response document that stores the unique id of its parent.

\textit{fieldName}

Text. The name of a field on the document, enclosed in quotation marks. If you store the field name in a variable, omit the quotation marks here.

\textbf{Return value}

\textit{fieldValue}

Text or text list; number or number list; time-date or time-date range. The contents of the field on the specified document. Returns null if the UNID or field name is invalid.

\textbf{Usage}

This function does not work in column or selection formulas.

\textbf{Language cross-reference}

FieldGetText method of LotusScript NotesUIDocument class

GetItemValue method of LotusScript NotesDocument class

gGetItemValue method of Java Document class

\textbf{Examples: \@GetDocField}

1. You have a discussion database with main topics and responses. In each response, you want to store the subject of the parent document in a field called OriginalSubject. You want OriginalSubject to change if the subject of the main topic changes, so you write this formula for it. \$Ref is a special field on a response document that contains the unique ID of the parent document.

\begin{verbatim}
@If(@IsNewDoc; Subject; @GetDocField($Ref; "Subject"))
\end{verbatim}

2. The following formula can run a scheduled agent to update the contents of a child document, based on the parent.

\begin{verbatim}
FIELD Project:=@GetDocField($Ref; "Project");
\end{verbatim}

3. The following formula runs a scheduled agent to update the contents of one document based on the content of another. The documents don’t need to be parent and child. For example, these could be two parent documents or two child documents.

\begin{verbatim}
FIELD Body:=@GetDocField("BB791838F30B20ED852567BA0064DDAF"; "Body");
\end{verbatim}

\textbf{@GetField}

Returns the value of a specified field.

\textbf{Note:} This \texttt{\@} function is new with Release 6.

\textbf{Syntax}

\texttt{@GetField (fieldName )}
Parameters

fieldName

Text. The name of a field in the current document.

Return value

value

The value of the specified field.

Usage

This @function returns null if the field does not exist.

This @function is useful in writing portable code and in other instances where you want to vary the name of the field.

If the field specified in fieldName is marked to Allow multiple values, this function returns the first value only.

Language cross-reference

FieldGetText method of LotusScript NotesUIDocument class

GetItemValue method of LotusScript NotesDocument class

getItemValue method of Java Document class

Examples: @GetField

1. This code, when added to a computed field on a form and accessed on the Web or in Notes, displays Hello if "Hello" is the default value of the greeting field.
   @GetField("greeting")

2. This computed field formula multiplies values from two fields. The fields are named by adding suffixes to the name of the current field.
   @GetField(@ThisName + "_Quantity") * @GetField(@ThisName + "_Cost")

@GetFocusTable

Returns the name, current row, or current column of the table that is in focus.

Note: This @function is new with Release 6.

Syntax

@GetFocusTable ( [ tableInfoRequest ] )

Parameters

[ tableInfoRequest ]

Keyword. The table information to be returned. One of the following:

[CELLROW]

Returns the current row number starting at "1"; returns "0" if a table is not in focus.
[CELLCOLUMN]

Returns the current column number starting at "1"; returns "0" if a table is not in focus.

[TABLENAME]

Returns the table name (Name/Id under the Table Programming tab in Table Properties); returns a null string if a table is not in focus or the table has no name.

**Return value**

`tableInfo`

Text. The requested information.

**Usage**

This `@function` works in the OnHelp event of a form. It is triggered by selecting Help - Context Help from the menu bar or pressing F1 when:

- The cursor is in a field in a table cell when the document is in edit mode
- Text or an object is selected in a table cell and the document is in read mode

When focus is in the tab of a tabbed table, [CELLCOLUMN] always returns zero.

You cannot use this `@function` in Web applications.

**Examples: @GetFocusTable**

This onHelp event returns the name, row, and column of a table that is currently in focus.

```plaintext
row := @GetFocusTable([CELLROW]);
@If(row = "0"; @Prompt([OK]; "*No table*"; "Not in a table");
@Do(column := @GetFocusTable([CELLCOLUMN]);
name0 := @GetFocusTable([TABLENAME]);
name := @If(name0 = "; "No name"; name0);
@Prompt([OK]; "*" + name + ";*";
"Row " + row + ", column " + column))
```

---

[@GetHTTPHeader](#)

In a Web application, returns the value of an HTTP header from the browser client request being processed by the server.

**Note**: This `@function` is new with Release 6.

**Syntax**

```plaintext
@GetHTTPHeader(requestHeader)
```

**Parameters**

`requestHeaderField`

Text. The name of a request-header field, for example, "From," "Host," or "User-Agent."
Return value

requestHeaderValue

Text. The value of the request-header field, or null if the field does not exist.

Usage

@GetHTTPHeader is useful in formulas that run in the context of a browser.

The Notes client always returns null for this formula.

See http:/www.w3.org/Protocols for the specification of a request header.

See @SetHTTPHeader for setting a response header value.

Language cross-reference

Table of CGI variables

GetURLHeaderInfo method of LotusScript NotesDatabase class

getURLHeaderInfo method of Java Database class

Headers property of LotusScript NotesMIMEEntity class

Headers property of Java MIMEEntity class

Examples: @GetHTTPHeader

The examples below return header field content based on this standard HTTP request:

GET /yourdb.nsf/All?OpenView HTTP/1.0
User-Agent: Mozilla 4.0 (X; I; Linux-2.0.35i586)
Host: mylinuxbox.ibm.com
Accept: image/gif, image/jpeg, */*

1. This computed field formula returns "Mozilla 4.0 (X; I; Linux-2.0.35i586."
   @GetHTTPHeader("User-Agent")

2. This computed field formula returns "mylinuxbox.ibm.com."
   @GetHTTPHeader("Host")

@GetIMContactListGroupNames

Returns the group names in the Instant Messaging Contact List.

Note: This function is new with Release 6.5.

Syntax

@GetIMContactListGroupNames

Return value

nameList

Text list. All group names in the Instant Messaging Contact List.
Usage
You cannot use this function in Web applications.

@GetPortsList
Returns a list of enabled or disabled ports.

Syntax
@GetPortsList( [ portType ] )

Parameters
[ portType ]
Keyword. Must be enclosed in brackets. Use one of the following keywords:

[ENABLED]
Returns a list of currently enabled ports.

[DISABLED]
Returns a list of currently disabled ports.

Return value
portsList
Text list. Each port name is one element of the list.

Usage
@GetPortsList is used by the Public and Personal Address books to determine the list of available ports for each Location record. You can then select a port from that list.

This function does not work in column formulas, selection formulas, or selective replication formulas.

You cannot use this @function in Web applications.

Examples: @GetPortsList
1. This example returns Lan0;TCP;AppleTalk if those are the currently enabled ports.
   @GetPortsList([Enabled])
2. This example returns COM1;COM2 if those are the currently disabled ports.
   @GetPortsList([Disabled])

Note: The text list uses the multi-value separator specified for the current field, or the list separator specified for the current column in a view.

@GetProfileField
Retrieves a field from a profile document, and caches the field value for the remainder of the session.

Syntax
@GetProfileField( profilename ; filename ; uniqueKey )
Parameters

`profilename`

Text. The name of the profile document that contains the field you want to access.

`fieldname`

Text. The name of the field you want to access.

`uniqueKey`

Text. Optional. The unique key that identifies a profile document.

Return value

`fieldvalue`

The value of the field.

Usage

This function does not work in column, hide-when, section editor, or view selection formulas. You can use it in toolbar buttons or agents.

You can use this function on the Web. Use `@SetProfileField` to create a profile document in a Web application. If no profile document by the name specified as the first parameter to `@SetProfileField` exists, Notes creates one. This function enables you to access the fields in that profile document.

Language cross-reference

FieldGetText method of LotusScript NotesUIDocument class

GetItemValue method of LotusScript NotesDocument class

getItemValue method of Java Document class

Examples: `@GetProfileField`

1. This example gets the contents of the "ProfileCategories" field of the "Interest Profile" document.
   ```
   @GetProfileField("Interest Profile";
   "ProfileCategories")
   ```

2. This example gets the contents of the "ProfileCategories" field of the "Interest Profile" document for the profile document for Monday if weekday has "Monday" as its default value.
   ```
   @GetProfileField("Interest Profile";
   "ProfileCategories"; "weekday")
   ```

3. This example gets two field values from the age and job fields of the "userprofile" profile document and displays them vertically in a view column. The following code is in the "profile" field of a user-accessible form:
   ```
   @Explode(@GetProfileField("userprofile"; "age"; @UserName):@GetProfileField("userprofile"; "job"; @UserName); ":")
   ```
   The column formula of the view that displays these two values has its view properties set to Lines per row = 2 and Shrink rows to content and column properties set to Multi-value separator = New Line. The column value formula is the following:
   ```
   @Trim(profile)
   ```
   The @Explode function replaces the semicolon (;) that returns to the profile field with the colon (:) which indicates to the @Trim function in the column formula that the two values are a text list.
4. The following code, when added to the Update Info action button in a Web form, retrieves the user name and address information from the user’s profile document ("Profile") and fills the name and address fields on the Web form with that information:

```plaintext
tempName := @GetProfileField("Profile";"userName";@UserName);
tempAddress := @GetProfileField("Profile";"userAddress";@UserName);
@SetDocField(@DocumentUniqueID;"name";tempName);
@SetDocField(@DocumentUniqueID;"address";tempAddress);
```

---

**@GetViewInfo**

Returns a view attribute.

**Note:** This function is new with Release 6.

**Syntax**

```
@GetViewInfo ( [ attribute ] ; column )
```

**Parameters**

[ attribute ]

Keyword. Must be enclosed in brackets. Use one of the following keywords:

- [CalendarViewFormat]
  Returns as a numeric value the number of days displayed: 1, 2, 5, 7, and so on. Applies to calendar views only.

- [ColumnValue]
  Returns as a text value the value of a column for the current row. Requires the second parameter.

- [IsCalViewTimeSlotOn]
  Returns @True if time slots are displayed on the lefthand side, @False otherwise. Applies to 1-day and 2-day calendar views only.

**column**

Number. Required for [ColumnValue]; otherwise does not apply. The column number starting with 0 for the first column and counting hidden columns.

**Return value**

value

The value of the attribute as described above.

**Usage**

This function is not available in selection or column formulas and will return null if used there.

Look at the view design or design synopsis to determine column numbers. You cannot count columns by looking at a view in the Notes client or a browser.

- If you look at the view in Domino Designer, the first column is 0, the second column is 1, and so on.
• If you look at a view synopsis (Database - Design Synopsis), subtract 1 from the column number listed there.

Examples: @GetViewInfo
1. This hide-column formula hides the "End date" column in a calendar view if time slots are enabled or the format is for 30 days.
   @GetViewInfo(IsCalViewTimeSlotOn) = @True |
   @GetViewInfo(CalendarViewFormat) = 30

2. This hide-action formula hides an action if column 4 (a hidden column) has the programmatically assigned value "Task."
   @GetViewInfo(ColumnValue; 4) = "Task"

3. This default field value on a form displays the value of the third column in the view used with that form.
   @GetViewInfo(ColumnValue; 2)

@HardDeleteDocument
In an agent that runs a formula, @HardDeleteDocument permanently removes the document currently being processed from the database if the database has soft deletions enabled. If the database does not have soft deletions enabled, @HardDeleteDocument performs the same action as @DeleteDocument.

Note: This function is new with Release 5.0.1.

Syntax
@HardDeleteDocument

Usage
This function is intended only for use in agents that run formulas; it has no effect when run elsewhere.

To mark a document for deletion from an icon, view, or form action, use @Command[EditClear].

To soft delete a document, use @DeleteDocument.

To create an agent that deletes documents from a database without using a formula, use the Simple action "Delete from Database."

You cannot use this @function in Web applications.

Language cross-reference
Delete Document URL command
DeleteDocument method of LotusScript NotesUIDocument class
RemovePermanently method of LotusScript NotesDocument class
removePermanently method of Java Document class

@HashPassword
Encodes a string.

Note: This @function is new with Release 6.
**Syntax**

@HashPassword( string )

**Parameters**

string

Text. The string that you want to encode.

**Return value**

encodedString

Text. The passed-in string, double digest encoded for maximum security.

**Usage**

Some person records contain a $SecurePassword hidden field, which is double digest encoded in the @HashPassword format. If this field is not present in the record, the digest is encoded in the @Password format. @HashPassword creates a more secure password than the @Password function does.

---

**@Hour**

Returns the number of the hour in the specified time–date.

**Syntax**

@Hour( timeDateValue )

**Parameters**

timeDateValue

Time-date. The value with the hour that you want to extract.

**Return value**

hour

Number. A number representing the hour contained in timeDateValue. Hours are represented as 0 through 23 for 12 AM through 11 PM. Returns -1 if the time-date provided contains only a date and not a time value.

**Language cross-reference**

Hour function of LotusScript language

**Examples: @Hour**

1. This example returns 9.
   @Hour([9:30])

2. This example returns 8 if the time in the Date field is 8:56:34 AM.
   @Hour(Date)

3. This example returns 20 if a Date field is made up of the date and time: 7/30/90 8:56:34 PM.
   @Hour(Date)

4. This example returns 3 if the current document was created on 2/15/92 at 3:00:12 A.M.
   @Hour(@Created)
**@If**

Evaluates a condition; if the condition is True, Lotus Notes/Domino performs the action appearing immediately after that condition, and stops. If the condition is False, Lotus Notes/Domino skips to the next condition and tests it, and so on. If none of the conditions is True, Lotus Notes/Domino performs the else_action.

**Syntax**

@If(condition1 ; action1 ; condition2 ; action2 ; ... ; condition99 ; action99 ; else_action )

**Parameters**

*condition*

Expression that returns a Boolean. If this expression returns True, action is performed. If it’s False, Lotus Notes/Domino skips to the next condition, if there is one. Otherwise, Lotus Notes/Domino performs else_action.

*action*

An action to be performed or a value to be returned if the governing condition returns True.

*else_action*

An action to be performed or a value to be returned if none of the conditions returns True.

**Usage**

In its simplest form, the If statement looks like this: @If( condition ; action ; else_action ).

You can list up to 99 conditions and corresponding actions, followed by just one action to be performed when all the conditions are False. As soon as a condition evaluates to True, Lotus Notes/Domino performs the associated action and ignores the remainder of the @If statement.

Lotus Notes/Domino accepts the form @If( condition ), with only one condition and no action, but does not perform any action based on the condition.

If you compare a field to a value (for example, Year > 1995) and the field is unavailable, the comparison is False. However, you should check for fields that may not be present with @IsUnavailable.

**Language cross-reference**

If...Then...Else statement of LotusScript language

If...GoTo statement of LotusScript language

If...Then...ElseIf statement of LotusScript language

**Examples: @If**

1. This formula tests the single value in the CostOfGoods field. If the value is greater than or equal to 12.45, the condition is True, and the string "Over Budget" is returned. If the value is less than 12.45, the condition is False and the string "Bill of Materials OK" is returned.

   @If(CostOfGoods>=12.45;"Over Budget";"Bill of Materials OK")

2. In this example, if CostOfGoods is less than 12.45, the null string is returned.

   @If(CostOfGoods=12.45;"Over Budget";;")
3. In this example, @If looks at the value in the CostOfGoods field; if the value is greater than 12.45, then the string "Over Budget" is returned; if not, Notes skips to the next condition. The second condition also evaluates the CostOfGoods field and if the value is less than 12.45, then the condition is True and Notes returns the string "Bill of Materials OK." If the value is neither greater than nor less than 12.45, Notes moves on to the "else" action specified, and the string "Estimate Right on Target" is returned.

@If(CostOfGoods<12.45; "Bill of Materials OK"; "Estimate Right on Target")

4. Notes first checks that the document has never been saved; if the condition is True, the value in the field NewNoteTitle is returned. If the first condition is False, Notes then checks whether the view is the Author View; if this is True, the value in the field ByAuthorTitle is returned. If both conditions are False, the value in the field StandardTitle is returned.

@If(@IsNewDoc; NewNoteTitle; @ViewTitle = "Author View"; ByAuthorTitle; StandardTitle)

5. This code, when used as the Input Validation for the phoneNumber field prohibits a form from being saved until the user enters a value in the phoneNumber field. This formula demonstrates how to test more than one statement, since a phone number is only required if the contactMe field is set to Yes, indicating that the user wants to be contacted.

@If((contactMe="Yes") & (@ThisValue = "") ; @Failure("You must enter a value in "); @ThisValue; @Success)

Using @ThisValue and @ThisName instead of hard-coding in field names enables you to copy and paste this code into all the other fields you want to require input for, the firstName and lastName fields, for example.

@IfError

Returns a null string (""") or the value of an alternative statement if a statement returns an error.

Note: This @function is new with Release 6.

Note: This @command is obsolete in Release 7.

Syntax

@IfError( statement1 ; statement2 )

Parameters

statement1

A formula statement. This statement executes first.

statement2

Optional. A formula statement. This statement, if available, executes if the first statement returns an error.

Return value

statementReturn

• Returns the value of the first statement if it is not an error.

• Returns the value of the second statement if the value of the first statement is an error and the second statement is supplied.

• Returns a null string ("") if the value of the first statement is an error and the second statement is omitted.
Usage

Use $Error in the second statement to get the value of the error.

This command should be replaced with the following series of commands.

\[
\text{result} := \text{statement1};
\]
\[
\text{@If(}@\text{IsError(result);}\text{result2}:@\text{result})
\]

Language cross-reference

On Error statement of LotusScript language

Examples: @IfError

1. This agent tests the return value of an @DbLookup statement for an error. If the @DbLookup statement causes an error, the agent returns the text "Not available."

\[
\text{FIELD Phone} :=
\]
\[
\text{@IfError(}
\]
\[
\text{@DbLookup("","Snapper");"names.nsf";"People";}
\]
\[
\text{@Right(Name;}" ") + "+" + @Left(Name;" ");"OfficePhoneNumber";}
\]
\[
"Not available"
\]

This agent does the same thing, using @If instead of @IfError.

\[
\text{result} := \text{@DbLookup("":"Snapper");"names.nsf";"People";}
\]
\[
\text{@Right(Name;}" ") + "+" + @Left(Name;" ");"OfficePhoneNumber";}
\]
\[
\text{FIELD Phone} := \text{@If(}@\text{IsError(result);}"Not available";\text{result})
\]

2. The following code, when added to a Computed for display field, displays the price of the product entered in the "product" field, after a page refresh. Enter the text, "Enter product name here" as the default value for the product field. Once a user enters a product name in the product field and presses F9, the price is extracted from the Goods view, which contains the product name in the first sorted column and its price in the second column. If the product name is not recognized or any other error occurs during the lookup, the message, "Unable to retrieve requested price. Aborting lookup" displays. You could add a Get Price action button that contains the code: @Command([ViewRefreshFields]) to prompt the user to refresh the page.

\[
\text{@If(product="Enter product name here";0;}\]
\[
\text{@IfError(}@\text{DbLookup("";"product/server";"filename\productdatabase.nsf";"Goods";product)}
\]
\[
2;}"Unable to retrieve requested price. Aborting lookup")
\]

3. This formula, when added to the 'Apply font' hotspot button, applies the font a user selects from the "fonts" Dialog list field to the text the user enters or highlights in the "Body" Rich Text field. The "fonts" field contains an @FontList function in the Use formula for choices box in its Field Properties box, which displays a list of available fonts. If no font was selected from the "fonts" field, an error message displays which instructs the user to select one.

\[
\text{@Command([EditGoToField]);"Body";}
\]
\[
\text{@Command([EditSelectAll]);}
\]
\[
\text{@IfError(}@\text{Command([setTextFontFace];fonts});@Prompt([Ok];"Error encountered";"You must select a font first")}
\]

@Implode

Concatenates all members of a text list and returns a text string.

Syntax

@Implode( textlistValue ) or @Implode( textlistValue ; separator )

Parameters

textlistValue
Text or text list. List containing the items you want to concatenate into a single string. If you send a single piece of text instead of a list, @Implode returns the text unaltered.

**separator**

Text. Used to separate the values in the concatenated string. If you don’t specify a separator, a space is used.

**Return value**

implodedString

Text. String containing each member of textListValue, separated by separator.

**Language cross-reference**

Join function of LotusScript language

**Examples: @Implode**

1. This example returns Minneapolis Detroit Chicago if the contents of the City field are "Minneapolis";"Detroit";"Chicago."
   ```java
   @Implode(City)
   ```

2. This example returns Minneapolis, Detroit, Chicago if the contents of the City field are "Minneapolis";"Detroit";"Chicago."
   ```java
   @Implode(City;",")
   ```

3. This example returns European Capitals/Bonn : European Capitals/Lisbon : European Capitals/Madrid if the contents of the Categories field are European Capitals, and the content of the Cities field is a list consisting of Bonn, Lisbon, and Madrid.
   ```java
   @Implode(Categories + "/" + City ; ": ")
   ```

**@InheritedDocumentUniqueID**

The unique ID of the current document’s inheritance parent. See [@DocumentUniqueID](#) for a description of unique IDs.

**Syntax**

@InheritedDocumentUniqueID

**Usage**

This function works in a document being created with a form with field values inherited from the selected document.

In documents that do not inherit, @InheritedDocumentUniqueID returns the same value as @DocumentUniqueID.

**Language cross-reference**

ParentDocumentUNID property of LotusScript NotesDocument class

ParentDocumentUNID property of Java Document class
Examples: @InheritedUniqueID

1. In a response document, this field formula creates a doclink to the selected main topic document. The response document must be created with a form that inherits values from the selected main topic document.

   @InheritedDocumentUniqueID

   The next time you access this response document, the field still contains a doclink to the parent document.

2. This example, when used as the default value for a Computed When Composed field on a response form, displays the contents of the "userName" field from the parent document. The form must have the "Formulas inherit values from selected document" property selected on the Default tab of the Form Properties box.

   @GetDocField(@InheritedDocumentUniqueID;"userName")

3. This example, when added to a hotspot button in a response form, displays the contents of the "userName" field in the parent document in a message box. This button only works if the response document has already been saved. On the Paragraph Hide When tab of the Button Properties box, select the "Hide paragraph if formula is true" property. Add @IsNewDoc as the formula.

   id := @Text($REF);
   @Prompt([OK];"Parent field value";@GetDocField(id;"userName"))

@Integer

Truncates the values in a number or number list at the whole number, leaving off any decimals. The values in the resulting list are separated using the multi–value separator that is selected for display in the field containing the formula.

Syntax

@Integer( numberValue )

Parameters

numberValue

Number or number list. The value(s) you want to truncate.

Return value

truncatedValue

Number or number list. The truncated value(s).

Usage

When using this function with a number list, the list concatenation operator takes precedence over any other operators. Negative numbers must be enclosed in parentheses.

For more information, see "List Operator" in "Formula Language Rules."

Language cross-reference

CInt function of LotusScript language

CLng function of LotusScript language
**Examples: **@Integer

1. This example returns 123;789.
   
   @Integer(123.001 : 789.999)

2. This example returns 127580;5;7341 if the numbers in the Sales, CommissionRate, and Commission fields are 127580.35, 5.75, and 7341.62015, respectively.
   
   @Integer(Sales:CommissionRate:Commission)

3. This example returns 3.
   
   @Integer(3.12)

4. This example returns 6.
   
   @Integer(6.735)

---

**@IsAgentEnabled**

Indicates whether or not a scheduled agent is enabled.

**Syntax**

@IsAgentEnabled( agent )

**Parameters**

*agent*

Text. The name of the agent. Not case-sensitive.

**Return value**

*flag*

Number

- 1 (True) indicates that the agent is enabled
- 0 (False) indicates that the agent is disabled, or that an agent by that name does not exist

**Usage**

A database must be open. If a database is not open, returns 0.

@IsAgentEnabled returns 1 for macros created in Lotus Notes Release 3, and for any agents that are not scheduled.

@IsAgentEnabled does not work in column or selection formulas and is not intended for use in window title or form formulas.

You cannot use this function in Web applications.

**Language cross-reference**

IsEnabled property of LotusScript NotesAgent class

IsEnabled property of Java Agent class

---

**Examples: **@IsAgentEnabled

This example returns 1 if the UnderCover agent is enabled; otherwise, it returns 0.

@IsAgentEnabled( "UnderCover" )
@IsAppInstalled
Indicates whether the specified type of application is installed.

Note: This @function is new with Release 5.

Syntax
@IsAppInstalled( application )

Parameters
application
Text. Specify "Designer" to check if the Domino Designer is installed on the system, or "Admin" to check if the Domino Administrator is installed.

Return value
flag
Boolean
• True indicates that the specified application is installed
• False indicates that the specified application is not installed

Usage
This @function is generally used in hide-when formulas.

@IsAvailable
Checks a document for the existence of a field.

Syntax
@IsAvailable( fieldName )

Parameters
fieldName
Field. The name of a field. Do not treat the name as text. Enter the exact name of the field and do not enclose it in quotes.

Return value
flag
Boolean
• 1 (True) indicates that the field is contained in the document
• 0 (False) indicates that the field is not contained in the document

Usage
Use @IsAvailable to provide a default value for documents created with forms that do not include a field name.

This function can be used with select and column formulas using Summary fields only. Non-Summary fields are not available.
For information on creating a field in an existing document if it does not exist, see the FIELD keyword.

**Language cross-reference**

HasItem method of LotusScript NotesDocument class

hasItem method of Java Document class

**Examples: @IsAvailable**

1. This formula returns the value of the Dept field if it exists in the document, otherwise it returns Consultant.
   
   ```
   @If(@IsAvailable(Dept);Dept;"Consultant")
   ```

2. This formula returns the value of the field named Topic if it exists in the document, otherwise it returns the value contained in the field named Subject.
   
   ```
   @If(@IsAvailable(Topic);Topic;Subject)
   ```

3. This formula, when added to a hotspot button, checks for the existence of the Priority field in a form, then sets its value if found or creates a new Priority field and sets its value, if not found.
   
   ```
   @If(@IsAvailable(Priority);@SetField("Priority";"High");FIELD Priority := "High")
   ```

   **Note:** If you create the field using this formula, it is not visible on the form, but you can get its value using the @GetField function. Be sure you use the correct spelling and capitalization when checking for the field in the document.

**@IsCategory**

In a column formula, returns a specified string if any item in the row of a view is defined as a category.

**Syntax**

```
@IsCategory @IsCategory( trueString ) @IsCategory( trueString ; falseString )
```

**Parameters**

*trueString*

Text. A string to return if an item in the view row is a category.

*falseString*

Text. A string to return if no item in the row is a category.

**Return value**

*specifiedString*

Text.

No parameters:

- * (asterisk) indicates that the entry is a category
- If the entry is a document, returns nothing

Single *trueString* parameter:

- Returns *trueString* instead of *

Both *trueString* and *falseString* parameters
• Return `trueString` instead of *
• Return `falseString` instead of nothing

**Usage**

Use `@IsCategory` only in column formulas.

This function only looks at the columns to its right, so be sure to place it to the left of the categorized column to which you are referring.

This function does not work in Web applications running version 4.5.

**Examples: @IsCategory**

1. This example returns * if the row is a category, or nothing if the row is not a category.
   
   `@IsCategory`

2. This example returns C if the row is a category, or nothing if the row is not a category.
   
   `@IsCategory("C")`

3. This example returns Y if the row is a category, or N if the row is not a category.
   
   `@IsCategory("Y";"N")`

**@IsDB2**

Given a server and filename or server and replica ID, indicates if the specified database is backed by DB2 or not.

**Note:** This @function is new with Release 7.

**Syntax**

`@IsDB2(server : file)`

`@IsDB2(server ; replicaID)`

**Parameters**

*server*

Text. The name of the server. Use an empty string (""") to indicate the local computer.

*file*

Text. The path and file name of the database. Specify the database path and file name using the appropriate format for the operating system.

*replicaID*

Text. The replica ID of the database.

**Return value**

*flag*

Boolean

• 1 (True) indicates that the specified database is backed by DB2
• 0 (False) indicates that the specified database is not backed by DB2
This function will return an error via @Error if:

- The server cannot be reached
- The database specified in file or replicaID cannot be found

**Usage**

This function works in all contexts where @function use is supported, including view selection formulas, column formulas, and from the Web.

If the database has been replicated to a local replica, and an empty string is specified for the server parameter, @IsDB2 will produce an error on the replica. For this reason, it is extremely important to use @Error processing with @IsDB2 when using relative paths.

**Examples: @IsDB2**

1. This formula returns 0 (False), since the local names.nsf database is not in DB2:
   ```
   @IsDB2("":names)
   ```
2. These formulas both return DB2 information about the current database:
   ```
   @IsDB2(@DbName)
   @IsDB2("":")
   ```
3. This formula returns 1 if FRITES.NSF in the MAIL directory on the server Belgium is DB2 backed. Otherwise it returns 0.
   ```
   @IsDB2( "Belgium" : "mail\frites.nsf" )
   ```
4. This formula returns DB2 information about a database using its replica ID instead of its file name:
   ```
   @IsDB2("Cheshire";"85255600:00576146")
   ```
5. This example of a column formula first uses @IsDB2 to find out if the local database referenced in the dbname field of the document is a DB2 database, so that a more meaningful error message may be displayed from @DB2Schema:
   ```
   result1 := @IsDB2("":dbname);
   result2 := @DB2Schema("":dbname)
   @If(@IsError(result1);"Unable to find database or lost server connection";
     @If(@IsError(result2);
       "Unable to find database or lost server connection";result2);
     "Not a DB2 database")
   ```

---

**@IsDocBeingEdited**

Checks the current status of the document and returns 1 (True) if the document is being edited; otherwise returns 0 (False).

**Syntax**

@IsDocBeingEdited

**Return value**

*flag*

Boolean

- 1 (True) indicates that the document is being edited
- 0 (False) indicates that the document is not being edited

**Usage**

This function does not work in column, selection, agent, form, or view action formulas. It’s intended for use in button, hide-when, field, and form action formulas.
Language cross-reference
EditMode property of LotusScript NotesUIDocument class

Examples: @IsDocBeingEdited
This code, when added to an action button, checks whether the current document is in edit mode. If it’s not, it changes the document’s mode to edit in order to execute the @DocLock function, which requires that the current document be in edit mode. It then locks the current document.

```lotus
@If(@IsDocBeingEdited; @True; @Command([EditDocument]; 1));
@DocLock([Lock])
```

@IsDocBeingLoaded
 Checks the current status of the document and returns 1 (True) if the document is being loaded into memory for display; otherwise returns 0 (False).

**Syntax**

```
@IsDocBeingLoaded
```

**Return value**

*flag*

Boolean

- 1 (True) indicates that the document is actually being loaded into memory
- 0 (False) indicates that the document is not being loaded into memory

**Usage**

Use function in field and form formulas. It does not work in toolbar button, selection, column, agent, section editor, hotspot, form action, or view action formulas.

**Examples: @IsDocBeingLoaded**

1. This example returns 1 when the document is being loaded into memory.

   ```
   @IsDocBeingLoaded
   ```

2. This example returns 0 when the document is saved.

   ```
   @IsDocBeingLoaded
   ```

3. This example, when used in a computed field named "Editors," displays the contents of $UpdatedBy when the document is being loaded. When the user recalculates the field (by pressing F9), the field displays the user’s name as the current editor, followed by previous editors’ names. When the document is saved, the value of "Editors" remains unchanged.

   ```
   @If(@IsDocBeingLoaded;$UpdatedBy;
   @IsDocBeingRecalculated;("Current Editor - " + @UserName):$UpdatedBy;Editors)
   ```

@IsDocBeingMailed
 Checks the current status of the document and returns 1 (True) if the document is being mailed; otherwise, returns 0 (False).

**Syntax**

```
@IsDocBeingMailed
```
Return value

Flag

Boolean

• 1 (True) indicates that the document is actually being mailed

• 0 (False) indicates that the document is not being mailed

Usage

Use @IsDocBeingMailed in field formulas. This function is useful for calculating the number of times the user has mailed a document, including the number of times the document has been forwarded. It is also useful for changing a document during mailing; for example, you can change a document’s form when it is mailed.

It has limited usefulness in toolbar button, hotspot, and form action formulas. This function does not work in column, selection, agent, window title, form, or view action formulas.

You cannot use this function in Web applications.

Examples: @IsDocBeingMailed

1. This example returns 1 when the document is being mailed.
   @IsDocBeingMailed

2. This example returns 0 before and after the document has been mailed.
   @IsDocBeingMailed

3. This formula is used in a field to calculate the number of times a document has been mailed. When this formula is provided as the definition of a computed field called TimesMailed, the field is initialized to 0 (since the document has not been mailed). TimesMailed is incremented with every mail operation, so if the document has been mailed once, the contents become 1, and the count increases by one each time the document is mailed.
   @If(@IsUnavailable(TimesMailed);0;
   TimesMailed+@IsDocBeingMailed)

@IsDocBeingRecalculated

Checks the current status of the document and returns 1 (True) if the document is being recalculated; otherwise, returns 0 (False).

Syntax

@IsDocBeingRecalculated

Return value

Flag

Boolean

• Returns 1 (True) only when the fields on the document are actually being recalculated

• Returns 0 (False) when the fields on the document are not currently being recalculated

Usage

Use @IsDocBeingRecalculated in field formulas. It has limited usefulness in toolbar button, hotspot, and form action formulas. This function does not work in column, selection, agent, window title, form, or view action formulas.
If you are using this function to increment a counter, the count increases by one every time the user recalculates the fields on a form.

**Examples: @IsDocBeingRecalculated**
1. This example returns 1 while the document is being calculated or recalculated.
   ```plaintext
   @IsDocBeingRecalculated
   ```
2. This example returns 0 before and after the document is calculated or recalculated.
   ```plaintext
   @IsDocBeingRecalculated
   ```
3. This example can be used in a time–date field to display different dates under different circumstances. The formula causes the current time–date to be displayed if the document is recalculated during the editing process; otherwise, it displays the original creation date of the document.
   ```plaintext
   @If(@IsDocBeingRecalculated;@Now;@Created)
   ```

**@IsDocBeingSaved**
Checks the current status of the document and returns 1 (True) if the document is being saved; otherwise, returns 0 (False).

**Syntax**
```plaintext
@IsDocBeingSaved
```**Return value**

- **flag**
  - Boolean
  - Returns 1 (True) only when the fields on the document are actually being saved
  - Returns 0 (False) when the fields on the document are not currently being saved

**Usage**
Use @IsDocBeingSaved in field formulas. It has limited usefulness in toolbar button, hotspot, and form action formulas. This function does not work in column, selection, agent, window title, form, or view action formulas.

If you are using this function to increment a counter, the count increases by one every time the user saves the form.

**Examples: @IsDocBeingSaved**
1. This example returns 1 while the document is being saved.
   ```plaintext
   @IsDocBeingSaved
   ```
2. This example returns 0 before or after the document is saved.
   ```plaintext
   @IsDocBeingSaved
   ```
3. This formula sets the field named Readers, which is a Reader Names field, to Admins when the document is saved. Otherwise, it sets the Readers field to the value already in the field. This type of formula is useful for changing the Read Access of a document after it has been composed and saved.
   ```plaintext
   @If(@IsDocBeingSaved;"Admins";Readers)
   ```

**@IsDocTruncated**
Indicates whether the current document has been truncated.
Syntax
@IsDocTruncated

Return value
flag

Boolean
• Returns 1 (True) if the document is missing some data
• Returns 0 (False) if the entire document is present

Usage
You typically use @IsDocTruncated in a column formula to display the truncated document indicator. You can also use @IsDocTruncated in a variety of other formulas, including toolbar buttons, hide-when formulas, section editors, window title formulas, field formulas, form formulas, column formulas, selection formulas, and agents.

Documents may be truncated during database replication. Depending upon the type of truncation, a document can be missing an attached file, an OLE object, large rich text fields, or non-summary items.

If the document is truncated, you can obtain the entire document by choosing Action - Retrieve Entire Document, either in the background or during the next replication of the database. You cannot edit a truncated document.

Language cross-reference
Abstract property of LotusScript NotesReplication class

IsAbstract property of Java Replication class

Examples: @IsDocTruncated
This code, when added to a column formula, displays a negative (-) icon if the document was truncated in replication. The column must be set to Display values as icons in the Column Properties box.
@If(@IsDocTruncated;97;0)

@IsEmbeddedInsideWCT
Indicates whether any part of the current Notes session is embedded inside of Workplace client.

Note: This @function is new with Release 7.

Syntax
@IsEmbeddedInsideWCT

Return value
flag

Boolean
• Returns 1 (True) if any Notes content is open inside Workplace client (even bookmarks)
• Returns 0 (False) if this is a standalone Notes environment, or if in the current session, no Notes content is open inside Workplace client
Usage
This function allows you to globally enable or disable Notes UI elements depending on whether or not the Notes content is being viewed from within Workplace client.

You cannot use this function in Web applications.

Language cross-reference
IsEmbeddedInsideWCT property of the LotusScript NotesUIWorkspace class

Examples: @IsEmbeddedInsideWCT
If the Notes content is being viewed from within Workplace client, you will use Instant Messaging. If not, you will use Sametime. You create actions to use both, and want to show only the one that is appropriate for the current environment.

This function, placed in the Hide When tab of the Sametime action, will hide the action when any Notes content is open within Workplace client.

@IsEmbeddedWithinWCT

This function, placed in the Hide When tab of the Instant Messaging action, will hide the action in a standalone Notes environment or when no Notes content is open within Workplace client.

@Not(@IsEmbeddedWithinWCT)

@IsError
Returns 1 (True) if the value is an @ERROR value, returns 0 (False) if not an error.

Syntax
@IsError( value )

Parameters
value
Number. Can be a literal value or a field name containing data of type Number.

Return value
flag

Boolean
• Returns 1 (True) if the value is an @ERROR value
• Returns 0 (False) if not an error

Language cross-reference
Err function of LotusScript language

Error function of LotusScript language

Java NotesError and NotesException classes

Examples: @IsError
1. This example returns 1.
@IsError(1/0)
2. This example returns 0.
   @IsError(1/2)
3. This formula checks to see if there is an @ERROR in the Price field, and returns "There is an error in the price field" if it encounters an error; otherwise it returns 0.
   @If(@IsError(Price);
   Failure("There is an error in the price field"); Success)
4. This agent tests the return value of an @DbLookup statement for an error. If the @DbLookup statement causes an error, the agent returns the text "Not available."
   FIELD Phone := @DbLookup(""; "Snapper" : "names.nsf"; "People";
   @Right(Name; " ") + "; " + @Left(Name; " "); "OfficePhoneNumber");
   @If(@IsError(Phone); "Not available")

@IsExpandable
In column formulas, returns a specified string if a row in a view can be expanded.

Syntax
@IsExpandable @IsExpandable( trueString ) @IsExpandable( trueString ; falseString )

Parameters
trueString
Text. A string to return if the view row is expandable.

falseString
Text. A string to return if the view row is not expandable.

Return value
specifiedString
Text

No parameters:
• Returns + (plus) if the entry is expandable
• Returns - (minus) if the entry is not expandable

Single trueString argument:
• Returns the trueString instead of + if the entry is expandable
• Returns nothing if it is not expandable

Both trueString and falseString:
• Return trueString instead of +
• Return falseString instead of -

Usage
Use @IsExpandable in column formulas to indicate whether the current level of documents can be expanded. This function does not work in any other formula.

In the single parameter and two parameter forms, you should limit the string to a single character, especially if the lines already have a lot of text in them.
**Examples: @IsExpandable**

1. This example returns + if the document or category is expandable, or - if it is not expandable.
   
   @IsExpandable(1)

2. This example returns & if the document or category is expandable.
   
   @IsExpandable("&")

3. This example returns Y if the document or category is expandable, or N if it is not expandable.
   
   @IsExpandable("Y","N")

---

**@IsMember**

Indicates if a piece of text (or a text list) is contained within another text list. The function is case-sensitive.

**Syntax**

@IsMember( textValue ; textListValue ) @IsMember( textListValue1 ; textListValue2 )

**Parameters**

*textValue*

Text.

*textListValue*

Text list.

*textListValue1*

Text list.

*textListValue2*

Text list.

**Return value**

*flag*

Boolean

- Returns 1 (True) if the *textValue* is contained in *textListValue*
- Returns 0 (False) if not
- If both parameters are lists, returns 1 if all elements of *textListValue1* are contained in *textListValue2*

**Usage**

In processing lists, @IsMember differs from a simple = test. An = returns True if the pair-wise comparison of two entities has even one member; that is, it is not empty.

For more information on pair-wise operations, see the topic "Operations on lists" in the "Formula Language Rules" chapter.

@IsMember returns True only if the first parameter is an exact match, or a subset of the second parameter which is a list.
Language cross-reference
Like operator of LotusScript language

FindString method of LotusScript NotesUIDocument class

Examples: @IsMember
1. This example returns 1.
   @IsMember("computer":"printer":"computer":"monitor")
2. This example returns 0.
   @IsMember("computer":"Notes":"Notes":"printer":"monitor")
3. This example returns 1 if R&D is in the list in the Department field, returns 0 if R&D is not in the list.
   @IsMember("R&D":Department)
4. This example returns 1, since Fred is a subset of a list.
   @IsMember("Fred": "Barney":"Wilma":"Fred")

@IsModalHelp
Indicates whether the current document is a modal Help document.

Syntax
@IsModalHelp

Return value
flag

Boolean
• Returns 1 (True) if the document is a modal Help document
• Returns 0 (False) if the document is not a modal Help document

Usage
A modal Help document is a document that displays as a dialog box that you must dismiss before you can access any other currently open windows. Use @IsModalHelp to determine modality so you can execute a formula only when the document is (or isn’t) a modal Help document.

You cannot use this function in Web applications.

Language cross-reference
FieldHelp property of LotusScript NotesUIDocument class

@IsNewDoc
For a document being edited, indicates if the document has been saved to disk.

Syntax
@IsNewDoc

Return value
flag

Boolean
Returns 1 (True) if the document being edited has not yet been saved to disk
Returns 0 (False) if the document has been saved

Usage
This function evaluates the current state of the document when it is used in toolbar button, hide-when, section editor, window title, field, form, and form action formulas.

This function returns 0 if the document has not yet been saved, regardless of how the document was created. It always returns a 0, even if the document has been saved, when used in column, selection, agent, and view action formulas.

Language cross-reference
IsNewDoc property of LotusScript NotesUIDocument class
IsNewNote property of LotusScript NotesDocument class
IsNewNote property of Java Document class

Examples: @IsNewDoc
1. When used in a window title formula, this formula returns New Document while the document is composed the first time. When a document is opened after it has been saved, this formula returns the value of the Subject field.
   @If(@IsNewDoc; "New Document"; Subject)
2. If a new document is being created, the string New General Information appears in the window title. When an existing document is opened, the string General Information for, then the contents of the field EmpName, a slash, and then the contents of the field EmpNumber appear in the window title.
   @If(@IsNewDoc; "New General Information"; "General Information for" + EmpName + "/" + EmpNumber)

@IsNotMember
Indicates if a text string (or a text list) is not contained within another text list. The function is case-sensitive.

Syntax
@IsNotMember( textValue ; textListValue ) or @IsNotMember( textListValue1 ; textListValue2 )

Parameters

**textValue**
Text.

**textListValue**
Text list.

**textListValue1**
Text list.

**textListValue2**
Text list.
**Return value**

*flag*

Boolean
- Returns 1 (True) if the `textValue` is not contained in `textListValue`
- Returns 0 (False) if it is contained
- If both parameters are lists, returns 1 if all elements of `textListValue1` are not contained in `textListValue2`

**Usage**

In processing lists, `@IsNotMember` differs from a simple `!=` test. `!=` returns True if the pair-wise comparison of two entities has no entities in common and False only if the pair-wise comparison of the two entities finds all pairs to be equal. `@IsNotMember` does not perform a pair-wise comparison, but tests each element in `textListValue1` against all the elements in the `textListValue2` and returns False if it is equal to one of them.

For further details on pair-wise operators, see “Operations on lists” in the “Formula Language Rules” chapter.

`@IsNotMember` returns True only if no member of the first argument is contained in the second argument.

**Examples: @IsNotMember**

1. This example returns 0.
   
   ```
   @IsNotMember("computer";"printer";"computer";"monitor")
   ```

2. This example returns 1 if R&D is not in the list of values in the field name Department; returns 0 if R&D is in the list.
   
   ```
   @IsNotMember("R&D";Department)
   ```

3. This example returns Marketing in the Dept field if the current user is not contained in the list in the SalesDepartment field; otherwise Sales is returned in the Dept field.
   
   ```
   FIELD Dept:=@If(@IsNotMember(@Username;SalesDepartment);"Marketing";"Sales");
   ```

4. This example returns 1 if both the [WebTeam] and [ManageFiles] roles are assigned to the current user; it returns 0 if only one or neither of the roles is assigned to the user.
   
   ```
   @IsNotMember([WebTeam]:"[ManageFiles]";@UserRoles)
   ```

---

**@IsNull**

Tests for a null value. Returns true only if a value is a single text value that is null, otherwise it returns false. This function also returns false if the value is an error.

**Note:** This `@function` is new with Release 6.

**Syntax**

```
@IsNull( value )
```

**Parameters**

*value*

Any data type. Any value.
Return value

flag

Boolean
• Returns 1 (True) if the value is a text value that is null
• Returns 0 (False) if the value is not a text value, not null, or is an error

Usage
This function is useful for checking for empty fields before using them in other functions in which they might generate errors.

Examples: @IsNull
This function, when used as a field formula, finds the square root of each element in the text list in the OriginalList field. @IsNull is first used to test the OriginalList field to ensure that it contains a value and prevents the formula from calculating the square roots if it does not. If OriginalList contains 4: 25, the result is 2; 5. If OriginalList is a null field, the result is a null field, not an error.

@If(@IsNull(OriginalList); @Nothing;
@Transform(OriginalList; "x";
@If(x >= 0; @Sqrt(x); @Nothing)))

@IsNumber
Indicates if a given value is a number (or a number list).

Syntax
@IsNumber( value )

Parameters
value
Any data type. Any value.

Return value
flag

Boolean
• Returns 1 (True) if the value is a number or a number list
• Returns 0 (False) if the value is not a number or a number list

Usage
This is a useful function for checking to see that you have assigned field data types correctly.

Language cross-reference
TypeName function of LotusScript language
DataType function of LotusScript language
IsNumeric function of LotusScript language
Examples: @IsNumber
1. This example returns 1.
   @IsNumber(123)
2. This example returns 0.
   @IsNumber(@Created)
3. This example returns 1.
   @IsNumber(-345:2.78:997:.7)
4. This example returns 1 if the field named CostCenters contains a list of number values; returns 0 if the list contains at least one text string.
   @IsNumber(CostCenters)

@IsResponseDoc
Indicates whether a document is a response to another document.

Syntax
@IsResponseDoc

Return value
flag

Boolean
• Returns 1 (True) if the document is a response document
• Returns 0 (False) if the document is not a response document
• Returns 0 for new documents, since @IsResponseDoc doesn’t recognize a document type until after the document is saved

Usage
A response document is one that was composed with a form which has a type of either Response or Response to Response. The designer uses the Form InfoBox to specify the type.

Language cross-reference
IsResponse property of LotusScript NotesDocument
IsResponse property of Java Document

Examples: @IsResponseDoc
This example returns Response if the document is a response; Topic if the document is not a response.
@If(@IsResponseDoc;"Response";"Topic")

@IsText
Indicates whether a value is text (or a text list).

Syntax
@IsText( value )
Parameters
value
Any data type. Any value.

Return value
flag
Boolean
• Returns 1 (True) if the value is text or a text list
• Returns 0 (False) if the value is not text or a text list

Language cross-reference
TypeName function of LotusScript language
DataType function of LotusScript language

Examples: @IsText
1. This example returns 1.
   @IsText("Blanchard & Daughters")
2. This example returns 1 if the field named Branch Offices contains the text string list "New Orleans";"Houston";"Dallas";"Mobile."
   @IsText(BranchOffices)

@IsTime
Indicates whether a value is a time-date (or a time-date list).

Syntax
@IsTime( value )

Parameters
value
Any data type. Any value.

Return value
flag
Boolean
• Returns 1 (True) if the value is a time–date or a time–date list
• Returns 0 (False) if the value is not a time-date or a time-date list

Language cross-reference
TypeName function of LotusScript language
DataType function of LotusScript language
Examples: @IsTime
1. This example returns 1 if the DueDate field contains a time–date value.
   @IsTime(DueDate)
2. This example returns 0.
   @IsTime(123)

@IsUnavailable
Indicates whether a field name exists in a document.

Syntax
@IsUnavailable( filename )

Parameters
filename
The name of a field. Do not enclose the name in quotes.

Return value
flag
Boolean
• Returns 1 (True) if the field name is not contained in the document
• Returns 0 (False) if the field name is contained in the document

Usage
Use @IsUnavailable to provide default values for fields in documents created with forms that do not include a particular field name.

CAUTION:
Do not confuse @IsUnavailable with @Unavailable. @Unavailable deletes fields and can cause serious damage if used unintentionally in place of @IsUnavailable.

Language cross-reference
HasItem method of LotusScript NotesDocument class

hasItem method of Java Document class

Examples: @IsUnavailable
This example returns Consultant if the field Dept does not exist; if Dept does exist, the value contained in Dept is returned.
@If(@IsUnavailable(Dept);"Consultant";Dept)

@IsValid
Executes all validation formulas within the current form.

Syntax
@IsValid
Return value
flag

Boolean
• Returns 1 (True) if all validation formulas resolve to True
• Returns 0 (False) if all validation formulas do not resolve to True

Usage
Use @IsValid to initiate execution of all of a form’s validation formulas, as if the document were being saved.

If validation formulas are added to a form after some documents have already been saved, you can use @IsValid in a macro to determine which of those documents need corrections.

Language cross-reference
ComputeWithForm method of LotusScript NotesDocument class
computeWithForm method of Java Document class

Examples: @IsValid

You edit a form after it’s been in use for a while, and insert validation formulas into several fields. Now you want to test existing documents to be sure they meet the field validation requirements. You can create an additional field on the form and use this formula to indicate whether the document needs corrections:
@If(@IsValid;"Ok";"Needs corrections")

@IsVirtualizedDirectory

Indicates whether virtualized directories are enabled for the current server.

Note: This @function is new with Release 6.

Syntax
@IsVirtualizedDirectory

Return value
flag

Boolean
• Returns 1 (True) if virtualized directories are enabled
• Returns 0 (False) if virtualized directories are not enabled

Examples: @IsVirtualizedDirectory

This computed field displays the name of the current server if virtualized directories are enabled and a message otherwise.
@If(@IsVirtualizedDirectory; @UserName;
@Return("Virtualized directories not enabled"))

@Keywords

Given two text lists, returns only those items from the second list that are found in the first list.
Syntax
@Keywords( textList1 ; textList2 ) or @Keywords( textList1 ; textList2 ; separator )

Parameters

textList1
Text list. A list of items.

textList2
Text list. A list of items that you want to compare to textList1.

separator
Text. One or more characters to be used as delimiters between words. @Keywords considers each character (not the combination of multiple characters) to be a delimiter. For example, defining separator as ", , " (period, space, comma) tells the function to separate the text at each period, space, and comma into separate words.

When you do not specify a separator, the following word delimiters are used by default:

? . ; : [] () (question mark, period, space, comma, exclamation point, semicolon, colon, (brackets, parentheses, braces, quotation mark, and angle brackets)

A null separator, represented by an empty string (""), tells the function to use no delimiters.

Return value

resultTextList
Text list. When a separator is in effect, either by default or specification, @Keywords parses textList1 into words delimited by the separator and returns any word that exactly matches a keyword in textList2. When no separator is in effect (when you specify a null separator), @Keywords returns any sequence of characters in textList1 that matches a keyword specified in textList2.

Note: With Release 6, the order of the words returned in resultTextList match the order of textList1. Prior to Release 6, the order of the words returned in resultTextList matched the order of textList2. To retain the pre-Release 6 ordering of the resultTextList, prepend the formula with another @Keywords function as follows:

@Keywords(textList2;@Keywords(textList1;textList2))

Usage

When a keyword that you specify in textList2 is the very first word in the string you are searching AND you specify separators, @Keywords returns null. To prevent this behavior, prepend textList1 with one of the separators. For example, if you want to find the keyword, Sally, in a text list that contains employee names and positions, use the following formula:

@Keywords( " " + " ,"Mary Halen, Director of Sales":" ,"Sally Hall, VP of Marketing": " ,"Joe Halzy, Order entry": "Sally": " ,")

This formula returns Sally. Note that one of the formula’s separators, the space(" "), is prepended to textList1. This behavior does not occur if you accept the default separators or specify a null separator.

If one of the strings in textList2 contains any of the default delimiters, @Keywords will not return it. To search for Harvard University, for example, add a null separator to the formula. This tells @Keywords to
search for any sequence of characters. If you do not specify a separator, you allow the default delimiters to act. @Keywords does not return Harvard University because when it parses textList1, it breaks the phrase into two separate words, Harvard and University, where it finds the space, which is a default delimiter.

When using the quotation mark separator ("), precede it with a backslash (\) to indicate that the quotation mark is a text constant.

This function is case-sensitive; you must standardize the case of textList1 and textList2 if you want case to be ignored (use @LowerCase, @ProperCase or @UpperCase).

Examples: @Keywords

1. This formula returns Harvard;Yale.

   @Keywords(@ProperCase("EPA Head speaks at Harvard and yale":"The UCLA Chancellor Retires":
   "Ohio State wins big game":"Reed and University of Oregon share research facilities");
   "Harvard":"Brown":"Stanford":"Yale":"Vassar":"UCLA")

2. This formula returns "", a null string.

   @Keywords("EPA Head speaks at Harvard,yale":"UCLA Chancellor Retires";
   "Ohio State wins big game":"Reed and University of Oregon share research facilities";
   "harvard":"brown""stanford":"yale":"ucla")

3. This formula returns Harvard;Yale. It searches textList1 for the textList2 keywords that follow either a comma or a space.

   @Keywords("EPA Head speaks at Harvard, Yale hosts her next month":"UCLA Chancellor Retires":
   "Ohio State wins big game":"Reed and University of Oregon share research facilities";
   "Harvard":"Brown":"Stanford":"Yale";"UCLA";""

4. This formula returns Harvard;Yale University;UCLA.

   @Keywords("EPA Head speaks at Harvard, Yale University hosts her next month":
   "UCLA Chancellor Retires":"Ohio State wins big game":"Reed and University of Oregon share research facilities";
   "Harvard":"Brown";"Stanford":"Yale University";"UCLA";"

5. This formula returns Mary Jones. when used in the "Result" field on a form that also contains the "Applicants" field, which has a default value of: "Mary Jones.;"John Chen.;"Miguel Sanchez.".

   @Keywords(Applicants;"Mary Jones.;";"

6. This formula returns Mary Jones. when used in the "Result" field on a form that also contains the "Applicants" field, which has a default value of: "Mary Jones., who works downtown, is being interviewed on Friday.;"John Chen.;"Miguel Sanchez.".

   @Keywords(";" + Applicants;"Mary Jones.;";"

7. This formula returns book.

   @Keywords("<booklist> XML tag that represents a list of our books.";
   "<book> XML tag that represents a book.";"<sale> XML tag that represents the sale price of a book.";"book";"<>")

8. If list1 contains "guava";"eggplant";"date";"cherry";"banana";"apple" and list2 contains
   "apple";"banana";"date," this formula, when triggered from a Release 6 or later client, returns date,
   banana, apple. When triggered from a pre-Release 6 client, it returns apple, banana, date.

   @Keywords(list1;list2)

9. If list1 and list2 contain the same text lists as in the previous example, this formula returns apple,
   banana, date from all versions of Notes.

   @Keywords(list2;@Keywords(list1;list2))

@LanguagePreference

Returns user’s specified preferred language setting.

Note: This function is new with Release 5.
Syntax
@LanguagePreference ( [ key ] )

Parameters
[ key ]
Keyword. Specify a category for which you would like to get the preferred language. The following categories are available:

[REGION]
Returns preferred language for region.

[CONTENT]
Returns preferred language for database contents.

[ALTERNATENAME]
Returns preferred language for alternate name.

Return value
preferredlanguage
Text or Text list. Language and country code for user’s preferred setting. [REGION] language is set as the default. If @LanguagePreference cannot find the language setting for the specified category, it returns the language for [REGION].

Usage
@LanguagePreference is used to implement mechanisms for handling language-dependent features. A database that is designed to store data in multiple languages can select the language in which the data should be published for each user by using @LanguagePreference[Content].

See @Locale for a list of language codes.

@LanguagePreference supports the Web browser client. When the browser client calls @LanguagePreference, it returns a list of languages specified in the Web browser. This returned list is normalized based on the key parameter of the @function.

Language cross-reference
Language property of LotusScript NotesName class
Language property of Java Name class

Examples: @LanguagePreference
1. The following example returns "fr" if your region language setting is French.
   @LanguagePreference([REGION])
2. The following example returns "en" if you call this function from the Web client and your Web browser’s accept language is "English(United States)."
   @LanguagePreference([ALTERNATENAME])
@LaunchApp
Launches the requested Domino application.

Note: The @function is new with Release 5.

Syntax
@LaunchApp( application )

Parameters
application

Text. The type of application you want to launch. Specify any one of the following:

<table>
<thead>
<tr>
<th>Notes</th>
<th>This launches the Lotus Notes client.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>This launches Lotus Domino Designer, if installed.</td>
</tr>
<tr>
<td>Admin</td>
<td>This launches Lotus Domino Administrator, if installed.</td>
</tr>
</tbody>
</table>

Usage
If the requested application is already running, it will be brought to the front and it will have focus.

This @function is generally used in action formulas.

You cannot use this function in Web applications.

Language cross-reference
Shell function of LotusScript language

@LDAPServer
Returns the URL and port number of the LDAP listener in the current domain. Notes looks for this information in several places, following this search sequence:
1. Searches on the current server.
2. Searches for the notes.ini variable labeled LDAPSERVER=.
3. Queries the administration server, which runs the LDAP service automatically, by default.

Note: This function is new with Release 6.

Syntax
@LDAPServer

Examples: @LDAPServer
The following code, when used as the default value for a field on a form that resides in a database hosted by the ocean/bay server, displays LDAP://ocean.acme.com:379 when the form is accessed by a Web browser. This indicates that the LDAP listener is located at port 379 for the current domain.

@LDAPServer

@Left
Searches a string from left to right and returns the leftmost characters of the string.
Syntax

@Left( stringToSearch ; numberOfChars ) @Left( stringToSearch ; subString )

Parameters

stringToSearch
Text. The string where you want to find the leftmost characters.

numberOfChars
Number. The number of characters to return. If the number is 2, the first two characters of the string are returned; if the number is 5, the first five characters are returned, and so on. If the number is negative, the entire string is returned.

subString
Text. A substring of stringToSearch. @Left returns the characters to the left of subString. It finds subString by searching stringToSearch from left to right.

Return value

resultString
Text. The leftmost characters in stringToSearch. The number of characters returned is determined by either numberOfChars or subString. @Left returns "" if subString is not found in stringToSearch.

Language cross-reference

Left function of LotusScript language

StrLeft function of LotusScript language

Examples: @Left

1. This example returns Len.
   @Left("Lennard Wallace";3)
2. This example returns Lennard Wal if the string in the Contact field is Lennard Wallace.
   @Left(Contact;"la")
3. This example returns Tim if the string in the Author field is Timothy Altman.
   @Left(Author;3)
4. This example returns Timothy if the string in the Author field is Timothy Altman.
   @Left(Author;" ")

@LeftBack

Searches a string from right to left and returns a substring.

Syntax

@LeftBack( stringToSearch ; numToSkip ) or @LeftBack( stringToSearch ; startString )

Parameters

stringToSearch
Text. The string where you want to find the leftmost characters.
numToSkip

Number. Counting from right to left, the number of characters to skip. All the characters to the left of that number of characters are returned. If the number is negative, the entire string is returned.

startString

Text. A substring of stringToSearch. All the characters to the left of startString are returned.

Return value

resultString

Text. The leftmost characters in stringToSearch. The number of characters returned is determined by either numToSkip or startString.

Language cross-reference

StrLeftBack function of LotusScript language

Examples: @LeftBack

1. This example returns Lennard Wall.
   @LeftBack("Lennard Wallace";3)
2. This example returns Lennard.
   @LeftBack("Lennard Wallace";"")
3. This example returns Timothy Alt if the string in the Author field is Timothy Altman.
   @LeftBack(Author;3)

@Length

Returns the number of characters in a text string.

Syntax

@Length( string ) or

@Length( stringList )

Parameters

string

Text. A single string with the length you want to find.

stringList

Text list. A list of strings.

Return value

length

- If the parameter is a text string, @Length returns the number of characters in the specified string, including spaces and punctuation.
- If the argument is a text list, @Length searches the list of strings and returns the number of characters in each string as a number list.
Language cross-reference

Len function of LotusScript language

Examples: @Length

1. This example returns 45.
   @Length("The boy crossed the wide, but gentle, stream.")

2. This example returns the number list 0:5:3, which displays as 0;5;3 if the multi-value separator for the
   field is a semicolon.
   @Length("": "abcde": "xyz")

3. This example returns the number list 16:10:22 if the contents of the fields From, Topic, and Date are
   "Stephen Brewster", "News Flash", and @Now (where the current date is 04/01/2001 16:45:10 PM),
   respectively. The number list displays as 16,10,22 if the multi-value separator for the field is a comma.
   @Length(From: Topic: @Text(Date))

@Like

Matches a string with a pattern. It is case-sensitive and supports the NotesSQL ® ODBC driver.

Syntax

@Like( string ; pattern ) @Like( string ; pattern ; escape )

Parameters

string

Text. The value to be tested to see if it matches pattern.

pattern

Text. The sequence of characters to search for within string. May also contain any of the wildcard
characters listed below.

escape

Text. Optional. A character to use before a wildcard character to indicate that it should be treated literally.

Wildcard characters and symbols are:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Where C is any character. Matches any single, non–special character C.</td>
</tr>
<tr>
<td>_</td>
<td>(Underscore). Matches any single character.</td>
</tr>
<tr>
<td>%</td>
<td>Matches any sequence of zero or more characters.</td>
</tr>
</tbody>
</table>

Return value

flag

Number

• Returns 1 (True) if the pattern matches the string
• Returns 0 (False) if the pattern does not match the string

Language cross-reference

Like operator of LotusScript language
Examples: @Like

1. This example returns 0. The underscore matches only a single character.
   @Like( "A big test" ; "A_test" )

2. This example returns 1. The five underscores match "<space>big<space>."
   @Like( "A big test" ; "A_____test" )

3. This example returns 1. The % matches "A big ." 
   @Like( "A big test" ; "%test" )

4. This example returns 0. @Like is case-sensitive.
   @Like( "A big test" ; "A BIG test" )

5. This example returns 1. The first percent matches "100." The "%/" matches the percent sign because "/" is specified as the escape character. The last percent matches "ement."
   @Like( "A 100% improvement" ; "A ½/½ improv½" ; "/" )

@Ln

Returns the natural log of a number. Natural logs use e (approximately 2.718282) as their base.

Syntax
@Ln( number )

Parameters
number

Number. May be any value greater than 0, and can contain up to 15 decimal places.

Return value
naturalLog

Number. The natural log of number.

Usage
Use @Ln in formulas requiring natural logs, such as compound growth or loss.

@Ln is the inverse of @Exp.

Language cross-reference

Log function of LotusScript language

Examples: @Ln

This example returns 0.693147180559945.
@Ln(2)

@Locale

Returns information about language codes.

Note: This @function is new with Release 5.
Syntax
@Locale( [ action ] )

@Locale( [ action ]; locale-tag )

Parameters
[ action ]

Keyword. One of the following:

[NotesLocale] without locale-tag returns a text list containing all the content language codes.

[NotesLocale] with locale-tag returns a text list or value containing each specified content language code, or a null string if the code is not recognized. A code is recognized if it is exact regardless of case. If the language is recognized but not the country or region, the language code alone is returned.

[AltNameLocale] without locale-tag returns a text list containing all the alternate name language codes.

[AltNameLocale] with locale-tag returns a text list or value containing each specified alternate user language code, or a null string if the code is not recognized. A code is recognized if it is exact regardless of case. The country or region is ignored where it is not part of the alternate user language code (most cases).

[LanguageName] with locale-tag returns a text list or value spelling out the language for each specified language code, or a null string if the code is not recognized.

[CountryName] with locale-tag returns a text list or value spelling out the country or region for each specified language code, or a null string if the code has no country or region, or it is not recognized.

[LocaleName] with locale-tag returns a text list or value spelling out the language and country (or region), if applicable, for each specified language code, or a null string if the code is not recognized. The country or region is in parentheses and immediately (no space) follows the language.

[LocaleName] : [NotesLocale] (concatenating these two keywords) returns a text list containing, for each specified language code, the language name, the country or region name in parentheses, a vertical bar, and the language code. This list can be used in a keyword field where the locale name is the name and the language code is the alias.

[LocaleName] : [AltNameLocale] (concatenating these two keywords) returns a text list containing, for each alternate name language code, the language name, the country or region name in parentheses, a vertical bar, and the language tag. This list can be used in a keyword field where the locale name is the name and the language code is the alias.

locale-tag

Text or text list. A language code or list of language codes.

Supported language codes

<table>
<thead>
<tr>
<th>Tag</th>
<th>Language</th>
<th>Country or region</th>
<th>Locale</th>
</tr>
</thead>
<tbody>
<tr>
<td>af</td>
<td>Afrikaans</td>
<td></td>
<td>Notes</td>
</tr>
<tr>
<td>ar</td>
<td>Arabic</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>ar-AE</td>
<td>Arabic</td>
<td>United Arab Emirates</td>
<td>Notes</td>
</tr>
<tr>
<td>Tag</td>
<td>Language</td>
<td>Country or region</td>
<td>Locale</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>ar-BH</td>
<td>Arabic</td>
<td>Bahrain</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-DZ</td>
<td>Arabic</td>
<td>Algeria</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-EG</td>
<td>Arabic</td>
<td>Egypt</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-JO</td>
<td>Arabic</td>
<td>Jordan</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-KW</td>
<td>Arabic</td>
<td>Kuwait</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-LB</td>
<td>Arabic</td>
<td>Lebanon</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-MA</td>
<td>Arabic</td>
<td>Morocco</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-OM</td>
<td>Arabic</td>
<td>Oman</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-QA</td>
<td>Arabic</td>
<td>Qatar</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-SA</td>
<td>Arabic</td>
<td>Saudi Arabia</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-TN</td>
<td>Arabic</td>
<td>Tunisia</td>
<td>Notes</td>
</tr>
<tr>
<td>ar-YE</td>
<td>Arabic</td>
<td>Yemen</td>
<td>Notes</td>
</tr>
<tr>
<td>be</td>
<td>Byelorussian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>bg</td>
<td>Bulgarian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>ca</td>
<td>Catalan</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>cs</td>
<td>Czech</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>da</td>
<td>Danish</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>de</td>
<td>German</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>de-AT</td>
<td>German</td>
<td>Austria</td>
<td>Notes</td>
</tr>
<tr>
<td>de-CH</td>
<td>German</td>
<td>Switzerland</td>
<td>Notes</td>
</tr>
<tr>
<td>de-DE</td>
<td>German</td>
<td>Germany</td>
<td>Notes</td>
</tr>
<tr>
<td>de-LI</td>
<td>German</td>
<td>Liechtenstein</td>
<td>Notes</td>
</tr>
<tr>
<td>de-LU</td>
<td>German</td>
<td>Luxembourg</td>
<td>Notes</td>
</tr>
<tr>
<td>el</td>
<td>Greek</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>en</td>
<td>English</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>en-AU</td>
<td>English</td>
<td>Australia</td>
<td>Notes</td>
</tr>
<tr>
<td>en-CA</td>
<td>English</td>
<td>Canada</td>
<td>Notes</td>
</tr>
<tr>
<td>en-GB</td>
<td>English</td>
<td>United Kingdom</td>
<td>Notes</td>
</tr>
<tr>
<td>en-HK</td>
<td>English</td>
<td>Hong Kong</td>
<td>Notes</td>
</tr>
<tr>
<td>en-IE</td>
<td>English</td>
<td>Ireland</td>
<td>Notes</td>
</tr>
<tr>
<td>en-IN</td>
<td>English</td>
<td>India</td>
<td>Notes</td>
</tr>
<tr>
<td>en-JM</td>
<td>English</td>
<td>Jamaica</td>
<td>Notes</td>
</tr>
<tr>
<td>en-NZ</td>
<td>English</td>
<td>New Zealand</td>
<td>Notes</td>
</tr>
<tr>
<td>en-PH</td>
<td>English</td>
<td>Philippines</td>
<td>Notes</td>
</tr>
<tr>
<td>en-SG</td>
<td>English</td>
<td>Singapore</td>
<td>Notes</td>
</tr>
<tr>
<td>en-US</td>
<td>English</td>
<td>United States</td>
<td>Notes</td>
</tr>
<tr>
<td>en-ZA</td>
<td>English</td>
<td>South Africa</td>
<td>Notes</td>
</tr>
<tr>
<td>es</td>
<td>Spanish</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>es-AR</td>
<td>Spanish</td>
<td>Argentina</td>
<td>Notes</td>
</tr>
<tr>
<td>es-BO</td>
<td>Spanish</td>
<td>Bolivia</td>
<td>Notes</td>
</tr>
<tr>
<td>es-CL</td>
<td>Spanish</td>
<td>Chile</td>
<td>Notes</td>
</tr>
<tr>
<td>Tag</td>
<td>Language</td>
<td>Country or region</td>
<td>Locale</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>es-CO</td>
<td>Spanish</td>
<td>Colombia</td>
<td>Notes</td>
</tr>
<tr>
<td>es-CR</td>
<td>Spanish</td>
<td>Costa Rica</td>
<td>Notes</td>
</tr>
<tr>
<td>es-DO</td>
<td>Spanish</td>
<td>Dominican Republic</td>
<td>Notes</td>
</tr>
<tr>
<td>es-EC</td>
<td>Spanish</td>
<td>Ecuador</td>
<td>Notes</td>
</tr>
<tr>
<td>es-ES</td>
<td>Spanish</td>
<td>Spain</td>
<td>Notes</td>
</tr>
<tr>
<td>es-GT</td>
<td>Spanish</td>
<td>Guatemala</td>
<td>Notes</td>
</tr>
<tr>
<td>es-HN</td>
<td>Spanish</td>
<td>Honduras</td>
<td>Notes</td>
</tr>
<tr>
<td>es-MX</td>
<td>Spanish</td>
<td>Mexico</td>
<td>Notes</td>
</tr>
<tr>
<td>es-NI</td>
<td>Spanish</td>
<td>Nicaragua</td>
<td>Notes</td>
</tr>
<tr>
<td>es-PA</td>
<td>Spanish</td>
<td>Panama</td>
<td>Notes</td>
</tr>
<tr>
<td>es-PE</td>
<td>Spanish</td>
<td>Peru</td>
<td>Notes</td>
</tr>
<tr>
<td>es-PR</td>
<td>Spanish</td>
<td>Puerto Rico</td>
<td>Notes</td>
</tr>
<tr>
<td>es-PY</td>
<td>Spanish</td>
<td>Paraguay</td>
<td>Notes</td>
</tr>
<tr>
<td>es-SV</td>
<td>Spanish</td>
<td>El Salvador</td>
<td>Notes</td>
</tr>
<tr>
<td>es-US</td>
<td>Spanish</td>
<td>United States</td>
<td>Notes</td>
</tr>
<tr>
<td>es-UY</td>
<td>Spanish</td>
<td>Uruguay</td>
<td>Notes</td>
</tr>
<tr>
<td>es-VE</td>
<td>Spanish</td>
<td>Venezuela</td>
<td>Notes</td>
</tr>
<tr>
<td>et</td>
<td>Estonian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>fi</td>
<td>Finnish</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>fr</td>
<td>French</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>fr-BE</td>
<td>French</td>
<td>Belgium</td>
<td>Notes</td>
</tr>
<tr>
<td>fr-CA</td>
<td>French</td>
<td>Canada</td>
<td>Notes</td>
</tr>
<tr>
<td>fr-CH</td>
<td>French</td>
<td>Switzerland</td>
<td>Notes</td>
</tr>
<tr>
<td>fr-FR</td>
<td>French</td>
<td>France</td>
<td>Notes</td>
</tr>
<tr>
<td>fr-LU</td>
<td>French</td>
<td>Luxembourg</td>
<td>Notes</td>
</tr>
<tr>
<td>gu</td>
<td>Gujarati</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>he</td>
<td>Hebrew</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>hi</td>
<td>Hindi</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>hr</td>
<td>Croatian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>hu</td>
<td>Hungarian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>id</td>
<td>Indonesian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>is</td>
<td>Icelandic</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>it</td>
<td>Italian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>it-CH</td>
<td>Italian</td>
<td>Switzerland</td>
<td>Notes</td>
</tr>
<tr>
<td>it-IT</td>
<td>Italian</td>
<td>Italy</td>
<td>Notes</td>
</tr>
<tr>
<td>ja</td>
<td>Japanese</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>ko</td>
<td>Korean</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>lt</td>
<td>Lithuanian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>lv</td>
<td>Latvian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>mk</td>
<td>Macedonian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>mr</td>
<td>Marathi</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>Tag</td>
<td>Language</td>
<td>Country or region</td>
<td>Locale</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>ms</td>
<td>Malay</td>
<td></td>
<td>AltName</td>
</tr>
<tr>
<td>ms-MY</td>
<td>Malay</td>
<td>Malaysia</td>
<td>Notes</td>
</tr>
<tr>
<td>nl</td>
<td>Dutch</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>nl-BE</td>
<td>Dutch</td>
<td>Belgium</td>
<td>Notes</td>
</tr>
<tr>
<td>nl-NL</td>
<td>Dutch</td>
<td>Netherlands</td>
<td>Notes</td>
</tr>
<tr>
<td>no</td>
<td>Norwegian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>no - NO</td>
<td>Norwegian</td>
<td>Norway</td>
<td>Notes</td>
</tr>
<tr>
<td>ny - NO</td>
<td>Nynorsk</td>
<td>Norway</td>
<td>Notes</td>
</tr>
<tr>
<td>pl</td>
<td>Polish</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>pt</td>
<td>Portuguese</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>pt-BR</td>
<td>Portuguese</td>
<td>Brazil</td>
<td>Notes</td>
</tr>
<tr>
<td>pt-PT</td>
<td>Portuguese</td>
<td>Portugal</td>
<td>Notes</td>
</tr>
<tr>
<td>ro</td>
<td>Romanian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>ro - MD</td>
<td>Romanian</td>
<td>Moldavia</td>
<td>Notes</td>
</tr>
<tr>
<td>ro - RO</td>
<td>Romanian</td>
<td>Romania</td>
<td>Notes</td>
</tr>
<tr>
<td>ru</td>
<td>Russian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>sk</td>
<td>Slovak</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>sl</td>
<td>Slovenian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>sq</td>
<td>Albanian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>sr</td>
<td>Serbian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>sv</td>
<td>Swedish</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>ta</td>
<td>Tamil</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>te</td>
<td>Telugu</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>th</td>
<td>Thai</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>tr</td>
<td>Turkish</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>uk</td>
<td>Ukrainian</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>vi</td>
<td>Vietnamese</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>x-KOK</td>
<td>Konkani</td>
<td></td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>zh-CN</td>
<td>Chinese</td>
<td>China</td>
<td>Notes &amp; AltName</td>
</tr>
<tr>
<td>zh-HK</td>
<td>Chinese</td>
<td>Hong Kong</td>
<td>Notes</td>
</tr>
<tr>
<td>zh-MO</td>
<td>Chinese</td>
<td>Macau</td>
<td>Notes</td>
</tr>
<tr>
<td>zh-SG</td>
<td>Chinese</td>
<td>Singapore</td>
<td>Notes</td>
</tr>
<tr>
<td>zh-TW</td>
<td>Chinese</td>
<td>Taiwan</td>
<td>Notes &amp; AltName</td>
</tr>
</tbody>
</table>

**Language cross-reference**

Language property of LotusScript NotesName class

Language property of Java Name class

**Examples: @Locale**

1. The following formulas return "French."

   @Locale([LanguageName]; "fr")

350  Prrogramming Guide, Volume 1: Overview and Formula Language
2. The following formula returns "Canada."
   \( \text{@Locale([[LanguageName]; "fr-CA"])} \)

3. The following formula returns "French(Canada)."
   \( \text{@Locale([[CountryName]; "fr-CA"])} \)

4. The following formula returns "fr-CA."
   \( \text{@Locale([[LocaleName]; "fr-CA"])} \)

5. The following formula returns "fr."
   \( \text{@Locale([[NotesLocale]; "FR-CA"])} \)

6. The following formula returns a list of all the content language codes.
   \( \text{@Locale([[NotesLocale]])} \)

7. The following formula returns a list of all the alternate user name language tags.
   \( \text{@Locale([[NotesLocale]])} \)

8. The following field keyword formula returns a list of each content language code preceded by its locale name and a vertical bar. This formula allows the user to select from a list of names and stores the corresponding language code (which is an alias to the name).
   \( \text{@Locale([[LocaleName]]: [NotesLocale])} \)
   
   It is equivalent to:
   \( \text{@Locale([[LocaleName]]; @Locale([[NotesLocale]])} + "|" + @Locale([[NotesLocale]]) \)

@Log

Returns the common logarithm (base 10) of any number greater than zero.

Syntax

\( \text{@Log( number )} \)

Parameters

number

Number. Must be greater than zero.

Return value

commonLog

Number. The log of number.

Usage

Use @Log in any formula requiring a common log, such as the formula to calculate the root of a number. @Log is the reciprocal of scientific notation.

Examples: @Log

1. This example returns 0.60205999132796.
   \( \text{@Log(4)} \)

2. This example returns 14.
   \( \text{@Log(1.0E+14)} \)
@LowerCase
Converts the uppercase letters in the specified string to lowercase.

Syntax
@LowerCase(string)

Parameters
string
Text. The string you want to convert to lowercase.

Return value
lowerCaseString
Text. The string, converted to lowercase letters.

Usage
This function is useful when you want to search for a particular value and cannot predict whether it appears in lowercase or uppercase letters, or a combination of the two. You can also use it as an input translation formula to convert the contents of a field to lowercase.

Language cross-reference
StrConv function of LotusScript language
LCase function of LotusScript language

Examples: @LowerCase
1. This example returns juan mendoza.
   @LowerCase("Juan Mendoza")
2. This example returns arm chair if the Furniture field contains "Arm Chair," "Arm chair," "arm chair," or "ARM CHAIR," or any other variation.
   @LowerCase(Furniture)
3. This example returns fletcher if William Fletcher is the name associated with the current hierarchical User ID.
   @LowerCase(@Right(@Name([CN];@UserName); ""))

@MailDbName
Returns the name of the Domino server and the name of the current user's Mail database.

Syntax
@MailDbName

Return value
server; path
Text list with two elements:
• server is the hierarchical name of the server on which the current database resides.
  This @function returns an empty string ("") if:
The database is local
The formula is used in a Scheduled agent running on the server
Use \[@Name\] to extract a part of the name; for example, \[CN\] to extract the common name.

- *path* is the path and file name of the database.

**Usage**
This function works in any formula except column formulas. When a formula runs on a server, the server is considered the current user, so *@MailDbName* returns the name of the server.

The returned value is formatted as a two–item text list specifying the Server;Directory\Database.NSF, as in:

ACMEMAIL;LEGAL\DLEE.NSF

If the database is stored on the user’s own computer, Notes/Domino returns the null string for the server name. For example, dial up user Debbie Lee may keep a local replica of her Mail database on her workstation; when she is set up for workstation–based mail, *@MailDbName* returns:

;DLEE.NSF

This is useful in applications that send mail; for example, you can use it to determine whether the current user is set up for server–based mail, and determine the appropriate course of action based on the result.

You cannot use this function in Web applications.

**Language cross-reference**
GetUserInfo method of LotusScript NotesRegistration class
getUserInfo method of Java Registration class

**Examples: *@MailDbName***

1. This example returns ";MTSEN.NSF" if the user’s mail is in the MTSEN.NSF database stored on the user’s own computer, and the user is set up to use workstation–based mail.

   @MailDbName

2. This example returns "SALES1;MAIL\MTSEN.NSF" if the user's mail is stored in MTSEN.NSF in the MAIL directory on the SALES1 server, and the user is set up to use server–based mail. If the database is stored at the server’s root directory (that is, it is not stored in a subdirectory), the result would be "SALES1;MTSEN.NSF."

   @MailDbName

3. This example returns "MTSEN.NSF," the file name, since this is the last element in the list returned by @MailDbName.

   @Subset(@MailDbName;–1)

**@MailEncryptSavedPreference**
Indicates whether the user has selected “Encrypt saved mail” in the User Preferences dialog box.

**Syntax**

@MailEncryptSavedPreference
**Return value**

flag

Boolean
- Returns 1 (True) if "Encrypt saved mail" is selected
- Returns 0 (False) if "Encrypt saved mail" is not selected

**Usage**

@MailEncryptSavedPreference is used in the Mail template to determine whether to encrypt saved memos. This function is not available in column formulas, selection formulas, or selective replication formulas.

You cannot use this function in Web applications.

**Examples: @MailEncryptSavedPreference**

You design your own Mail form. To determine whether memos created with your form and then saved should be encrypted, use @MailEncryptSavedPreference to determine the current user’s preference. This returns 1 if the “Encrypt saved mail” check box is selected in the User Preferences dialog box, and 0 if the Encrypt saved mail check box is not selected.

@MailEncryptSavedPreference

---

**@MailEncryptSentPreference**

Indicates whether the user has selected “Encrypt sent mail” in the User Preferences dialog box.

**Syntax**

@MailEncryptSentPreference

**Return value**

flag

Boolean
- Returns 1 (True) if "Encrypt sent mail" is selected
- Returns 0 (False) if "Encrypt sent mail" is not selected

**Usage**

@MailEncryptSentPreference is used in the Mail template to determine whether to encrypt sent memos. This function is not available in column formulas, selection formulas, or selective replication formulas.

You cannot use this function in Web applications.

**Language cross-reference**

EncryptOnSend property of LotusScript NotesDocument class

IsEncryptOnSend property of Java Document class

**Examples: @MailEncryptSentPreference**

You can design your own Mail form. To determine whether outgoing memos should be encrypted automatically, use @MailEncryptSentPreference to determine the user’s preference. This returns 1 if the “Encrypt sent mail” check box is selected in the User Preferences dialog box, and 0 if the Encrypt sent mail check box is not selected.
@MailSavePreference

Indicates which option the user has selected for the "Save sent mail" setting in the User Preferences dialog box.

Syntax
@MailSavePreference

Return value
flag

Integer
• Returns 0 if "Don’t keep a copy" is selected
• Returns 1 if "Always keep a copy" is selected
• Returns 2 if "Always prompt" is selected

Usage
@MailSavePreference is used in the Mail template to determine whether to save copies of outgoing memos. This function is not available in column formulas, selection formulas, or selective replication formulas.

You cannot use this function in Web applications.

Language cross-reference
SaveMessageOnSend property of LotusScript NotesDocument class

IsSaveMessageOnSend property of Java Document class

Examples: @MailSavePreference
You design your own Mail form. To determine whether outgoing memos should be automatically saved, use @MailSavePreference to determine the user’s preference. This returns 2 if the "Save sent mail" list has "Always prompt" selected, 1 if the "Save sent mail" list has "Always keep a copy" selected, and 0 if the "Save sent mail" list has "Don’t keep a copy" selected.

@MailSavePreference

@MailSend

There are two ways to use @MailSend:
• When used with no parameters, @MailSend mails the current document (the one being processed when the @function is evaluated) to the recipient designated in the document’s SendTo field. The document must have a SendTo field.
• When used with one or more parameters, @MailSend composes a new mail memo based on the information you supply in the arguments list, and sends it to the recipients listed in the sendTo, copyTo, and blindcopyTo arguments.

Syntax
@MailSend @MailSend( sendTo ; copyTo ; blindCopyTo ; subject ; remark ; bodyFields ; { flags } )
Parameters

sendTo

Text or text list. The primary recipient(s) of the mail memo.

copyTo

Text or text list. Optional. The copy recipient(s) of the mail memo.

blindCopyTo

Text or text list. Optional. The blind copy recipient(s) of the mail memo.

subject

Text. Optional. The text you want displayed in the Subject field. This is equivalent to the Subject field on a mail memo; the message is displayed in the Subject column in the views in the recipients’ mail databases.

remark

Text. Optional. Any text you want at the beginning of the body field of the memo.

bodyFields

Text. The names of one or more fields from the current document that you want included in the mail memo. The fields must be of type text or text list, and are appended to the memo in the order in which you list them. (You can store @Text of a numeric field in a variable and use the variable name as a field name.) Enclose each field name in quotation marks. If you want to list multiple fields, use the list format: "description"; "issues"; "resolution." If you store the name of the field in a variable, omit the quotation marks here.

When you use the [IncludeDocLink] flag (described below) to include a link to the current document, you should set the bodyFields parameter to null (""). If Notes/Domino cannot locate a field by name, it uses the string literal instead.

[ flags ]

Keyword. One or more flags indicating the priority and security of the memo. If you specify multiple flags, format them as a list, as in [SIGN]; [PRIORITYHIGH]; [RETURNRECEIPT]. Enclose each flag in square brackets, as shown.

The available flags are:

[SIGN]

Electronically sign the memo when mailing it, using information from the user’s ID. Signing does not occur unless you include this flag. This flag cannot be used in Web applications.

[ENCRYPT]

Encrypt the document using the recipient’s public key, so that only the recipient whose private key matches can read the document. Encryption does not occur unless you include this flag. This flag cannot be used in Web applications.
[PRIORITYHIGH]

Immediately routes the message to the next-hop server, as defined by the combination of Mail Connection records and server records. If a phone call has to be made in order to route the message, then the call is placed immediately, regardless of the schedule set in the Remote Connection record. If you omit this flag, the priority defaults to Normal.

[PRIORITYNORMAL]

Routes the message to the next-hop server based on the schedule defined in the Mail Connect records. If the recipient’s mail file resides on a server on the same Domino network, then delivery occurs immediately. If you omit this flag, the priority defaults to Normal.

[PRIORITYLOW]

Routes the message overnight if the recipient’s mail file does not reside on a server on the same Notes/Domino network. If the recipient’s mail file does reside on a server on the same Notes/Domino network, then delivery occurs immediately. Low Priority mail can also be controlled by a Notes/Domino environment variable called MailLowPriorityTime=x, where x is equal to a number from 0 to 23. When placed in the server notes.ini file, this variable tells the server when to route Low Priority mail. If you omit this flag, the priority defaults to Normal.

[RETURNRECEIPT]

Notify the sender when each recipient reads the message. No receipt is returned unless you include this flag.

[DELIVERYREPORTCONFIRMED]

Notify the sender whether delivery of the memo was successful or not. By default, the Basic delivery report is used, which notifies the sender only when a delivery failure occurs.

[INCLUDEDOCLINK]

Include a link pointing to the document that was open or selected when @MailSend was used. You must include this flag if you want that document linked to the mail memo. A new document must be saved.

Usage

Use @MailSend in agents, form actions, form events, view actions, view events, and toolbar buttons. @MailSend is especially useful with scheduled agents as a means of sending mail at a predetermined interval; for example, to send reminders about a departmental meeting. One view from the database must be selected as the Default when database is first opened for the scheduled agent to work correctly. This function does not work in column, selection, hide-when, or window title formulas.

If the user’s notes.ini file includes the statement

NoExternalApps=1

then any formula involving @MailSend is disabled. The user doesn’t see an error message; the formula fails to execute.

Sending rich text fields

You can specify a rich text field as one of the bodyfields in an agent formula only.
Mail-related fields in a document

When you use \MailSend with no parameters, the current document may contain one or more mail-related fields; if it does, those fields are used when routing the document.

- If the document contains the CopyTo or BlindCopyTo fields, it is routed to those recipients at the same time.
- If the document contains the DeliveryPriority, DeliveryReport, or ReturnReceipt fields, they are used to control the delivery priority, generation of a delivery report, and generation of a return receipt, just as they are used in the Actions - Send Document command. If the document doesn’t contain these fields, they default to normal priority, no delivery report, and no return receipt, respectively.

Language cross-reference

Send method of LotusScript NotesDocument class

Send method of LotusScript NotesUIDocument class

Send method of Java Document class

Examples: \MailSend

1. This formula sends a memo to David Lee with a blind copy to Joseph Smith in Support. The memo is titled “Status Report,” and its body contains the message “Sorry it’s late!” plus the contents of the STATUS and PLANS fields from the current document. The document is mailed with the following options: it is signed, delivery confirmation is requested, and a return receipt will be sent when each recipient reads the memo. The recipients are listed using distinguished naming syntax (available to Release 3 users only). The copyTo information was omitted, and was replaced with the null string because additional arguments follow.

\MailSend("David Lee/";";"Joseph Smith/Support/";"Status Report/"; "Sorry it's late!"; "STATUS:";"PLANS/"; [SIGN : [DELIVERYREPORTCONFIRMED] : [RETURNRECEIPT]]

2. This formula sends a memo to Mary Tsen and to Joseph Smith in Support. The subject uses the text stored in the current document’s TOPIC field, and the body of the memo draws from the COMMENTS field. The copyTo, blindCopyTo, and remark arguments were omitted, and were replaced with null strings because additional arguments still followed. The flags were omitted, but because no arguments followed their position, the null string was not needed.

\MailSend("Mary Tsen/";";"Joseph Smith/Support/";";";"TOPIC/";";"COMMENTS")

3. This formula sends a memo to Mary Tsen with the message "Follow this link" in the Subject field, and a link to the original document in the Body field.

\MailSend("Mary Tsen/";";"Follow this link/";";"[IncludeDocLink])

4. This agent formula sends Martha O’Connell the contents of the Comments rich text field in a memo with the subject Feedback. The agent is triggered on an Action menu selection event and its target is the selected documents.

\MailSend("Martha O'Connell/MA/Acme/";";"Feedback/";"Comments")

@MailSignPreference

Indicates whether the user has selected “Sign sent mail” in the User Preferences dialog box.

Syntax

@MailSignPreference
Return value

flag

Boolean

- Returns 1 (True) if "Sign sent mail" is selected
- Returns 0 (False) if "Sign sent mail" is not selected

Usage

@MailSignPreference is used in the Mail template to determine whether to attach an electronic signature to outgoing memos. This function is not available in column formulas, selection formulas, or selective replication formulas.

You cannot use this function in Web applications.

Language cross-reference

SignOnSend property of LotusScript NotesDocument class

IsSignOnSend property of Java Document class

Examples: @MailSignPreference

You design your own Mail form. To determine whether outgoing memos should be electronically signed, use @MailSignPreference to determine the user’s preferences. This returns 1 if the "Sign sent mail" check box is selected in the User Preferences dialog box, and 0 if the "Sign sent mail" check box is not selected.

@MailSignPreference

@Matches

Tests a string for a pattern string. Because the pattern string can contain a number of "wildcard" characters and logical symbols, you can test for complex character patterns.

Syntax

@Matches( string ; pattern )

Parameters

string

Text. The string you want to scan in quotes. You can also enter the field name of a field that contains the string you want to scan; do not surround the field name in quotes.

pattern

Text. The pattern you want to scan for in strings surrounded by quotation marks. May contain wildcard characters and symbols (see table below). The following symbols require a preceding backslash unless the pattern is enclosed in braces {} as a set: ?, *, &, !, |, \, +. The symbols require two preceding backslashes instead of one if the pattern is specified as a literal. This is because the backslash is an escape character in string literals, so "\?" passes "?" to the matching engine, where it is treated as a wildcard, while "\\?" passes "\?" to the matching engine, where it is treated as a question mark character.

Return value

flag

Boolean
• Returns 1 (True) if the string contains the pattern
• Returns 0 (False) if the string does not contain the pattern

The wildcard characters and symbols are as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Where C is any character. Matches any single, non–special character C</td>
</tr>
<tr>
<td>?</td>
<td>Matches any single character</td>
</tr>
<tr>
<td>*</td>
<td>Matches any string (any number of characters)</td>
</tr>
<tr>
<td>[ABC]</td>
<td>Matches any character in set ABC</td>
</tr>
<tr>
<td>+C</td>
<td>Matches any number of occurrences of C</td>
</tr>
<tr>
<td>!</td>
<td>Complements logical meaning of the pattern (logical NOT)</td>
</tr>
<tr>
<td></td>
<td>Performs logical OR of two patterns</td>
</tr>
<tr>
<td>&amp;</td>
<td>Performs logical AND of two patterns</td>
</tr>
</tbody>
</table>

Note: When specifying sets, be sure to enclose them in { } (curly braces). For example, the set A...F is represented as {A–F}.

Examples of pattern matching:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>&quot;ABC&quot;</td>
</tr>
<tr>
<td>[ABC]</td>
<td>[ABC]</td>
</tr>
<tr>
<td>A?C</td>
<td>Any three–character string that starts with &quot;A&quot; and ends with &quot;C&quot;</td>
</tr>
<tr>
<td>???</td>
<td>Any three–character string</td>
</tr>
<tr>
<td>+?</td>
<td>Any string, including the null string</td>
</tr>
<tr>
<td>+?[A–Z]</td>
<td>Any string that ends in a capital letter</td>
</tr>
<tr>
<td>+/-[A–Z]</td>
<td>Any string that does not contain a capital letter</td>
</tr>
</tbody>
</table>

Language cross-reference
Like operator of LotusScript language

Examples: @Matches

1. This example returns 0.
   @Matches("A big test";"a?test")

2. This example returns 1.
   @Matches("A big test";"a????test")

3. This example converts the contents of the State field to lowercase, and returns 1 for any value in the field that contains "mont," for example Vermont or Montana.
   @Matches(@Lowercase(State);"*mont")

4. This example is the default value formula for a field named SalesNumber. The formula returns the number 224 if the content of the Division field is either Central or Midwest. If the content of Division is anything else, the formula returns the number 124.
   @If(@Matches(Division;"Central | Midwest");224;124)
5. This code, when added as the validation formula for a number field called input, displays the error message, "Value cannot be a letter" if the user enters any lowercase or uppercase letter between A and Z.

@If(@Matches(@Text(input);"+[!A-z]");@Success;@Failure("Value cannot be a letter"))

**Note:** The validation error message is also triggered if the user enters a backslash, underscore, or square brackets because specifying A-z specifies all ASCII characters between the uppercase A and lowercase z. The backslash, underscore, and square brackets are included in this set of characters.

6. This code, when added as the validation formula for the US_State editable text field, displays the error message, "Entry must be a valid two-letter state abbreviation" if the user enters anything besides two upper-case letters.

@If(@Matches(US_State;"{A-Z}{A-Z}");@Success;@Failure("Entry must be a valid two-letter state abbreviation"))

---

**@Max**

Returns the largest number in a single list, or the larger of two numbers or number lists.

**Note:** The single-parameter form of this @function is new with Release 6.

**Syntax**

@Max( number1 )

@Max( number1 ; number2 )

**Parameters**

*number1*

Number or number list.

*number2*

Number or number list.

**Return value**

*maxNumber*

(Single parameter) Number. The largest number in *number1*.

(Two parameters) Number or number list. Either *number1* or *number2*, whichever is larger. If the parameters are number lists, @Max returns a list that is the result of pair-wise computation on the list values.

**Usage**

When using this function with a number list constant, remember that the list concatenation operator takes precedence over other operators. Enclose negative numbers in parentheses.

For more information, see "List concatenation operator" in the "Formula Language Rules" chapter.

**Examples: @Max**

1. This example returns 3.

   @Max(1;3)

2. This example returns 99;6;7;8.
3. This example returns -2; 45; 54.
\@Max((-2.6):45:(-25);(-2):(-50):54)

4. This formula finds the larger of the values in the fields named Commission and Salary, and compares the value to 50,000; if it is larger than 50,000, the Bonus field is changed to 0; if it is smaller than 50,000, Bonus becomes 10% of the value in the Salary field.
FIELD Bonus:=@If(@Max(Commission;Salary)>50000; 0; (0.10 * Salary));

5. This example returns 99.
\@Max(99 : 2 : 3)

6. This formula compares the corresponding sales figures in the jian_yrtotal and julie_yrtotal fields (which contain number lists) and returns a number list containing the larger of the two figures per element. If there are seven elements in the jian_yrtotal field and five in the julie_yrtotal field, this formula only returns the five largest numbers. Unlike the default behavior, which is to first repeat the fifth element until the list lengths are equal and then compare all seven corresponding elements in the lists, it does not repeat the fifth number in the julie_yrtotal field twice before performing the comparison.
tjian := @Count(jian_yrtotal);
tjulie := @Count(julie_yrtotal);
dif := (tjian - tjulie);
result := @If(@Sign(dif) = -1; @Subset(julie_yrtotal;(tjulie - @Abs(dif)));
@Subset(jian_yrtotal;(tjian - dif)));
result2 := (@If(@Sign(dif) = -1;
@Max(jian_yrtotal;result);@Max(result;julie_yrtotal));"
@If(@IsError(result2);"One of your list fields is empty";result2)

@Member
Given a value, finds its position in a text list.

Syntax
@Member(value; stringlist)

Parameters
value
Text. The value you want to find in stringlist.

stringlist
Text list

Return value
position
Number
• Returns 0 if the value is not contained in stringlist
• Returns 1 to n if the value is contained in the stringlist, where 1 to n is the position of the value in the stringlist

Examples: @Member
1. This example returns 0.
@Member("Sales";"Finance";"Service";"Legal")
2. This example returns 12 if the value in the ReportName field is the 12th value in a list contained in the RequiredReading field; otherwise it returns 0.

@Member(ReportName;RequiredReading)

@Middle

Returns any substring from the middle of a string. The middle is found by scanning the string from left to right, and parameters determine where the middle begins and ends.

Syntax

@Middle( string ; offset ; numberchars ) @Middle( string ; offset ; endstring ) @Middle( string ; startString ; endstring ) @Middle( string ; startString ; numberchars )

Parameters

string

Text. Any string.

offset

Number. A character position in string that indicates where you want the middle to begin, always counting from left to right. The middle begins one character after the offset.

startString

Text. A substring of string that indicates where you want the middle to begin, always counting from left to right. The middle begins one character after the end of startString.

numberchars

Number. The number of characters that you want in the middle. If numberchars is negative, the middle starts at offset or startString and continues from right to left. If numberchars is positive, the middle starts one character past the offset or startString and continues from left to right.

endstring

Text. A substring of string that indicates the end of the middle. @Middle returns all the characters between offset and endstring, or between startString and endstring.

Return value

middle

Text. The substring from the middle of string which begins at the offset or startString you specify and ends at the endstring you specify, or after the numberchars have been reached.

Language cross-reference

Mid function of LotusScript language

Examples: @Middle

1. This example returns "hC. The offset is positioned at the "t" (the fourth character from the left), and the count starts with the first character after the offset, moving from left to right.

@Middle("North Carolina";4;3)
2. This example returns ort. The offset is positioned at the "t" (the fourth character from the left), and the count begins at the offset, moving from right to left.
   @Middle("North Carolina";4;-3)

3. This example returns Car. The offset is positioned at the first space in the string "North Carolina" and the count starts with the first character after the offset.
   @Middle("North Carolina";" ";3)

4. This example returns or. The offset is positioned at the substring “th” and the count starts with the first character after the entire offset, moving from right to left.
   @Middle("North Carolina";"th";-2)

5. This example returns " is the " with spaces before and after "is the." The return string is everything from the fifth character through the character before "text."
   @Middle("This is the text"; 4; "text")

6. This example returns " the " with a space before and after "the." The return string is everything after " is" and before "text." The startString " is" begins with a space; this prevents @Middle from returning a string that starts at the "is" in the word "This."
   @Middle("This is the text"; " is"; "text")

@MiddleBack

Returns any substring from the middle of a string. The middle is found by scanning the string from right to left, and parameters determine where the middle begins and ends.

Syntax

@MiddleBack( string ; offset ; numberchars ) @MiddleBack( string ; offset ; endstring ) @MiddleBack( string ; startString ; endstring ) @MiddleBack( string ; startString ; numberchars )

Parameters

string

Text. Any string.

offset

Number. A character position in string that indicates where you want the middle to begin, always counting from right to left. The middle begins one character after the offset. Always add 1 to this number; the end of the string is marked by one non-visible character and must be counted in the offset.

startString

Text. A substring of string that indicates where you want the middle to begin, always counting from right to left. The middle begins one character after the end of startString.

numberchars

Number. The number of characters that you want in the middle. If numberchars is negative, the middle starts at offset or startString and continues from right to left. If numberchars is positive, the middle starts one character past the offset or startString and continues from left to right.

endstring

Text. A substring of string that indicates the end of the middle. @MiddleBack returns all the characters between offset and endstring, or between startString and endstring.
Return value

middle

Text. The substring from the middle of string, which begins at the offset or startString you specify and ends at the endString you specify, or after the numberChars have been reached.

Examples: @MiddleBack

1. This example returns Alt if the content of the Author field is Timothy Altman.
   @MiddleBack(Author;" ";;3)
2. This example returns an empty string if the content of the Author field is any string with no spaces, for example "Smith."
   @MiddleBack(Author;" ";;3)
3. This example returns "from right to left" with a space before "from."
   @MiddleBack("Middleback searches the string from right to left"; "ing";;25)
4. This example returns "searches the string ."
   @MiddleBack("@MiddleBack searches the string from right to left"; "from"; -20)
5. This example returns " is the " with spaces before and after "is the." The return string is everything from the fifth to the last character through the character after "This."
   @MiddleBack("This is the text"; 5; "This")
6. This example returns " the " with spaces before and after "the." The return string is everything before "text" and after "is."
   @MiddleBack("This is the text"; "text"; "is")

@Min

Returns the smallest number in a single list, or the smaller of two numbers or number lists.

Note: The single-parameter form of this @function is new with Release 6.

Syntax

@Min( number1 )

@Min( number1 ; number2 )

Parameters

number1

Number or number list.

number2

Number or number list.

Return value

minNumber

(Single parameter) Number. The smallest number in number1.

(Two parameters) Number or number list. Either number1 or number2, whichever is smaller. If the parameters are number lists, @Max returns a list that is the result of pair–wise computation on the list values.
Usage
When using this function with a number list constant, remember that the list concatenation operator takes precedence over other operators. Enclose negative numbers in parentheses.

For more information, see "List concatenation operator " in the “Formula Language Rules” chapter.

Examples: @Min
1. This example returns 35.
   @Min(35;100)
2. This example returns 5;2;3;3.
   @Min(99:2:3:5:6:7:8)
3. This example returns the contents of the field containing the smallest value. If Precinct1 contains 150,000 and Precinct2 contains 100,000, then this formula returns 100,000.
   @Min(Precinct1;Precinct2)
4. This example returns 85,000 if 100,000 is the smallest number contained in either of the fields AreaAPopulation or AreaBPopulation, and the field DistrictPopulation contains the value 15,000.
   @Min(AreaAPopulation;AreaBPopulation) – DistrictPopulation
5. This example returns -3.5;35;54.
   @Min((-3.5):(-35):100;(-2):45:54)
6. This example returns 2.
   @Min(99 : 2 : 3)

@Minute
Extracts the number of minutes from the specified time–date.

Syntax
@Minute( time–date )

Parameters
time–date

Time-date.

Return value
minutes

Number. The number of minutes in the minute part of the time. Returns -1 if the time-date provided contains only a date and not a time value.

Language cross-reference
Minute function of LotusScript language

Examples: @Minute
1. This example returns 30.
   @Minute([9:30])
2. This example returns 56 if the Time field contains 8:56:34 P.M.
   @Minute(Time)
3. This example returns 59 if the Date field contains: 7/30/95 9:59:59.
4. This example returns 00 if the current document’s created date was 9/29/95 3:00:12 A.M.
   @Minute(@Created)

@Modified

Returns a time–date value indicating when the document was last edited and saved.

Syntax
@Modified

Return value

lastModified

Time-date. The date when the current document was last modified.

Usage

@Modified works correctly in column formulas and computed-for-display formulas.

When used in computed fields, @Modified returns a value representing the next-to-last time the
document was saved. For example, if you modified and saved a document on the mornings of May 5th
and 6th, then accessed the document in the afternoon on May 6th, the @Modified computed field would
return the May 5th modification date, since the 5th was the next-to-last time the document was saved.

This function does not work in navigators, mail agent, paste agent, hide-when, section editor, or form
formulas.

@Modified is similar to @Accessed, which indicates the last time a document was accessed for reading or
writing.

Language cross-reference

LastModified property of LotusScript NotesDocument class

LastModified property of Java Document class

Examples: @Modified

1. This example returns 9/30/95 11:00:00 AM if the document was last saved on September 30, 1995 at
   11:00 A.M.
   @Modified

2. This example returns a string made up of the contents of the Topic field, then a space, then the string
   Last Edited: and then the time–date value of the last time the document was saved, converted to text.
   Topic + " " + "Last Edited: " + @Text(@Modified)

@Modulo

Returns the remainder of a division operation.

Syntax

@Modulo( number1 ; number2 )
Parameters

number1

Number or number list.

number2

Number or number list. If this is equal to 0, @Modulo returns @ERROR.

Return value

remainder

Number or number list. The remainder of number1 divided by number2. If the parameters are number lists, @Modulo returns a list that is the result of pair-wise computation on the list values. The sign of the result is always the same as the sign of the number1.

Usage

A common use of @Modulo is to determine whether a number is odd or even; if the result of @Modulo(number;2) is 1, the number is odd; if the result is 0, the number is even.

When using this function with a number list, the list concatenation operator takes precedence over any other operators; negative numbers must be enclosed in parentheses.

For more information, see "List concatenation operator" in the "Formula Language Rules" chapter.

Language cross-reference

Mod operator of LotusScript language

Examples: @Modulo

1. This example returns 1.
   
   @Modulo(4;3)

2. This example returns 0.
   
   @Modulo(4;2)

3. This example returns -2.
   
   @Modulo((-14);3)

4. This example returns -1;2;3;-3.
   
   @Modulo((-4);6:8:(-9);3:4:5:6)

@Month

Extracts the number of the month from the specified time-date.

Syntax

@Month( time-date )

Parameters

time-date

Time-date.
Return value

*month*

Number. The number of the month. Returns -1 if the time-date provided contains only a time value and not a date.

**Language cross-reference**

Month function of LotusScript language

**Examples: @Month**

1. This example returns 1.
   
   @Month([1/15/88])

2. This example returns 12 if it is December.
   
   @Month(@Now)

3. This example returns 2 if it is any date in December other than the 30th or the 31st. If it is December 30th or 31st, returns 3.
   
   @Month(@Adjust(@Now;2;2;2;2;2;2))

4. This formula returns a formatted date string based on the contents of the dueDate field. For example, if dueDate contains "06/26/95" the formula returns June 26, 1995. If dueDate contains "01/24/96 3:40:43 P.M.," the formula returns January 24, 1996.

   ```
   space:=" ";
   comma:=",";
   month:=@Select(@Month(dueDate);"January";"February";"March";
   "April";"May";"June";"July";"August";"September";"October";
   "November";"December");
   day:=@Text(@Day(dueDate));
   year:=@Text(@Year(dueDate));
   month + space + day + comma + space + year
   ```

**@Name**

Allows you to manipulate hierarchical names. You can abbreviate the canonical format of a name, expand an abbreviated name to its canonical format, identify particular components within the name, and reverse the order of the components so that you can categorize a view by hierarchical names.

Enables you to convert a name between the Domino and LDAP formats.

**Note:** LDAP conversion is new with Release 6.

**Syntax**

```
@Name( [ action ]; name )
```

**Parameters**

[ action ]

Keyword. Indicates what you want done to the name -- whether you want to expand it, abbreviate it, convert it, and so on (see list of possible actions below).

With @Name, you can perform the following actions:

[A]

Returns the ADMD component (administration management domain name) of a hierarchical name.
[ABBREVIATE]
Abbreviates a hierarchical name, removing the component labels. This saves space in the display, and looks friendlier.

[ADDRESS821]
Note: This keyword is new with R5.
Returns an Internet address in the format based on RFC 821 Address Format Syntax regardless of whether the original address was in RFC 821 or RFC 822 form. Case must be exact.

[C]
Returns the country/region component of a hierarchical name.

[CANONICALIZE]
Expands an abbreviated name, adding in whatever components are missing, as well as their labels. Missing components are taken from the current user ID, not from the Domino Directory.

[CN]
• Returns the common name component of a Domino name.
• Returns the local part of an Internet address in the format based on RFC 821 Address Format Syntax.
• Returns the phrase part of an Internet address in the format based on RFC 822 Address Format Syntax.

[G]
Returns the given name component (the first name) of a hierarchical name.

[HIERARCHYONLY]
Note: This keyword is new with R5.
Strips the CN component of a hierarchical name and returns the remaining components.

[I]
Returns the initials component of a hierarchical name.

[LP]
Note: This keyword is new with R5.
Returns the LocalPart of a standard Internet address based on RFC 822 Address Format Syntax.

[O]
Returns the organization component of the hierarchical name.

[OU n ]
Returns the specified organizational unit component of a hierarchical name; n can be from 1 to 4, as in OU1. In the canonical form of the name, the OU components are not numbered; however, they are
counted from right to left so that the first occurrence of the OU label is treated as OU1, the second occurrence is treated as OU2, and so on. Notes/Domino does not accept [OU] as a keyword.

[P]

Returns the PRMD component (private management domain name) of a hierarchical name.

[PHRASE]

Note: This keyword is new with R5.

Returns the Phrase part of a standard Internet address based on RFC 822 Address Format Syntax.

[Q]

Returns the generation component (such as "Jr") of a hierarchical name.

[S]

Returns the surname component (the last name) of a hierarchical name.

[TOAT]

Note: This keyword is new with Release 6.

Returns the LDAP AttributeType name when a Domino field name is provided.

[TODATATYPE]

Note: This keyword is new with Release 6.

Returns the Domino data type name when an LDAP Syntax name is provided.

[TOFIELD]

Note: This keyword is new with Release 6.

Returns the Domino field name when an LDAP AttributeType name is provided.

[TOFORM]

Note: This keyword is new with Release 6.

Returns the Domino form name when an LDAP ObjectClass name is provided.

[TOKEYWORD]

Reverses the order in which the naming components are displayed, and replaces slashes with backslashes: Country\Organization\Organization Unit... This is useful when you want to categorize a view by the components of a user’s hierarchical name (backslashes represent subcategories in views). The [TOKEYWORD] option does not return the Common Name portion of the user name.

[TOOC]

Note: This keyword is new with Release 6.
Returns the LDAP ObjectClass name when a Domino form or subform name is provided.

[TOSYNTAX]

Note: This keyword is new with Release 6.

Returns the LDAP Syntax name when a Domino data type name is provided.

name

Text or names. A user or server name, entered in any form (Lotus Notes/Domino determines the full hierarchical name and then returns the requested components) or an LDAP AttributeType, ObjectClass, or Syntax name or a Domino form, subform, field, or data type name to be converted from LDAP to Domino format or vice-versa.

Usage

@Name is particularly useful for abbreviating hierarchical names in a view.

A hierarchical name is qualified with a series of components identifying the full name, organizational unit, organization, and country or region. Using hierarchical names guarantees that each user and server has a unique name.

As the database designer, you are responsible for controlling how user names are entered and displayed within Notes applications. For simplicity, you should allow users to enter names in abbreviated form; then you can use @Name to expand the name to its canonical format. You should also display names in abbreviated form, using @Name to convert the stored canonical format of the name to its abbreviated form.

When you use a Names, Readers, or Authors field, Lotus Notes/Domino automatically converts hierarchical names to an appropriate format for display and storage. If the user enters an abbreviated name, Lotus Notes/Domino expands it to canonical format when storing it; the name is always displayed on a form in abbreviated format.

When you display the contents of a hierarchical name field in a view there is no automatic conversion; the entire canonical format of the name is displayed. You may want to convert the name to its abbreviated form with @Name.

If you are using @Name to parse an Internet address, the address must conform to the format based on the standard RFC 821 or RFC 822 Address Format Syntax.

Note: If you attempt to use the parameters A, G, I, P, Q, or S in Lotus Notes/Domino with existing user IDs, it may appear as though the parameters do not work. These parameters were added to take advantage of the addressing used for external mail and gateway products. When a mail message is received within Lotus Notes/Domino from an external mail source, the naming convention can include additional components. The @Name function can be used to manipulate the hierarchical name, including these additional components. Domino IDs and names do not use these additional components, therefore, it is not possible to use these six parameters with a standard Domino ID and name.

Below is an example of a full hierarchical name that takes advantage of every parameter.

G=Joe/I=JS/S=Smith/Q=Jr/CN=Joseph
Smith/OU=Assembly/OU=Engineering/O=Acme/P=PrivAdmin/ A=PubAdmin/C=US
Language cross-reference
LotusScript NotesName class

Examples: @Name

1. This example returns Mary Tsen/Illustration/ Documentation/Development/ R&D/WorkSavers/US.
   @Name([ABBREVIATE];AUTHOR)
   If a user is looking at a document where the AUTHOR field contains the hierarchical form of Mary Tsen's name.

2. This example returns Mary Tsen.
   @Name([CANONICALIZE];"Mary Tsen")
   Since there is no slash following the name, it is a nonhierarchical name and has no additional components.

3. This example returns CN=MaryTsen/ OU=Illustration/OU=Documentation/OU=Development/OU=R&D/O=Acme/C=US if that is the current user ID. The hierarchy of the current user ID is appended to the name; no lookup occurs in the Domino Directory.
   @Name([CANONICALIZE];"Mary Tsen/")

4. This example returns Mary Tsen in an informational dialog box format, if the AUTHOR field in the document contains: CN=Mary Tsen/OU=Illustration/O=Acme.
   @Prompt([Ok]; "Common Name"; @Name([CN]; AUTHOR))

5. This example returns Development.
   @Name([OU2];AUTHOR)

6. This example returns US\Acme\R&D\Development\Documentation\Illustration. The slashes are now backslashes, which allow the naming components to be used as subcategories in a view. The common name component is not returned.
   @Name([TOKEYWORD];AUTHOR)

7. This example returns SStreitfeld if the User_Name field contains this Internet address in RFC 822 format "Streitfeld, Sara (Miami)" <SStreitfeld@gazette.com>.
   @Name([LP]; User_Name)

8. This example returns "Streitfeld, Sara (Miami)" if the User_Name field contains this Internet address in RFC 822 format "Streitfeld, Sara (Miami)" <SStreitfeld@gazette.com>.
   @Name([Phrase];User_Name)

9. This example returns SStreitfeld@gazette.com if the User_Name field contains this Internet address in RFC 822 format "Streitfeld, Sara (Miami)" <SStreitfeld@gazette.com>.
   @Name([ADDRESS821];User_Name)

10. This example returns Cam/Lotus If the User_Name field contains John Doe/Cam/Lotus.
    @Name([HIERARCHYONLY];User_Name)

11. This example returns "secretary," the LDAP AttributeType name for the Domino term, "assistant."
    @Name([TOAT];"assistant")

12. This example returns "Internet Address," the Domino term equivalent to the LDAP AttributeType name "mail."
    @Name([TOFIELD];"mail")

13. This example returns "Number," the Domino term equivalent to the LDAP data type, "Integer."
    @Name([TODATATYPE];"Integer")

14. This example returns "Directory String," the syntax used in the LDAP directory for the Domino data type "Text."
    @Name([TOSYNTAX];"Text")
@NameLookup

Searches for each specified user name across all Domino Directories and returns a list of single text values for each specified user name.

Note: This @function is new with Release 5.

Syntax
@NameLookup( [lookupType ]; username; itemtoreturn )

Parameters
[lookupType]
Keyword. Specifies the type of lookup to perform. Supply one or more of the following keywords:

[NoUpdate]
Default. Returns a list of user names. Corresponds to NAME_LOOKUP_NOUPDATE flag for Notes API. You can specify this keyword along with the other keywords excluding[ForceUpdate].

[ForceUpdate]
Forces the name space (view) to be updated. Corresponds to NAME_LOOKUP_UPDATE flag for Notes API. You can specify this keyword along with the other keywords excluding[NoUpdate].

The following keywords can be used along with the [NoUpdate] or [ForceUpdate].

[NoSearching]
Searches only the first Domino Directory containing the "($Users)" view, which is the local Names.nsf database, and returns a list of single text values for each specified user name. This keyword specifies to not retrieve values from the mail server’s directory. An empty string is returned for no match found. Corresponds to NAME_LOOKUP_NOSEARCHING flag for Notes API.

[Exhaustive]
Searches all Domino Directories listed in the Notes Directory Assistance and returns all information in a text list. This keyword returns values from the local Names.nsf database as well as the mail server’s directory. If the mail server is unavailable, it retrieves values from the current name server. If you are using LDAP, it also retrieves values from the LDAP directory. The user’s value is omitted if there is no match found.

[TrustedOnly]
Searches only those Domino Directories that contain trust information and returns a list of single text values for each specified user name. An empty string is returned for no match found. Corresponds to NAME_LOOKUP_TRUSTED_NAMESPACES flag for Notes API.

username
Text or text list. Specify primary or alternate Notes/Domino user names to retrieve their information from the Domino Directory.

itemtoreturn
Text. Item or field name from the Domino Directory Contact record that you would like to retrieve information from.

**Return value**

`valuelist`

Text list. When other flags besides `[Exhaustive]` have been specified, `@NameLookup` returns a list of single values for each specified user. An empty string is returned for no match found. When `[Exhaustive]` has been specified, `@NameLookup` returns a list of all information matched for the specified user. No value is returned for unmatched users. To display the return values in a dialog box using `@Prompt`, enclose this function in an `@Text` function.

**Usage**

`@NameLookup` cannot be used in form selection and view column formulas.

All the users from secondary directories, including the LDAP directory, need to be authenticated first, and then authorized to access a Notes/Domino database administered by the Domino server. The Directory Assistance derived from the Master Domino Directory uses trusted name rules to authenticate users. Once a user name is authenticated, it is added to the list of trusted names. This user name is then compared to the ACL for authorization.

For more information on searching the LDAP directory, see “Setting up Notes to search an Internet directory for addresses” in the Lotus Notes Help.

**Examples: `@NameLookup`**

You have three Domino Directories on your local environment, namely, `Names_A.nsf`, `Names_B.nsf`, and `Names_C.nsf`. Each Directory has the following entries:

<table>
<thead>
<tr>
<th></th>
<th><code>Names_A.nsf</code></th>
<th><code>Names_B.nsf</code></th>
<th><code>Names_C.nsf</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>View:</td>
<td>($Users)</td>
<td>Does not exist</td>
<td>Exists</td>
</tr>
<tr>
<td>User:</td>
<td>Katsushi</td>
<td>User: Katsushi</td>
<td>User: Katsushi</td>
</tr>
<tr>
<td></td>
<td>Item: Katsushi_A</td>
<td>User: Katsushi_B</td>
<td>Item: Katsushi_C</td>
</tr>
<tr>
<td>User:</td>
<td>Jones</td>
<td>User: Jones</td>
<td>User: Jones</td>
</tr>
<tr>
<td></td>
<td>Item: Jones_A</td>
<td>Item: Jones_B1</td>
<td>Item: Jones_B2</td>
</tr>
<tr>
<td>User:</td>
<td>Smith</td>
<td>User: Smith</td>
<td>Does not exist</td>
</tr>
<tr>
<td></td>
<td>Item: Smith_A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User:</td>
<td>Yoshito</td>
<td>Does not exist</td>
<td>Does not exist</td>
</tr>
</tbody>
</table>

1. The following formulas return "Katsushi_B" : "Jones_B1" : "Smith_C" : ""
   
   `@NameLookup` ( [NoUpdate]; "Katsushi";"Jones";"Smith";
   "Yoshito"; "Item")
   
   `@NameLookup` ( [ForceUpdate]; "Katsushi";"Jones";"Smith";
   "Yoshito"; "Item")

2. The following formula returns "Katsushi_B" : "Jones_B1" : "" : ""
   
   `@NameLookup` ( [NoSearching]; "Katsushi";"Jones";"Smith";
   "Yoshito"; "Item")

3. The following formula returns "Katsushi_B" : "Katsushi_C" : "Jones_B1" : "Jones_B2" : "Smith_C"
   
   `@NameLookup` ( [Exhaustive]; "Katsushi";"Jones";"Smith";
   "Yoshito"; "Item")
4. If the current user is a software engineer, the following code, when added as the default value for a field, displays SOFTWARE ENGINEER.

```
@NameLookup(Exhaustive; UserName; "JobTitle")
```

@Narrow

Converts full-pitch alphanumeric characters (double byte characters -- DBCS) in the specified string to half-pitch alphanumeric characters (single byte characters -- SBCS). This function works in Japanese, Korean, Simplified Chinese, and traditional Chinese environments. In the Japanese environment, this function can convert full-pitch Katakana as well.

**Note:** This @function is new with Release 5.

**Syntax**

```
@Narrow( string )
```

**Parameters**

*string*

Text. The string that you want to convert to single byte characters.

**Return value**

*returnstring*

Text. The string converted to single byte characters.

**Usage**

This function can be used in input translation formulas to convert the contents of a field to single byte characters or in computed field formulas to save space for displaying the string.

**Language cross-reference**

StrConv function of LotusScript language

**Examples: @Narrow**

1. This input translation formula returns "Tokyo" as a half-pitch character, if the Location field contains a full-pitch character expression of "Tokyo."

   ```
   @Narrow(Location)
   ```

2. This computed field formula returns "New York" as a half-pitch character to save space for displaying the string.

   ```
   @Narrow("New York")
   ```

@NewLine

Inserts a new line (carriage return) into a text string.

**Syntax**

```
@NewLine
```
Return value  
*carriageReturn*  

Text. A carriage return.

**Usage**  
On the Web, this function does not work in selection, form, or window title formulas.

In Lotus Notes, this function does not work in selection, hide-when, column, window title, form formulas, or inside of @Prompt.

If you need to insert a carriage return inside an @Prompt formula, see [@Char](#).

**Tip:** To add multiple lines to a single column row:

1. In the View Properties box:
   - Change the Lines per row to the number of carriage returns you want to include in the row.
   - Select Shrink rows to content.
2. In the Column Properties box:
   - Choose New Line as the Multi-value separator.
   - Deselect the Show multiple values as separate entries check box.
3. In the code for the column formula, specify each string or number that you want to display on a new line as a separate value. Since you set the Multi-value separator to New Line, this inserts a carriage return between each value. For example, the following column formula vertically lists the content of the FirstName field above the content of the LastName field in the column row:

   ```
   first:= FirstName;
   last := LastName;
   @Trim(first : last)
   ```

**Language cross-reference**  
AddNewLine method of LotusScript NotesRichTextItem class  
addNewLine method of Java RichTextItem class

**Examples:**  
[@NewLine](#)  

1. This returns  
   
   Hi There  
   
   "Hi"+@NewLine+"There"

2. This returns  
   
   Foster, Steven  
   
   in the EmpName field if the string in the LastName field is Foster, and the string in the field named FirstName is Steven.  
   
   FIELD EmpName:= LastName + "," + @NewLine + FirstName;

3. This input translation formula uses @Newline to replace all occurrences of "%" with a carriage return. If the description field contains "Here we are now%Entertain us," the formula translates it to:

   ```
   Here we are now Entertain us  
   @Implode(@Explode(description; "\%"; @NewLine))
   ```

[@No](#)  

Returns the number 0.
Syntax
@No

Return value
no

Number. Zero (0).

Usage
This function is equivalent to @False.

Examples: @No
1. This example returns 0.
   @No
2. This example returns 1 if the value in the Cost field is greater than 100; otherwise returns 0.
   @If(Cost>100;@Yes;@No)

@NoteID
The ID number of the current document.

Syntax
@NoteID

Return value
NT idnumber

String. The prefix NT followed by the note ID.

Usage
This function does not work in forms or navigators.

Language cross-reference
NoteID property of LotusScript NotesDocument class

NoteID property of Java Document class

@Nothing
Use with an @Transform formula. Returns no list element (reducing the return list by one element). Not valid in any other context.

Note: This @function is new with Release 6.

Syntax
@Nothing
Return value

`nothingOrNull`

Nothing or null. Returns nothing in an @Transform formula, or null.

Usage

See `@Transform` for additional information and examples.

`@Now`

Returns the current time-date.

Syntax

```plaintext
@Now( flags ; serverNames )
```

Note: The `flags` and `serverNames` parameters are new with Release 6.

Parameters

`flags`

Keyword or keyword list. Optional.

- `[SERVERTIME]` gets the time-date from the server containing the database if `serverNames` is not specified or from `serverNames` if `serverNames` is specified.
- `[LOCALTIMEONERROR]` gets the time-date from the local computer if an error occurs getting it from a specified server.

`serverNames`

Text or text list. Optional. A server name or a list of server names. This parameter applies when `[SERVERTIME]` is specified.

Return value

`now`

Time-date or time-date list. The current time-date of the local computer, the server containing the current database, or one or more specified servers. See the “Usage” section that follows.

Usage

@Now gets the time-date of the local computer in the following cases:

- No parameters are specified.
- `[SERVERTIME]` is specified, but the database is local and `serverNames` is not specified.
- `[LOCALTIMEONERROR]` is specified, `serverNames` is specified, and an error occurs getting the time-date from a server.

@Now gets the time-date of the server containing the current database if `[SERVERTIME]` is specified and `serverNames` is not specified.

@Now gets the time-date or time-dates of one or more specified servers if `[SERVERTIME]` and `serverNames` are specified.

An error occurs if @Now cannot get the time from a server specified in `serverNames` and `[LOCALTIMEONERROR]` is not specified.
Using @Now in column or selection formulas may impact the efficiency of your application. It also causes the view refresh indicator to display constantly.

The @Now function returns the current time with one hundredths of a second precision. However, if you use @Now to specify the current time in a computed field, the hundredths of a second value is always rounded up to the next second, which can result in the current time being one second fast. You can avoid this by replacing @Now with the following formula:

\[
timelow := \text{@Now}; \\
\text{@Date(\text{@Year(timelow)}; \text{@Month(timelow)}; \text{@Day(timelow)}; \text{@Hour(timelow)}; \text{@Minute(timelow)}; \text{@Sec(timelow)})}
\]

**Language cross-reference**

Now function of LotusScript language

**Examples: @Now**

1. This field value formula returns **01/21/96 7:30:45 AM** at 7:30:45 A.M. on January 21, 1996.
   
   \[
   \text{@Now}
   \]

2. This agent displays the times on the two servers named Snapper and Tornado.
   
   \[
   \text{@Prompt([Ok]; } \\
   \text{"Server time"; } \\
   \text{@Implode("Snapper" : "Tornado" + " ") + } \\
   \text{@Text(\text{@Now([ServerTime] : [LocalTimeOnError]}; } \\
   \text{"Snapper" : "Tornado"); @Char(13))}
   \]

**@OptimizeMailAddress**

Given a mail address, returns it with all unnecessary domains removed.

**Syntax**

\[
\text{@OptimizeMailAddress( address )}
\]

**Parameters**

\[
\text{address}
\]

String. The mail address to optimize.

**Return value**

\[
\text{optimizedAddress}
\]

String. The optimized address.

**Usage**

All domains between two duplicate domains, including the duplicate domain, are removed.

**Examples: @OptimizeMailAddress**

1. This example returns "username @ firstdomain @ thirdomain."
   
   \[
   @OptimizeMailAddress ("username @firstdomain @secondomain @firstdomain @thirdomain")
   \]

2. This example returns "username @ firstdomain @ secondomain."
   
   \[
   @OptimizeMailAddress ("username @firstdomain @firstdomain @secondomain")
   \]
@OrgDir

In a Service Provider (xSP) environment, returns the name of the subdirectory for the company with which the currently authenticated user is registered. Lotus Notes/Domino retrieves this information from the organization’s certifier document.

Note: This function is new with Release 6.

Syntax
@OrgDir

Return value
subdirectory name

String. The name of the subdirectory containing the data directory for the company with which the current user is registered.

Usage
If the currently authenticated user is not registered in a hosted organization, is authenticated as an anonymous user, or if the function is invoked in a non-xSP environment, Lotus Notes/Domino returns an empty string (""").

Examples: @OrgDir
If the full path name of the data directory subdirectory for a hosted organization called Acme is C:\Lotus\Notes\Data\Acme, the following code opens a database on the same server that has the same name as the current database, but that resides in the Acme organization’s subdirectory. @OrgDir returns ”Acme” in the formula below.
@Command([[FileOpenDatabase];@ServerName + ":" + @OrgDir + "\" + @DbName[2])

@Password

Encodes a string.

Syntax
@Password( string )

Parameters
string

Text or text list. The string that you want encoded.

Return value
encodedString

Text or text list. The encoded string.

Usage
@Password is especially useful in an input translation formula to protect a user’s password from being seen by others.

Note: There is no way to decode the original string once it has been encoded by @Password.
Note: Strings that begin with an open parenthesis "(" are not encoded.

Examples: @Password
1. This example returns (0449960361D30391DDA7747D537C32F8).
   @Password("chocolate")
2. This example returns (EFFF7C4218F3CBD6D7B509CD2E021DD8).
   @Password("vanilla")
3. This example returns (0449960361D30391DDA7747D537C32F8).
   @Password("chocolate":"vanilla")

@PasswordQuality
Evaluates the return value of a Password data type field with a number.

Note: This function is new with Release 5.0.1.

Syntax
@PasswordQuality( field_name )

Parameters
field_name

The name of a field with a password data type.

Return value
passwordQuality

Number. A rating indicating the level of complexity of a password. A high number indicates a complex password that is difficult to decipher.

Usage
This function is supported on the Web.

Language cross-reference
MinPasswordLength property of LotusScript NotesRegistration class
MinPasswordLength property of Java Registration class

@Pi
Returns the constant value pi, accurate to fifteen decimal places. The value pi is the ratio of the circumference of a circle to its diameter.

Syntax
@Pi

Return value
pi

The number 3.14159265358979.
Language cross-reference
PI constant of LotusScript language

Examples: @Pi
1. This formula returns the circumference of a circle with a radius that equals 5.
   \[ 2 \times \pi \times 5 \]
2. This formula converts an angle from degrees to radians. One degree equals \( \pi/180 \) radians. Thus an angle of 360 degrees equals 2\( \pi \) radians, 180 degrees equals \( \pi \) radians, and so on.
   \[ \left( \text{angle} \times \pi \right) / 180 \]
3. Given the latitude of a particular location, you can find a location’s distance from the equator. The numeric field latitude holds the latitude in degrees. The numeric field distance computes the distance from the equator using this formula.
   First, latitude is converted to radians. Next, it’s multiplied by 6440, the approximate radius of the earth in kilometers. This gives us the length of the arc from the equator to the given latitude.
   Lotus Notes/Domino treats an empty numeric field as a text field, so the formula uses @If to check for an empty latitude field.
   \[ @If( \text{latitude} = "" ; 0 ; ( \text{latitude} \times \pi ) / 180 ) \times 6440 \]

@PickList
Displays a modal window that contains either:
- A view you specify from which the user can select one or more documents. @PickList returns a column value from the selected document(s).
- A dialog box, displaying information from all available Domino Directories. The user can select one or more person, group, server, room, or resource names, and @PickList returns those names.

Syntax
@PickList( [CUSTOM] : [SINGLE] ; server : file ; view ; title ; prompt ; column ; categoryname )

@PickList( [NAME] : [SINGLE] )

@PickList( [ROOM] )

@PickList( [RESOURCE] )

@PickList( [FOLDERS] : [SINGLE] ; server : database )

@PickList( [FOLDERS] : [SHARED] ; server : database )

@PickList( [FOLDERS] : [PRIVATE] ; server : database )

@PickList( [FOLDERS] : [NODESKTOP] ; server : database )

Parameters
[CUSTOM]
Keyword. Indicates that you want to display a view in a dialog box.

[NAME]
Keyword. Opens dialog box for selecting one or more names.
[SINGLE]
Keyword. Optional. Limits the selection to a single document.

[ROOM]
Keyword. Opens dialog box for selecting room.

[RESOURCE]
Keyword. Opens dialog box for selecting resources.

[FOLDERS]
Keyword. Returns a multi-select, text list of all folder names both in the database and from the desktop. The following keywords can be combined with [Folders]:

[SINGLE]
Keyword. Optional. Limits selection to a single folder.

[SHARED]
Keyword. Optional. Limits selection to only shared folders.

[PRIVATE]
Keyword. Optional. Limits selection to only private folders (both in the database and on the desktop).

[SHARED];[PRIVATE]
Keyword. Optional. Includes in selection all shared and private folders.

[NODESKTOP]
Keyword. Optional. Excludes folders in the desktop from selection.

server : file
Text list. The server is the name of the server where the database is. The file is the path and file name of the database you want to open. Specify the name and location of the database using the appropriate format for the operating system.

You can use a replica ID in place of a server and file name as the parameter following the [custom] keyword only. The replica ID must be text and must include the colon between the two sets of eight hex digits. For example:

@PickList([CUSTOM]; "852564A0:006B7872"; "By Category"; "Testing replica ID"; "Test prompt"; 3)

Use "" to specify the currently open database.

view
Text. The name of the view that you want to open in the database.

title
Text. The window title for the dialog box.

*prompt*

Text. The prompt that you want to appear inside the dialog box. Only one line of text is displayed. Longer lines are truncated.

*column*

Number. A number indicating which column value you want @PickList to return. Use 1 to indicate the first column, 2 to indicate the second column, and so on. Unlike @DbColumn and @DbLookup, @PickList counts all columns, regardless of the types of formula they contain.

*categoryname*

**Note:** This parameter is new with Release 5.

Text. Optional. Displays the specified category in the view. The view should be categorized in order to use this parameter.

**Return value**

*columnValue*

Text list. The value(s) in the specified column for the document(s) that the user selected.

**Usage**

This function is useful in button, manual agent, paste agent, form action, and view action formulas. It does not work in column, selection, mail agent, scheduled agent, hide-when, window title, or form formulas.

Although @PickList([CUSTOM]) operates similarly to @DbColumn and @DbLookup, @PickList is preferable because it:

- Stores more data
- Performs the lookup faster
- Allows you to quickly locate the desired document by typing the first few characters

@PickList doesn’t offer a NoCache option like @DbColumn and @DbLookup because lookup results are never stored. Each time @PickList is executed, a new lookup is performed.

For a calendar view, @PickList displays two days starting with today, without time slots. The user can click on the date picker button to navigate to other days.

You cannot use this function in Web applications.

@PickList can return no more than 64K bytes of data. Use the following equations to determine how much of your data can be returned using @PickList.

For lookups that return text:

\[ 2 + (2 \times \text{number of entries returned}) + \text{total text size of all entries} \]

For lookups that return numbers or dates:

\[ (10 \times \text{number of entries returned}) + 6 \]
However, @PickList can access a view of any size, so there are no limits to the number of choices it can present. Only the return value is limited in size.

**Language cross-reference**
PickListStrings method of LotusScript NotesUIWorkspace class

**Examples: @PickList**

1. This formula displays the Products view of PROD.NSF in a dialog box. If the user selects a Staple remover and Stapler from the products view, the temporary variable choice gets assigned the following text list: `Staple remover; Stapler`
   ```plaintext
   choice:=@PickList( [CUSTOM] ; "" ; "Products" ; "Select a product" ; "Please select the products you want to order" ; 1 );
   ```
2. This formula achieves the same result as the one above, but uses @DbName to display the Products view of the current database.
   ```plaintext
   choice:=@PickList( [CUSTOM] ; @DbName ; "Products" ; "Select a product" ; "Please select the products you want to order" ; 1 );
   ```
3. This formula also displays the Products view of the current database, but returns the contents of the second column in the view.
   ```plaintext
   choice:=@PickList( [CUSTOM] ; @DbName ; "Products" ; "Select a product" ; "Please select the products you want to order" ; 2 );
   ```
4. This formula is the same as above but limits the selection to a single document.
   ```plaintext
   choice:=@PickList( [CUSTOM] ; [SINGLE] ; @DbName ; "Products" ; "Select a product" ; "Please select the products you want to order" ; 2 );
   ```
5. This formula opens the By Category view of the current database and displays only the items in the Leather category.
   ```plaintext
   choice:=@PickList( [CUSTOM] ; "" ; "By Category" ; "Select a product" ; "Please select the products you want to order" ; 5 ; "Leather" );
   ```
6. This formula displays the Names dialog box. The names of the people, groups, or servers that the user selects are placed in the person field on the current document.
   ```plaintext
   FIELD person:=person;
   @SetField( "person" ; @PickList( [NAME] ) )
   ```

---

**@Platform**

Returns the name of the currently running platform version of Lotus Notes/Domino.

**Syntax**

```
@Platform( [S PECIFIC ] )
```

**Parameters**

[S PECIFIC ]

Keyword. Optional. Returns more detailed information; for example, the version number in addition to the name of the platform.

**Return value**

`platform`
Text or text list. Without the parameter, returns the name of the platform. May be any of the following:

<table>
<thead>
<tr>
<th>Windows/16</th>
<th>OS/2v1</th>
<th>Macintosh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows/32</td>
<td>OS/2v2</td>
<td>UNIX</td>
</tr>
<tr>
<td>MS-DOS</td>
<td>OS/400</td>
<td>NetWare</td>
</tr>
</tbody>
</table>

When you use the [Specific] keyword, @Platform returns a text list containing the following items:

- **PrimaryOSName**
  One of the platform names listed below:

<table>
<thead>
<tr>
<th>Windows/16</th>
<th>OS/2v1</th>
<th>SUN Sparc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows/95</td>
<td>OS/2v2</td>
<td>SOLARIS x86</td>
</tr>
<tr>
<td>Windows/NT</td>
<td>IBM OS/400</td>
<td>SOLARIS Sparc</td>
</tr>
<tr>
<td>MS-DOS</td>
<td>IBM OS/390</td>
<td>SCO OpenDeskTop</td>
</tr>
<tr>
<td>NetWare</td>
<td>AIX</td>
<td>Linux</td>
</tr>
<tr>
<td>Macintosh/68K</td>
<td></td>
<td>UNIXWARE</td>
</tr>
<tr>
<td>Macintosh/PowerPC</td>
<td></td>
<td>HP UNIX</td>
</tr>
</tbody>
</table>

- **PrimaryOSVersionNumber**
  The current version number of the primary operating system. The number is specific; for example, 3.11. For the UNIX platform, @Platform([SPECIFIC]) returns only the specific platform name, not the version number.

- **SecondaryOSName**
  The name of the secondary operating system. For example, MS-DOS is the secondary operating system when Windows/16 runs on top of it. The values are the same as those for the primary operating system. Most platforms don’t have a secondary operating system.

- **SecondaryOSVersionNum**
  The current version number of the secondary operating system.

On a Windows 16-bit platform, for example, @Platform([Specific]) returns Windows/16;3.11;MS-DOS;6.22

**Usage**

When it is used in column, selection, or scheduled agent formulas, @Platform returns the current platform where the database resides. If the database resides on a server, @Platform returns the server platform; if the database resides locally, @Platform returns the workstation platform.

Your application may perform certain operations that are not available in all platform versions of Lotus Notes/Domino (such as the DDE–related functions). Rather than receive an error, you could use @Platform to determine whether or not to perform the operation.

You can use @Platform([Specific]) to distinguish between Windows 32 platforms (NT versus 95), and between UNIX or OS/2 platforms.

This function returns the server platform only. Use @ClientType to distinguish between Web and Notes/Domino users.

In Web applications, @Platform returns the platform only.
Language cross-reference
Platform property of LotusScript NotesSession class
Platform property of Java Session class

@PolicyIsFieldLocked
Indicates whether a field is locked by an administration policy and cannot be modified.

Note: This @function is new with Release 7.

Syntax
@PolicyIsFieldLocked( fieldName )

Parameters
fieldName
Text or text list. The name(s) of the field(s) being queried.

Return value
flag
Number.
• Returns 1 (True) if the field is locked. For a list, all the fields must be locked.
• Returns 0 (False) if the field, or any field in a list, is not locked.

Usage
This function is intended for use in hide-when and Input Enabled formulas.
This function does not work in view column, view selection, or view action formulas.
A policy lock is indicated by the presence in the document of a computed field whose name is or begins with "$DPLocked" and whose value is the name of the locked field.
A document may contain any number of locked fields.

Examples: @PolicyIsFieldLocked
This field hide-when formula hides the field if it is locked by an administration policy.
@PolicyIsFieldLocked(@ThisName)

@PostedCommand
Executes a Notes/Domino command. Most of the standard menu commands can be executed using @PostedCommand. In addition, a number of specialized commands are available. In a formula, any command invoked using @PostedCommand executes after the rest of the formula has been evaluated.

Syntax
@PostedCommand( [ command ] ; parameters )
**Parameters**
See @Commands for a list of available commands.

See the specific @Command topic for details on parameters available for that command.

**Usage**
This function does not work in column, selection, hide-when, section editor, window title, field, or form formulas, or in agents that run on a server. It’s intended for use in toolbar button, hotspot, and action formulas.

**Note:** If your formula will be executed in Notes Release 3, use @PostedCommand instead of @Command; Notes Release 3 cannot execute an @Command formula constructed in Release 4 or later. If your @command formulas constructed in Release 3 are compiled in Release 4 or later, Lotus Notes/Domino automatically changes each occurrence of @Command to @PostedCommand.

**@Power**
Raises a number to the power of an exponent.

**Syntax**
@Power( base ; exponent )

**Parameters**

*base*

Number. The value that you want raised to *exponent*. May be positive or negative.

*exponent*

Number. The power.

**Return value**

*result*

Number. The value of *base* raised to the power of *exponent*.

**Language cross-reference**

Exponentiation operator of LotusScript language

**Examples: @Power**

1. This example returns 8 (2 raised to the power of 3, or \(2^3\)).
   @Power(2;3)

2. This example returns -8 (-2 raised to the power of 3, or \((-2)^3\)).
   @Power(-2;3)

3. This example returns 0.125 (2 raised to the power of -3, or \(2^{-3}\)).
   @Power(2;-3)
**@Prompt**

Displays a dialog box to the user and returns a text value based on the user's actions in the dialog box. @Prompt is useful for prompting a user for information and determining a course of action based on the user's input.

### Summary of Dialog Box Styles

This table shows the different styles of dialog boxes you can display. @Prompt accepts parameters and returns a value based on the style you indicate.

<table>
<thead>
<tr>
<th>Style</th>
<th>Purpose</th>
<th>Contains</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChooseDatabase</td>
<td>Allows user to select a database</td>
<td>Controls and displays for browsing databases; Open, Select, Cancel, Browse, Help, and About buttons</td>
<td>Text list of 3 values. Server name, file name, and title of database. Returns null for server name if the database is local.</td>
</tr>
<tr>
<td>LocalBrowse</td>
<td>Allows user to select a file name from the local file system</td>
<td>Controls and displays for browsing local file system; Select, Cancel, and Network or Help buttons</td>
<td>Text. File name that user selected or entered.</td>
</tr>
<tr>
<td>Ok</td>
<td>Displays an informational message</td>
<td>Title and prompt; OK button</td>
<td>1 (True).</td>
</tr>
<tr>
<td>OkCancelCombo</td>
<td>Allows user to select one value from a drop-down list of choices</td>
<td>Title and prompt; List of choices; OK and Cancel buttons</td>
<td>Text. Value that user selected.</td>
</tr>
<tr>
<td>OkCancelEdit</td>
<td>Allows user to type in text input</td>
<td>Title and prompt; Text box for input; OK and Cancel buttons</td>
<td>Text. Value that user entered.</td>
</tr>
<tr>
<td>OkCancelEditCombo</td>
<td>Allows user to select one value from a list of choices, or type in a different value</td>
<td>Title and prompt; List of choices with text box; OK and Cancel buttons</td>
<td>Text. Value that user selected or entered.</td>
</tr>
<tr>
<td>OkCancelList</td>
<td>Allows user to select one value from a list of choices</td>
<td>Title and prompt; List of choices; OK and Cancel buttons</td>
<td>Text. Value that user selected.</td>
</tr>
<tr>
<td>OkCancelListMult</td>
<td>Allows user to select multiple values from a list of choices</td>
<td>Title and prompt; List of choices; OK and Cancel buttons</td>
<td>Text list. All values that user selected.</td>
</tr>
<tr>
<td>Password</td>
<td>Allows user to enter password without displaying it on the screen</td>
<td>Title and prompt; Text box that accepts and hides user input; OK and Cancel buttons</td>
<td>Text. Password that user entered.</td>
</tr>
<tr>
<td>YesNo</td>
<td>Allows user to make a Yes/No decision</td>
<td>Title and prompt; Yes and No buttons</td>
<td>1 (True, Yes) or 0 (False, No).</td>
</tr>
<tr>
<td>YesNoCancel</td>
<td>Allows user to make a Yes/No decision, or Cancel</td>
<td>Title and prompt; Yes, No, and Cancel buttons</td>
<td>1 (True, Yes), 0 (False, No), or -1 (Cancel).</td>
</tr>
</tbody>
</table>

### Syntax

@Prompt( [ style ] : [ NoSort ] ; title ; prompt ; defaultChoice ; choiceList ; filetype )

### Parameters

[ style ]

Keyword. Required. Indicates the type of dialog box to display. May be any of the following:
Include the square brackets ([ ]); these identify the style parameters as keywords. If no [NoSort] keyword is provided, follow the style parameter with a semicolon (;).

[NoSort]

Keyword. Optional. Include this keyword if you want the members of choiceList to appear in the exact order in which you enter them. If you omit this keyword, the members of choiceList are sorted alphabetically.

title

Text. The text you want displayed in the dialog box's title bar. Required for all styles, although you can specify a null string with "". The maximum number of characters you can include in a title is 65. Provide a null string with the [ChooseDatabase] keyword; you cannot replace the default "Choose Database" title.

prompt

Text. The text you want displayed within the dialog box. Required for all styles, except LocalBrowse. If you use a formula for prompt and that formula returns a list, only the first item in the list is displayed as the prompt. To display the entire list, use @Implode. You can specify a field name to display the contents of the field as the prompt, but the field must be a text field. If it is a number or datetime field, precede it with @Text. @NewLine cannot be used in prompt. Use @Char(13) to insert a carriage return. The maximum number of characters you can include in the text that displays is 255. Provide a null string for the [ChooseDatabase] keyword; you cannot display custom text in the Choose Database dialog box.

defaultChoice

Text. The value that will be used as the default value for the user's input. The input section of the dialog box is primed with the value; the user can either accept it by clicking OK or replace it with another value. Not applicable to dialog boxes of style [Ok], [YesNo], [YesNoCancel], [LocalBrowse], or [Password]. Required for all other styles. For [OkCancelListMult], you can specify multiple default values as a text list "item1":"item2."

choiceList
Text list. The values that you want displayed in the dialog box's list box. The user can select one of these values as the input. Separate the values with colons, as in: "PHONE.NSF":@MailDbName. Each value in your list can be a text string, or an @function that returns a text string. Required only with styles [OkCancelList], [OkCancelCombo], [OkCancelEditCombo], and [OkCancelListMult].

`filetype`

Text. A value that specifies the types of files to display initially: "1" for NSF files only; "2" for NTF files only; "3" for files of all type. Required only with style [LocalBrowse].

**Return value**

`choice`

- If the user enters a value, returns the value as text or a text list.
- If the user selects Yes, returns 1 (True).
- If the user selects No, returns 0 (False).
- If the user selects Cancel, formula evaluation stops. The exception is [YesNoCancel], which returns -1 if the user selects Cancel.
- `@Prompt([OkCancelEdit])` returns only the first 254 characters of the text entered.

**Usage**

Use `@Prompt` in field formula, toolbar button, manual agent, form action, and view action formulas. This function does not work in column, selection, mail agent, or scheduled agent formulas, and has limited usefulness in window title and form formulas.

You cannot use this function in Web applications.

**Language cross-reference**

InputBox function of LotusScript language

DialogBoxCanceled property of LotusScript NotesUIDocument class

Prompt method of LotusScript NotesUIWorkspace class

**Examples: @Prompt**

1. `[[Ok] displays an informational message; the user clicks OK to close the dialog box. Use this style when you want to inform the user about something, without receiving anything back except an acknowledgement.]
   
   ```
   @Prompt([Ok];"Reminder";"Don't forget to run backup tonight.")
   ```

2. `[[YesNo] displays a warning, and gives the user a chance to proceed or cancel the operation. If the user selects Yes the numeric value 1 is returned. If the user selects No the numeric value 0 is returned.]
   
   ```
   @Prompt([YesNo];"Send memo?";"This memo will be sent to everyone listed in the To, CC, and BCC fields.")
   ```

3. `[[YesNoCancel] also displays a warning, and gives the user a chance to select Yes, No, or Cancel. If the user selects Cancel, the value -1 is returned.]
   
   ```
   result=@Prompt([YesNoCancel];"Send memo?";"This memo will be sent to everyone listed in the To, CC, and BCC fields")
   ```

4. `[[OkCancelEdit] prompts the user to enter his or her name, which is returned as a text string. The name defaults to the current user's Notes/Domino user name, which is calculated using @UserName. If the user selects Cancel, Lotus Notes/Domino cancels the formula evaluation. Note that @Prompt([OkCancelEdit]) returns only the first 254 characters of the text entered.]
   
   ```
   @Prompt([OkCancelEdit];"Enter Your Name";"Type your name in the box below.";@UserName)
   ```
5. [OkCancelList] displays a list box with database names (sorted alphabetically), prompts the user to select a database, and returns that database’s name as a text string for use in a subsequent operation. If the user selects Cancel, Lotus Notes/Domino cancels the formula evaluation.

   The third option in the list is the current user’s own mail database, the name of which is calculated with @MailDbName. The user must select one of the listed options; by default, Schedule is highlighted (the value listed as the default must also be included in the display list).

   @Prompt([OkCancelList]; "Select a Database"; "Select a database to open."; "Schedule"; "Schedule":"Phone Book"; @Subset(@MailDbName;–1))

6. [OkCancelCombo] displays a dialog box similar to example 5, except that a drop-down list is used, so that initially only the default value is displayed. The user clicks the down arrow on the box to display the rest of the list. As in example 5, the user must select one of the listed values; by default, Schedule is selected. This function returns the user’s selection. If the user selects Cancel, Lotus Notes/Domino cancels the formula evaluation.

   @Prompt([OkCancelCombo]; "Select a Database"; "Select a database to open."; "Schedule"; "Schedule":"Phone Book"; @Subset(@MailDbName;–1))

7. [OkCancelEditCombo] is similar to example 6, except here the user can edit the text box and type in any database name; this way, the user is not limited to the selections in the list. This function returns the user’s selection or entry. If the user selects Cancel, Lotus Notes/Domino cancels the formula evaluation.

   The default value must be included in the list, or the text box that displays initially will be blank.

   @Prompt([OkCancelEditCombo]; "Select a Database"; "Select a database to open, or type a database specification."; "Schedule"; "Schedule":"Phone Book"; @Subset(@MailDbName;–1))

8. [OkCancelListMult] displays a list of names, from which the user can select one or more (Mary Tsen appears as the default selection). This function returns the user’s selection(s). If the user selects Cancel, Lotus Notes/Domino cancels the formula evaluation.

   The default value must be included in the list.

   @Prompt([OkCancelListMult]; "Select a Name"; "Select one or more names as recipients for this request."; "Mary Tsen"; "Mary Tsen"; "Bill Chu"; "Michael Bowling"; "Marian Woodward")

9. [Password] displays a dialog box where the user can enter a password. Lotus Notes/Domino does not display the password on the screen. This function returns the password.

   @Prompt([Password]; "Password"; "Enter the password for Approach database.")

10. [LocalBrowse] provides controls and displays that allow you to browse and select a name from the local file system. This example opens the Notes/Domino database file the user selects from the local browser. The "1" restricts the initial display to .nsf files.

    file := @Prompt([LocalBrowse]; "Select a database to open"; "1");
    @If(file = ""; @Return(1); "");
    @Command([[FileOpenDatabase]; "":@Left(file; "")])

11. This code, in a form hotspot button, displays the Choose Database dialog box then sets field values on the document (which must be in edit mode) based on the result.

    result := @Prompt([ChooseDatabase];"";"");
    FIELD Server := result[1];
    FIELD Filename := result[2];
    FIELD Title := result[3]

©ProperCase

Converts the words in a string to proper-name capitalization: the first letter of each word becomes uppercase, all others become lowercase.

Syntax

@ProperCase( string )
Parameters

string

Text. The string you want to convert.

Return value

properString

Text. The string, converted to proper-name capitalization.

Usage

A "word" is a consecutive set of characters with no spaces. Hyphenated words are considered two words, as are words separated by any other punctuation except an apostrophe.

Language cross-reference

StrConv function of LotusScript language

Examples: @ProperCase

1. This example returns Every Child Loves Toys.
   @ProperCase("every CHILD LOves toys")
2. This example returns 3–Digit Code.
   @ProperCase("3–digit code")
3. This example returns Los Angeles if the string in the field named City contains the string los angeles, Los Angeles, LOS ANGELES, los Angeles, or any other variation.
   @ProperCase(City)

@Random

Generates a random number between 0 and 1, inclusive.

Syntax

@Random

Usage

To generate a random number between any two numbers x and y, use the formula

\[
(\frac{y - x}{x} + \text{@Random}\times x)
\]

Language cross-reference

Rnd function of LotusScript language

Examples: @Random

This formula generates a random number between 7 and 22, inclusive. For example, it might return 13.

15 \times \text{@Random} + 7
@RefreshECL

Copies the administration execution control list from a specified Address Book and name to your personal workstation ECL.

Syntax

@RefreshECL( server : database ; name )

Parameters

server : database

Text list. The server location and file name of the Address Book. Omit server or specify it as "" (null) for the local Notes/Domino directory.

name

Text. The name of the ECL. Specify "" (null) for the unnamed ECL.

Usage

You cannot use this function in Web applications.

Examples: @RefreshECL

This formula refreshes your personal workstation ECL from the administration ECL named "Developers" in the Address Book on the server Marketing.

@RefreshECL("Marketing" : "names.nsf"; "Developers")

@RegQueryValue

Queries the Windows registry for a specified value.

Note: This function is new in Release 5.0.2.

Syntax

@RegQueryValue( keyName ; subKeyName ; valueName )

Parameters

keyName

String. HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE. The registry key you want to query.

subKeyName

String. The name of the subkey under keyName that you want to query.

valueName

String. The name of the registry value you want to find.

Return value

string

The value associated with the value name specified in the valueName parameter.
Usage
@RegQueryValue is intended for use on the Windows platform. It returns an empty string on non-Windows platforms.

Examples: @RegQueryValue
This example obtains the current registered Notes executable directory in Windows.
@RegQueryValue("HKEY_LOCAL_MACHINE; "Software\Lotus\Notes\5.0; "Path")

REM
The REM reserved word allows you to add explanatory remarks (comments) to a formula. Quotation marks or braces delimit the text of the remark.

Note: Using braces to delimit a remark is new with Release 6.

Syntax
REM " comments " ;
REM { comments } ;

Usage
The backslash ( \ ) serves as an escape character in a remark. To embed quotation marks in a remark delimited by quotation marks, precede each embedded quotation mark with a backslash. To embed a right brace in a remark delimited by braces, precede each embedded right brace with a backslash. To embed a backslash in a remark, type two backslashes.

A compiled formula does not distinguish between quotation marks and braces. When you open a design element containing formulas, braces delimit all constants including those previously specified with quotation marks. A backward slash prefixes a right brace previously specified in a remark delimited by quotation marks.

If a comment doesn’t fit on one line, add additional REM statements to complete the comment.

Language cross-reference
%Rem directive of LotusScript language
Rem statement of LotusScript language
Comment property of Lotus Script NotesAgent class
Comment property of Lotus Script NotesOutline class
DisplayComment property of Lotus Script NotesRichTextDocLink class
Comment property of Lotus Script NotesTimer class
Comment property of Java Agent class
Comment property of Java Outline class

Examples: REM
1. This formula contains five lines of comments before the code.
REM "6/15/95"
REM "The following formula calculates the date"
REM "for the DueDate field"
REM "DueDate is the Date field + thirty days"
REM;
  @Adjust(Date; 0;0;30;0;0;0)

2. This formula contains five lines of comments before the code.
REM {1/15/01};
REM {The following formula calculates the date};
REM {for the "DueDate" field};
REM {{"DueDate" is the Date field + thirty days}};
REM;
  @Adjust(Date; 0;0;30;0;0;0)

---

@Repeat
Repeats a string a specified number of times.

Syntax @Repeat( string ; number ; numberchars )

Parameters

string
Text. The string you want to repeat.

number
Number. The number of times you want to repeat string.

numberchars
Number. Optional. The maximum number of characters you want returned. @Repeat truncates the result to this number.

Return value

repeatedString
Text. The string, repeated number times until numberchars(if specified) is reached.

Usage

The resultant string cannot be larger than 1,024 characters.

Language cross-reference

UString function of LotusScript language

Examples: @Repeat

1. This example returns HelloHelloHello.
   @Repeat("Hello";3)

2. This example returns ByeBy.
   @Repeat("Bye";2;5)

3. This example returns Great Month! Great Month! Great Month! in the Comments field if the amount in the field named Sales is greater than or equal to 100,000; otherwise it returns the string Good Month.
   FIELD Comments:=@If(Sales>=100000;@Repeat("Great Month!";3);"Good Month");
@Replace

Performs a find-and-replace operation on a text list.

Syntax
@Replace( sourcelist ; fromlist ; tolist )

Parameters
sourcelist
Text list. The list whose values you want to scan.

fromlist
Text list. A list containing the values that you want to replace.

tolist
Text list. A list containing the replacement values.

Return value
replacedList
Text list. The sourcelist, with any values from fromlist replaced by the corresponding value in tolist. If none of the values in fromlist matched the values in sourcelist, then sourcelist is returned unaltered.

Language cross-reference
Replace function of LotusScript language

Examples: @Replace
1. Both sourcelist and fromlist contain "Orange", which is the first value in fromlist. The first value in tolist replaces "Orange" in sourcelist. No other matches were found, so the remainder of sourcelist is left intact; the result is shown below:
   @Replace("Red":"Orange":"Yellow":"Green":"Orange":"Blue":"Black":"Brown")

<table>
<thead>
<tr>
<th>sourcelist</th>
<th>fromlist</th>
<th>tolist</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>Orange</td>
<td>Black</td>
<td>Red</td>
</tr>
<tr>
<td>Orange</td>
<td>Blue</td>
<td>Brown</td>
<td>Black (replaces &quot;Orange&quot;)</td>
</tr>
<tr>
<td>Yellow</td>
<td></td>
<td></td>
<td>Yellow</td>
</tr>
<tr>
<td>Green</td>
<td></td>
<td></td>
<td>Green</td>
</tr>
</tbody>
</table>

2. This formula looks at the Categories field in each document that it runs against. If one of the keywords in a document’s Categories field is "To be assigned" then that keyword is replaced with the name stored in that document’s AssignedTo field.

   FIELD Categories:= @Trim(@Replace(Categories; "To be assigned"; AssignedTo));

   You have a database where you log service requests. Incoming requests are automatically categorized as "To be assigned" by a mail/paste filter. Each day, you review the new (unassigned) service requests, and assign them to technicians by entering the appropriate name in the AssignedTo field. Once a request has been assigned, you want it to appear under that technician’s name in the view, instead of under "To be assigned."
Rather than manually categorizing each document a second time, you can write a filter macro, like the one above, to delete the documents from the “To be assigned” category and add them to the appropriate technician categories.

@ReplaceSubstring
Replaces specific words or phrases in a string with new words or phrases that you specify. Case sensitive.

Syntax
@ReplaceSubstring( sourceList ; fromList ; toList )

Parameters
sourceList
Text or text list. The string whose contents you want to modify.

fromList
Text or text list. A list containing the words or phrases that you want to replace.

toList
Text or text list. A list containing the replacement words or phrases.

Return value
newSourceList
Text or text list. The sourceList, with any values from fromList replaced by the corresponding value in toList. If none of the values in fromList matched the values in sourceList, then sourceList is returned unaltered.

Usage
If more strings are specified in the fromList than the toList, the extra strings in fromList are replaced with the last string in toList. Extra strings in toList are ignored. If no matches are found, @ReplaceSubString returns the unmodified sourceList.

If a list is specified for fromList, each subsequent list item is scanned against the resulting sourceList, with prior list item substitutions performed.

For example:
@ReplaceSubstring("first";"first":"second";"second":"third")

returns third.

First, @ReplaceSubstring substitutes "second" for "first" from the first list item in fromList. The resulting sourceList is now "second." The function substitutes "third" for "second" from the second list item in fromList.

Tip: Use @ReplaceSubString to remove carriage returns from text by replacing them with " " or "."

Language cross-reference
Mid statement of LotusScript language
Examples: @ReplaceSubstring

1. This example returns "I hate apples."
   @ReplaceSubstring( "I like apples" ; "like" ; "hate" )
2. This example returns "I hate peaches."
   @ReplaceSubstring( "I like apples" ; "like" : "apples" ; "hate" : "peaches"")
3. This example replaces all carriage returns in the Description field’s text with blank spaces.
   @ReplaceSubstring(Description;@Newline;"

@ReplicaID

Returns the replica ID of the current database.

Note: This @function is new with Release 6.

Syntax
@ReplicaID

Return value
title

Text. The replica ID of the current database.

Usage
The replica ID is a 16-character combination of letters and numbers that identifies a Notes database. Any databases with the same replica ID are replicas of one another.

Language cross-reference
ReplicaID property of LotusScript NotesDatabase class

ReplicaID property of Java Database class

DbReplicaID property of LotusScript NotesRichTextDocLink class

Examples: @ReplicaID

This agent mails the replica ID of the current database to the current user.
@MailSend(@UserName;"";"Replica ID";@ReplicaID)

@Responses

Returns the number of responses (in the current view) to the document.

Syntax
@Responses

Return value
numResponses

Special text. The number of responses to the document. Special text cannot be converted to a number.
Usage
Use @Responses in window title formulas. This function does not work in any other formula.

You cannot use this function in Web applications.

Language cross-reference
Responses property of LotusScript NotesDocument
Responses property of Java Document

Examples: @Responses
1. This example returns 5 if there are five responses to the document.
   @Responses
2. This formula returns the string No one has responded to this document if there are no responses to the current document; otherwise a blank is returned.
   @If(@Responses=0; "No one has responded to this document"; "")

@Return
Immediately stops the execution of a formula and returns the specified value. This is useful when you only want the remainder of the formula to be executed only if certain conditions are True.

Syntax
@Return( value )

Parameters
value

The value you want returned. You can specify another @function such as @Error, or a text string such as "Formula stopped," or a Boolean value (True or False). If you don’t want anything returned, use the null string ("").

Return value
result

Returns value.

Usage
@Return is most useful in field formulas, agents that run formulas, and toolbar buttons. Generally, you use it with @If to determine whether to perform @Return or to perform one or more other statements.

@Return should not be used in column formulas.

Language cross-reference
End statement of LotusScript language
Exit statement of LotusScript language

Examples: @Return
1. This formula displays a dialog box offering the user a Yes/No choice. If the user selects Yes, the next document in the view is opened; if the user selects No, the formula stops and nothing more happens.
2. This formula tests whether an environment variable called OrderNumber has been stored in the user's NOTES.INI or Notes Preferences file. If there is no such variable stored, @SetEnvironment is used to initialize it to zero. If a value has already been stored, @Return is used to return it and stop the formula from executing.

```
@If(@Environment(OrderNumber)="";@SetEnvironment("OrderNumber"; "0");@Return(@Environment("OrderNumber")))
```

3. The following code, when added to a field that displays the result of a database lookup, returns a customized error message if an error is encountered during that lookup. The temporary variable, "lookup," retrieves the job title (located in column 3 of the "People" view) of the person listed in the first sorted column of the "People" view. If an error is encountered during the lookup, the field displays the specified error message in a dialog box and "1" displays in the field, indicating that there was an error encountered.

```
lookup := @DbLookup(""
estarting point of record
"People"; "Jackie Brown"; 3);
@If(IsError(lookup); @Return(@Prompt([OK];"Error";"Error locating the requested job title.
Aborting lookup")); lookup)
```

---

**@Right**

Returns the rightmost characters in the string. You can specify the number of rightmost characters you want returned, or you can indicate that you want all the characters to the right of a specific substring.

**Syntax**

```
@Right( stringToSearch ; numberOfChars ) or @Right( stringToSearch ; subString )
```

**Parameters**

- **stringToSearch**
  
  Text. The string whose rightmost characters you want to find.

- **numberOfChars**
  
  Number. The number of characters to return. If the number is 2, the last two characters of `stringToSearch` are returned; if the number is 5, the last five characters are returned, and so on.

- **subString**
  
  Text. A substring of `stringToSearch`. @Right returns all of the characters to the right of `subString`. It finds `subString` by searching `stringToSearch` from left to right.

**Return value**

- **resultString**
  
  Text. The rightmost characters in `stringToSearch`. The number of characters returned is determined by either `numberOfChars` or `subString`. @Right returns "" if `subString` is not found in `stringToSearch`.

**Language cross-reference**

- Right function of LotusScript language
- RightBP function of LotusScript language
- StrRight function of LotusScript language
Examples: @Right
1. This example returns "ace," the rightmost 3 characters in the string.
   @Right("Lennard Wallace";3)
2. This example returns "Wallace," which represents everything to the right of the first occurrence of the blank space.
   @Right("Lennard Wallace";"")
3. This example returns "man" if the Author field contains "Timothy Altman."
   @Right(Author;3)
4. This example returns "Altman" if the Author field contains "Timothy Altman."
   @Right(Author;"")

@RightBack
Returns the rightmost characters in a string.

Syntax
@RightBack( stringToSearch ; numberOfChars )

@RightBack( stringToSearch ; subString )

Parameters
stringToSearch
Text. The string whose rightmost characters you want to find.

numberOfChars
Number. Counting from left to right, the number of characters to skip. All the characters to the right of that number are returned.

subString
Text. A substring of stringToSearch. @RightBack returns all the characters to the right of subString. It finds subString by searching stringToSearch from right to left.

Return value
resultString
Text. The rightmost characters in stringToSearch. The number of characters returned is determined by either numberOfChars or subString.

Language cross-reference
StrRightBack function of LotusScript language

Examples: @RightBack
1. This example returns "nard Wallace."
   @RightBack("Lennard Wallace";3)
2. This example returns a blank.
   @RightBack("Lennard Wallace";"")
3. This example returns "Wallace."
4. This example returns "othy Altman" if the name in the field named Author is Timothy Altman.
   @RightBack(Author;3)

5. This example returns lapalooza if the word in the show field is Lalapalooza.
   @RightBack(show;"La")

6. This example returns palooza if the word in the show field is lalapalooza.
   @RightBack(show;"la")

Note: @RightBack returns the string to the right of the last occurrence of the substring you are searching for.

@Round
Rounds the designated number to the nearest whole number; if an additional number is specified, it is used as the rounding factor.

Syntax
@Round( number ) @Round( number ; factor )

Parameters

number

Number or number list. Numbers to be rounded.

factor

Number. Optional. The rounding factor to use. For example, if factor is 10, @Round rounds to the nearest number that is a factor of 10. If you don't specify a factor, the number is rounded to the nearest whole number.

Return value

roundedNumber

Number. The value of number, rounded to the specified factor or to the nearest whole number. If number is a list, each number in the list is rounded to the specified factor or to the nearest whole number.

Usage

When using this function with a number list, the list concatenation operator takes precedence over any other operators.

For more information, see "List concatenation operator" in the "Formula Language Rules" chapter.

Language cross-reference

Round function of LotusScript language

Examples: @Round

1. This example returns 2.
   @Round(2.499)

2. This example returns 3.
   @Round(2.5)

3. This example returns 2.
@Round(1.5)

4. This example returns 12340 if the number in the field named NumberOfEmployees is 12338.
   @Round(NumberOfEmployees;10)

5. This example returns 1:3:3:4.
   @Round(1.333;2.897654:3.1:4)

6. This example returns 4510:45010:450010.
   @Round(4505:45005:450005;10)

7. This example returns 3.1430E+00 in a number field that has scientific formatting and is set to display four decimal places.
   @Round(3.142857; 0.001)

@Second

Extracts and returns the seconds value from the specified time–date.

Syntax
@Second( time–date )

Parameters

    time–date  

Time-date.

Return value

    seconds  

Number. The number of seconds in the second part of the time. Returns -1 if the time-date provided contains only a date and not a time value.

Language cross-reference

Second function of LotusScript language

Examples: @Second

1. This example returns 45.
   @Second([9:30:45])

2. This example returns 45 if the current time is 12:30:45 P.M.
   @Second(@Now)

3. This example returns 45 as a text string if the contents of the field named Date is any time–date value in which the number of seconds is 45.
   @Text(@Second(Date))

SELECT

The SELECT reserved word defines criteria for the selection of documents in an agent that runs a formula, in a view, or during replication. You use a SELECT statement before an expression to define the set of documents that you want to change, see in a view, or replicate.

Syntax

SELECT formula ;
Usage

- In an agent, you can use the Agent Properties box to select the documents you want to act upon.
- In an agent that runs a formula, you can include a SELECT statement in the formula. The agent acts upon the documents selected with the Agent Properties box and the documents selected by the SELECT statement.
- In a view, you can use the Search Builder to select the documents you want to see in the view. You can use SELECT to select documents and provide more complicated conditions for replication.
- For selective replication, you can use the Search Builder to select the documents you want to replicate. You can use SELECT to select documents and provide more complicated conditions for replication.

Using SELECT in the formula eliminates the need to go through the database to select the documents. You can run the filter macro on all the documents in the database, and the SELECT statement performs the selection process.

The word SELECT is automatically prepended to the view selection formula when the formula is saved.

Use SELECT @All to select all documents for an operation (for example, use it in the selection formula for a view that displays all of the database’s documents). @All should never be used without the SELECT reserved word. If your formula contains @All by itself, Lotus Notes/Domino appends the SELECT @All statement to your formula:

```
@All;
SELECT @All;
```

If you compare a field to a value (for example, Year > 1995) and the field is unavailable, the comparison is false. However, you should check for fields that may not be present with @IsUnavailable.

This reserved word does not work in column, hide-when, section editor, window title, hotspot, field, form, or form action formulas.

SELECT is not intended for use in toolbar buttons.

Language cross-reference

SelectionFormula property of LotusScript NotesView class

Examples: SELECT

1. You want to change the contents of the Status field in several documents to Closed. However, you do not want to change the Status field of any document that contains the value Unsigned Contracts in the Categories field.

   To make the desired change, you write and run an agent that runs a formula. When you write the formula, you specify the documents that you want Lotus Notes/Domino to scan to make the change. By adding a SELECT statement to the formula, you can further limit the documents that Lotus Notes/Domino looks at when you run the agent.

   ```
   SELECT Categories != "Unsigned Contracts";
   FIELD Status := "Closed";
   ```

2. This replication formula limits replication to documents that contain a Year field whose value is greater than 1995.

   ```
   SELECT @IsAvailable(Year) & Year > 1995
   ```

3. This replication formula limits replication to documents that do not contain a Year field or whose Year field is greater than 1995.

   ```
   SELECT !(@IsAvailable(Year) & Year > 1995)
   ```
@Select

Returns the value that appears in the number position. If the number is greater than the number of values, @Select returns the last value in the list. If the value in the number position is a list, returns the entire list contained within the value.

Syntax
@Select( number ; values )

Parameters
number

Number. The position of the value you want to retrieve.

values

Any number of values, separated by semicolons. A value may be a number, text, time-date, or a number list, text list, or time-date list.

Examples: @Select
1. This example returns 3.
   @Select(3;1;2;3)
2. This example returns 3.
   @Select(5;1;2;3)
3. This example returns Apr;May;Jun.
   @Select(2;"Jan":"Feb":"Mar":"Apr":"May":"Jun";
   "Jul":"August":"Sep":"Oct":"Nov":"Dec")
4. This example returns San Diego;Sydney;New York;Amsterdam if the field named TrainingCenters contains these city names.
   @Select(3;SalesOffices;ServiceOffices;TrainingCenters)

@ServerAccess

Checks if a specified user has a specified administrative access level to a server.

Note: This @function is new with Release 6.

Syntax
@ServerAccess( [ access ] ; userName ; serverName )

Parameters
[ access ]

Keyword. Supply one of the following keywords to represent the access level you want to check for:

[ACCESS]

User has administrative access to the server.

[CREATEDATABASE]

User can create a database on the server.
[CREATEREPlica]
User can create a replica of a database on the server.

[CREATETEMPLATE]
User can create a master template on the server.

[DATABASEACCESS]
User has administrative access to the server, which enables him or her to perform all the tasks that administrators with Access level access can perform, except users with DatabaseAccess cannot issue remote console commands.

[FULLACCESS]
User has full administrative access to the server and is given manager access to all databases hosted by the server, regardless of the database’s ACL settings.

[REMOTEACCESS]
User can issue remote console commands to the server.

[RESTRICTEDSYSTEMACCESS]
User can issue only those operating system commands that are listed as Restricted System commands.

[SYSTEMACCESS]
User can issue operating system commands to the server.

[TRACKMESSAGE]
User can track email messages, but cannot view the contents of the Subject field of mail memos.

[TRACKMESSAGESUBJECT]
User can track email messages and can view the contents of the Subject field of mail memos.

[VIEWONLYACCESS]
User can issue a subset of remote console commands that supply information about the server; they cannot execute remote commands that affect the server’s operation.

These access levels are set by the server administrator on the Security tab of the Current Server Document in the Server settings found on the Configuration tab of the Domino Administrator client.

userName
Text; not case-sensitive. Hierarchical name of the user whose access you want to check, enclosed in quotation marks. If you supply a short name, this function returns zero. You can use @UserName to supply the name of the current user to @ServerName.

serverName
Optional. Text; not case-sensitive. Name of the server you want to test the user’s access level to, enclosed in quotation marks. If not provided, tests the user’s access to the server hosting the current database. If the current database is Local, tests the user’s access to the server that is listed as the Administrative server in the database ACL for the current database. If no Administrative server is set, returns zero.

**Note:** This parameter is required when using @ServerAccess in a toolbar button.

### Return value

flag

Boolean.

- 1 (True) indicates that the specified user has the specified access
- 0 (False) indicates that the specified user does not have the specified access

### Examples: @ServerAccess

1. This code, when added as the default value of a field in a database on the ocean/bay server, returns 1 if Luisa Albright is listed as having standard administrative access to the ocean/bay server in the server document for ocean/bay.
   
   ```
   @ServerAccess([ACCESS];"Luisa Albright/bay";"ocean/bay")
   ```

2. This code, when added as the default value of a field in a Local database that has ocean/bay selected as its Administrative server on the Advanced tab of the database’s ACL dialog box, returns 1 if Luisa Albright has standard administrative access to the ocean/bay server.
   
   ```
   @ServerAccess([ACCESS];"Luisa Albright/bay")
   ```

3. This code, when added as the default value of a field in a database running on the ocean/bay server, returns 1 if the current user has full access to the ocean/bay server and all of its databases.
   
   ```
   @ServerAccess([FULLACCESS];@UserName)
   ```

4. This code, when added as the default value of a field, returns 0 because it does not recognize the short user name.
   
   ```
   @ServerAccess([ACCESS];"Luisa Albright";"ocean/bay")
   ```

5. This code, when added as the default value of a field, returns 0 if Luisa Albright does not have full access to the ocean/bay server and all of its databases.
   
   ```
   @ServerAccess([FULLACCESS];"Luisa Albright/bay";"ocean/bay")
   ```

### @ServerName

Returns the name of the server containing the current database. When the database is local, returns the user name.

**Note:** This @function is new with Release 6.

### Syntax

@ServerName

### Return value

`servername`

Text. The name of the server containing the current database or the user name if triggered from a local database.
Language cross-reference
ServerName property of LotusScript NotesSession class
Server property of LotusScript NotesDatabase class
ServerName property of Java Session class
Server property of Java Database class

Examples: @ServerName
1. This formula, when added to a hotspot button on a form running on the acme/central server, displays a Server name message box that reads "CN=acme/O=central."
   @Prompt([OK]; "Server name"; @ServerName)
2. This formula, when added to an action button on a form running on the acme/central server, displays a Server name message box that reads "acme."
   @Prompt([OK]; "Server name"; @Name([CN]; @ServerName))
3. When this code is added to a client toolbar button it displays "CN=Mary Anne Admin/O=central" if the button is triggered by Mary Anne while she is working with a form from a local database.
   @Prompt([OK]; "Server name"; @ServerName)

@Set
Assigns a value to a temporary variable for use within a formula.

Syntax
@Set( variableName ; value )

Parameters
variableName
Text. The name of a temporary variable.

value
Text, number, or time-date. The value you want to give to variableName.

Usage
With Release 6, you no longer need to declare the variable receiving the assignment prior to setting its value with @Set. For R5 and earlier clients, declare the variable by assigning it a null value at the beginning of the formula:
TemporaryVariable:=""

Language cross-reference
Set statement of LotusScript language

Examples: @Set
This formula determines whether the FirstName field is blank. If so, it sets the variable FullName to the concatenation of the Title field with the LastName field, as in "Ms. Tsen." If the FirstName field contains a value, the variable FullName is instead set to the concatenation of the FirstName with the LastName, as in "Mary Tsen."
@SetDocField

Given the unique ID of a document, sets the value of a specific field on that document. The document must reside in the current database.

Syntax

@SetDocField( documentUNID; fieldName; newValue )

Parameters

documentUNID

Text. The unique ID of a document. @DocumentUniqueID specifies the unique ID of the current document. To specify the unique ID of the parent document, use $Ref as the first parameter. $Ref is a special field on a response document that contains the unique ID of the parent document.

fieldName

Text. The name of a field on the document, enclosed in quotation marks. If you store the field name in a variable, omit the quotation marks here.

newValue

Text or text list; number or number list; time-date or time-date range. The value you want to give to the field.

Usage

This function does not work in column or selection formulas. @SetDocField is particularly useful in field, button, and agent formulas.

Note: Starting with Release 6, you can use @SetDocField to set the value of a field in the current document, not just in other documents in the same database.

Language cross-reference

FieldSetText method of LotusScript NotesUIDocument class

GetDocumentByUNID method of LotusScript NotesDatabase class

ReplaceItemValue method of LotusScript NotesDocument class

getDocumentByUNID method of Java Database class

replaceItemValue method of Java Document class

Examples: @SetDocField

1. This formula, if placed on a button in a response form, changes the Subject of the parent document to "More people are commuting by bicycle." $Ref is a special field on a response document that contains the unique ID of the parent document.

   @SetDocField($Ref; "Subject"; "More people are commuting by bicycle")

2. This button formula changes the value of the name field in the current document to Joseph Riley:

   @SetDocField(@DocumentUniqueID;"name";"Joseph Riley")
3. In a database, you want to update the parent Project document whenever its child Status document changes. Each Project document has one Status document. Specifically, you want to update the latestStatus field on the Project document so that it reflects the contents of the lastAction field on the child Status document.

You write this input translation formula for the lastAction field on the Status form:

```plaintext
@SetDocField($Ref; "latestStatus"; lastAction);
lastAction
```

4. This button formula uses @DbLookup to retrieve the unique ID of a particular document. It then changes the value of the "employee title" field in that document to "sales associate."

```plaintext
@SetDocField(@DbLookup("**"; "Magnet";"Personnel.nsf"; "Staff"; "Joe Smith";
"uniqueid"); "Employee Title"; "Sales Associate")
```

---

**@SetEnvironment**

Sets an environment variable stored in the user’s notes.ini file (Windows, OS/2, and UNIX) or Notes Preferences file (Macintosh).

**Syntax**

```plaintext
@SetEnvironment( variableName ; value )
```

**Parameters**

`variableName`

Text. The name of the environment variable, enclosed in quotation marks. If you enter a text list for the `variableName`, then every variable named in that list receives the specified `value`. If you store the field name in a variable, omit the quotation marks here.

`value`

Text. The value you want to give to `variableName`. If you use a text list for `value`, only the first value in the list is used; the rest are ignored.

**Usage**

Use @SetEnvironment when you want to set an environment variable from within another @function (such as @If or @Do). To set the environment variable outside of an @function, use @Environment or the ENVIRONMENT keyword.

@SetEnvironment cannot be used in column or selection formulas. Some formulas, such as scheduled agents, are run on the server instead of the user’s workstation. In this case, the environment variables affected are the server’s environment variables, not the workstation’s.

To get the value of an environment variable, use @Environment.

You cannot use this function in Web applications. However, in Web applications, you can use predefined field names to gather information about the Web user’s environment by requesting Common Gateway Interface (CGI) environment variables.

This function prepends a dollar sign ($) to the variable name when it stores the variable in the notes.ini (or Notes Preference) file. Use the SetEnvironmentVar method of the LotusScript NotesSession class or the setEnvironmentVar method of the Java Session class if you want to create a variable without the prepended dollar sign.
@SetField

Assigns a value to a field stored within a document (use @Set for temporary variables). This is similar to using the FIELD keyword, except that @SetField can be used within another @function. If the field does not exist, this command creates it and applies the specified value to it.

Syntax

@SetField( fieldName ; value )

Parameters

fieldName

The name of the field whose value you want to set, enclosed in quotation marks.

value

The value you want to give to fieldName. The value must be the same data type as the field; for example, if the field is numeric, the value must be a number.

Usage

This function is most useful in agents, hotspot buttons, actions, and toolbar buttons. It does not work in column, selection, hide-when, window title, or form formulas.

With Release 6, you no longer need to declare the field receiving the assignment prior to setting its value with @SetField. For R5 or earlier clients, declare the field at the beginning of the formula, as follows:

FIELD Fieldname:=Fieldname;

The field that @SetField creates and assigns the specified value to if the specified field does not exist in the document is not visible to the user. You can remove a field added to a form this way using the @DeleteField function.

Examples: @SetField

1. This formula checks the value of the Priority field; if the Priority is Low or Medium, the Status field is set to Closed; otherwise, the Status is set to Open. Before @SetField is encountered in the formula, the Status field is declared using the FIELD keyword.

FIELD Status:=Status;
@If(Priority="Low"|Priority="Medium";@SetField("Status";"Closed")
@SetField("Status";"Open")

2. This code, when used in a view action button, deletes fields x_1 through x_20 in the selected document.
@For(i := 1; i <= 20; i := i + 1;
@SetField("x_" + @Text(i);@DeleteField));

@SetHTTPHeader

In a Web application, sets the value of HTTP headers in the response being generated by the server for the browser client.

Note: This function is new with Release 6.

Syntax

@SetHTTPHeader( responseHeader ; value )

Parameters

responseHeader

String. The name of a response-header field, for example, "Content-Encoding," "Content-Length," or "Set-Cookie." See http://www.w3.org/Protocols for specifications of response headers. The following response headers are read-only and cannot be set or overwritten using this function:

- Connection
- Content-Type
- Date
- Server

value

Text, number, or date. A value for the field. Dates are converted to RFC 1123 format. An empty string (""") removes the header and its value from the HTTP response.

Return value

successOrFailure

Number. @True, or one (1), if the HTTP response header was successfully updated; @False, or zero (0), otherwise.

Usage

@SetHTTPHeader is useful in formulas that run in the context of a browser; the Notes client always returns @False, or zero (0), for this formula.

See @GetHTTPHeader for information on getting a request header value.

Language cross-reference

Headers property of LotusScript NotesMIMEEntity class

Headers property of Java MIMEEntity class

Examples: @SetHTTPHeader

This form action sets the value of the response-header field named "Set-Cookie" to "SHOP_CART_ID=4646." As a result, the browser client registers a cookie for the server using this name and value.

@SetHTTPHeader("Set-Cookie"; "SHOP_CART_ID=4646")
This function appends the Set-Cookie response header to the end of the following standard HTTP response:
HTTP/1.0 200 OK
Date: Thurs, 30 Aug 2001 16:17:52 GMT
Server: Domino/6.0
Content-type: text/html
Content-length: 1538
Last-modified: Mon, 27 Aug 2001 01:23:50 GMT
Set-Cookie: SHOP_CART_ID=4646

@SetProfileField
Sets the value of a field in a profile document or creates a profile document.

Syntax
@SetProfileField( profilename ; fieldname ; value ; uniqueKey )

Parameters
profilename
Text. The name of the profile document that contains the field you want to access. If no profile document exists by this name, Lotus Notes/Domino creates one.

fieldname
Text. The name of the field you want to access.

value
Text. The value to which you want to set the field.

uniqueKey
Text. Optional. A unique key that identifies the profile document.

Return value
value
The value to which you set the field.

Usage
Use this function to create a profile document in a Web application. The EditProfile @command does not work on the Web. If no document exists with the name specified as the first parameter of this function, Lotus Notes/Domino creates a profile document with that name. Use @GetProfileField to access data from the profile document.

Language cross-reference
EditProfile method of LotusScript NotesUIWorkspace class

Examples: @SetProfileField
1. This example sets the contents of the "Profile Categories" field of the "Interest Profile" document to the name of the current platform.
2. This example sets the contents of the "Profile Categories" field of the "Interest Profile" document for the current user to the name of the current platform.

3. This code, when added to the Set Profile Field action button in a Web application, creates a profile document called webProfile, creates a fname field and sets the value of the fname field on the profile document equal to the value of the fname field on the current document.

@SetProfileField("Interest Profile"; "ProfileCategories"; @Platform)

@SetProfileField("Interest Profile"; "ProfileCategories"; @Platform; @UserName)

@SetTargetFrame

Allows you to specify a target frame when opening a view, page, or frameset, or when composing or editing a document.

Note: This @function is new with Release 5.

Syntax

@SetTargetFrame( targetframe )

Parameters

targetframe

Text. The name of the frame that a view, page, frameset, or document should open into.

Usage

Use @SetTargetFrame before opening or refreshing the view, page, or frameset, or before composing or editing a document. The following @commands use the frame specified in the @SetTargetFrame:

- @Command([Compose])
- @Command([EditDocument])
- @Command([OpenFrameset])
- @Command([OpenPage])
- @Command([OpenView])
- @Command([RefreshFrame])

If you specify the newinstance parameter for @Command([OpenView]), the @SetTargetFrame function is ignored.

If you do not specify a viewName for@Command([OpenView]), then the last view is the one that opens in the specified targetFrame of @SetTargetFrame.

If you specify a targetFrame parameter for @Command([RefreshFrame]), the @SetTargetFrame function is ignored.

@SetTargetFrame can be used in action and hotspot formulas.

Language cross-reference

SetTargetFrame method of LotusScript NotesUIWorkspace class

FrameText property of LotusScript NotesOutlineEntry class
FrameText property of Java OutlineEntry class

**Examples: @SetTargetFrame**

Consider 2 framesets -- one that contains "Frame A" and "Frame B" and another frameset nested within "Frame B" that contains "Frame C" and "Frame D."

This example opens the view "My View" in "Frame A" of the first frameset.

```java
@SetTargetFrame("Frame A");
@Command([OpenView]; "My View");
```

This example is code in a button on "Frame C" of the nested frameset. It opens the form "My form" in "Frame D" of the same frameset:

```java
@SetTargetFrame("Frame D");
@Command([Compose]; "My form");
```

**@SetViewInfo**

In Standard Outline views, filters a view to display only documents from a specified category. In Calendar views, filters a view to display only documents that contain a specified string in a specified column.

**Note:** This @function is new with Release 6.

**Syntax**

In a Standard Outline view:

```java
@SetViewInfo( [SETVIEWFILTER] ; filterString ; columnName ; isCategory )
```

In a Calendar view:

```java
@SetViewInfo( [SETVIEWFILTER] ; filterString ; columnName ; exactMatch )
```

**Parameters**

[SETVIEWFILTER]

Keyword. Required. Indicates you want to qualify the documents that display in a view.

`filterString`

Text. Serves as the key to determine which documents display in a view. If this string is present in the column specified in `columnName`, includes the document in the view.

`columnName`

Text. The programmatic name of a column. The column specified here must contain the `filterString` for the document to display in the view.

`isCategory`

Number. Boolean value. Required in a Standard Outline view; not for use in Calendar views. 1 indicates that the column in the `columnName` value is a category. 0 indicates that it is not.

`exactMatch`
Number. Boolean value. Optional in a Calendar view; not for use in Standard Outline views. 1 indicates that the string in the columnName column must exactly match the string specified in filterString. 0 indicates that the filterString does not have to match exactly. For instance, if the filterString is "A," and exactMatch is set to 0, documents with "A" and "A plus" in the column specified in columnName will both be included in the view.

Usage
This @function is useful if you want to filter the documents in a view to display only a subgroup that contains specific data.

Examples: @SetViewInfo
1. This formula, when added to a hotspot button in a form, opens the Customers Standard Outline view, which is categorized by companyName. The view contains documents for people from several companies, but filters the view to display only those documents for individuals who work at the Acme Corp.
   @Command([OpenView];"Customers");
   @SetViewInfo([SETVIEWFILTER];"Acme Corp.";"companyName";1)
2. This code, when added to the Sort action button in a Standard Outline view, filters the contents of the current view to display only those documents that have employeeName fields that contain the current user's name. The view is categorized by employeeName.
   @SetViewInfo([SETVIEWFILTER];@Name([CN];@UserName);"employeeName";1)
3. This code, used in the View by Room action button in the Reservations template (resrc60.ntf), updates the Calendar view to display only calendar entries that specify as their resourceName value the resource chosen by the user from a pick list. $20 is the programmatic name of the Resource column, whose value is determined by the resourceName field.
   choice:=@PickList([CUSTOM] : [SINGLE]; @DbName; "Resources";"View by Room or Resource";"Select the room or resource whose calendar you want to see:"; 1);  
   @SetViewInfo([SETVIEWFILTER];choice;"$20";0)

@Sign
Indicates whether a number is positive, negative, or zero.

Syntax
@Sign( signedNumber )

Parameters
signedNumber

Number. The number whose sign you want to determine.

Return value

sign

Number. May be any of the following values:
- The signed number is negative, -1
- The signed number is zero, 0
- The signed number is positive, 1

Language cross-reference
Sgn function of LotusScript language
Examples: @Sign

This formula sets the result field to "Profit!" if the earnings field is greater than the expenses field, "Loss!" if expenses are greater than earnings, and "Break even" if they are equal.

```plaintext
field result:=result;
difference:=earnings - expenses;
r:=@If( ( @Sign( difference ) = 1); "Profit!"; ( @Sign( difference ) = -1 ); "Loss!"; "Break even" );
@SetField( "result"; r )
```

@Sin

Given an angle in radians, returns the sine of the angle. In a right triangle, the sine of an acute angle is the ratio of the length of its opposite side to the length of the hypotenuse.

Syntax

```
@Sin( angle )
```

Parameters

angle

Number. An angle expressed in radians.

Return value

sine

Number. The sine of angle, to 15 decimal places.

Language cross-reference

Sin function of LotusScript language

Examples: @Sin

1. This formula returns 1, the sine of the angle Pi/2 (90 degrees).
   ```plaintext
   @Sin( @Pi/2 )
   ```

2. You have a triangle ABC. You know the value of angle A in radians, and the lengths of sides a and b. This formula finds angle B, in radians. This formula is a version of the law of sines, which states that for any triangle ABC, (sin A / a) = (sin B / b) = (sin C / c).
   ```plaintext
   @ASin( ( sideB *( @Sin( angleA ) ) ) / sideA )
   ```

@Sort

Sorts a list.

Note: This @function is new with Release 6.

Syntax

```
@Sort( list ; [ order ]; customSortExpression )
```

Parameters

list

Text, number, or time-date list. The values to be sorted. Any alternate data types are returned unchanged.
[order]

Keyword. Optional. You can use the following keywords to specify the order of the sort:

[ACCENTSITIVE]
[ACCENTINSENSITIVE]
[ASCENDING]
[CASESENSITIVE]
[CASEINSENSITIVE]
[CUSTOMSORT]
[DESCENDING]
[PITCHSENSITIVE]
[PITCHINSENSITIVE]

Separate multiple order keywords with a colon(:). By default, the following keywords automatically format the sort order:

[ASCENDING]:[CASESENSITIVE]:[ACCENTSITIVE]:[PITCHSENSITIVE].

You can override a default sort order keyword by specifying its opposite keyword. For example, to override [ASCENDING], specify [DESCENDING] in the @Sort function. If conflicting keywords are passed, the last one in the list affects the sort order.

customSortExpression

Formula. Required when the [CUSTOMSORT] keyword is specified. A formula that uses the temporary variables $A$ and $B$ to compare the values of elements in the list two at a time. If $A$ is greater than $B$, the expression returns @True. If $B$ is greater than $A$, the expression returns @False.

An error is produced if the customSortExpression produces a data type other than a number.

Return value

list

Text, number, or time-date list. The sorted values.

Usage

The ascending, case-, and accent-sensitive sort sequence for the English character set is as follows: the numbers 0-9, the alphabetic characters aA-zZ, the apostrophe, the dash, and the remaining special characters. Pitch-sensitivity affects double-byte languages.

Note: The case sensitive sort only matters when terms are identical except for case. In that instance, the lower case is sorted first. For example, cat, Cat, CAT. If terms are not identical except for case, they are sorted without regard to case.

If you set Unicode standard sorting as the sorting option, you cannot select the following keywords or combinations:

• [PITCHINSENSITIVE]
• [CASESENSITIVE][ACCENTINSENSITIVE]

You specify Unicode standard sorting by setting the notes.ini variable $CollationType to @UCA, or by selecting the "Unicode standard sorting" checkbox that displays in the following dialog boxes:
• Sorting dialog box that displays when you choose File - Preferences - User Preferences - International - Sorting from the main menu
• Database Properties box*
• Design Document Properties box*

*The Unicode option is disabled in the Database and Design Document Properties boxes until you select a default sort order.

For more information on Unicode sorting, see http://oss.software.ibm.com/icu/

A date-time value with a wildcard time (no time specified) equals all date-time values for the same date. For example, the following dates are considered equal:


These values are sorted in random order and may be ordered differently with each sort if multiple sorts are performed on them.

Examples: @Sort

These examples are translation formulas. Assume that the initial value of the field is a list containing:
New Boston, Albany, new york, San Francisco.

1. This formula returns: Albany, New Boston, new york, San Francisco.
   @Sort(@ThisValue)

2. Same as above.
   @Sort(@ThisValue; [ASCENDING])

3. This formula returns: San Francisco, New Boston, new york, and Albany.
   @Sort(@ThisValue; [DESCENDING])

4. This formula returns: Albany, New Boston, New York, San Francisco.
   @Sort(@ProperCase(@ThisValue); [ASCENDING])

These examples are used as the default values for form fields.

1. This formula returns 1009;85;79 if the Price column (the 5th column in the Gear view) contains the prices 79, 85, and 1009 for three entries in the Ski Pants category:
   @Sort(@DbLookup("","Server/Name/Notes":"Ski\Clothing.nsf";"Gear";"Ski Pants";5);[DESCENDING])

2. This formula returns the contents of the movies field in order from the shortest title to the longest; it returns ET;casablanca;The Great Escape when the movies field contains "casablanca":"The Great Escape":"ET":
   @Sort(movies;[CASESENSITIVE];[CUSTOMSORT];@If(@Length($A) < @Length($B);-1;@Length($A) > @Length($B);1);0))

Note that the custom sort keyword overrides the case-sensitivity keyword.

3. This formula returns the following passwords in order from the strongest to the weakest:
   HE5ll+o;Hello;hello, when the pswd1 field contains "Hello", pswd2 field contains "HE5ll+o", and the pswd3 field contains "Hello."
   @Sort(pswd1:pswd2:pswd3;[CUSTOMSORT];@If(@PasswordQuality($A) < @PasswordQuality($B);1;@PasswordQuality($A) > @PasswordQuality($B);-1);0))

4. This formula returns: cat, Cat, CAT, dog when the animals field contains "CAT, Cat, dog, cat".
   @Sort(animals;[CASESENSITIVE])
@Soundex

Returns the Soundex (the Lotus Notes phonetic speller) code for the specified string.

Syntax
@Soundex( string )

Parameters
string

Text. The string whose Soundex code you want.

Return value
code

Text. The Soundex code. You cannot convert it to any other data type.

Usage
This function is used almost exclusively by the Domino Directory. You will rarely use this function.

Examples: @Soundex
1. This example returns F430.
   @Soundex("field")
2. This example returns P430.
   @Soundex("phield")

@Sqrt

Given a number, returns its positive square root.

Syntax
@Sqrt( number )

Parameters
number

Number. The number whose square root you want to find. The number must be positive, otherwise @Sqrt returns an error.

Language cross-reference
Sqr function of LotusScript language

Examples: @Sqrt
This example returns 4.
@sqr( 16 )
@StatusBar
Writers a message or messages to the status bar.

Note: This @function is new with Release 6.

Syntax
@StatusBar( statusBarText )

Return value
statusBarText
Text or text list. The text of the status bar message. A list produces one message per element.

Usage
This @function works only in the Notes client.

Examples: @StatusBar
1. This onLoad/Postopen event writes a message to the status bar.
   @StatusBar("Loaded "Form A" in "" + @Subset(@DbName; -1) + "")
2. This onLoad/Postopen event writes two messages to the status bar.
   @StatusBar("Loaded "Form A";
   ("Database is " + @Subset(@DbName; -1) + "")

@Subset
Searches a list from left to right and returns the number values you specify. If you specify a negative number, @Subset searches the list from right to left, but the result is ordered as from the beginning of the list.

Syntax
@Subset( list ; number )

Parameters
list
Text list, number list, or time-date list. The list whose subset you want.

number
Number. The number of values from list that you want. Specifying zero (0) returns the error, "The second argument to @Subset must not be zero."

Return value
subsetList
Text list, number list, or time-date list. The list, containing the number of values you specified.

Examples: @Subset
1. This example returns New Orleans;London.
   @Subset("New Orleans":"London":"Frankfurt":"Tokyo";2)
2. This example returns London;Frankfurt;Tokyo.
   @Subset("New Orleans":"London":"Frankfurt":"Tokyo":-3)

3. This example returns New Orleans;London;Frankfurt if the field named BranchOffices is made up of
   @Subset(BranchOffices;3)

---

@Success

Returns 1 (True). Use this function with @If in field validation formulas to indicate that the value entered
satisfies the validation criteria.

**Syntax**

@Success

**Return value**

true

Number. The number 1, meaning True.

**Usage**

Use @Success in input validation formulas for editable fields.

**Examples: @Success**

This example returns 1 and allows the document to be saved when the value in the field Price is less
than 100. This indicates that acceptable data was entered when used in an input validation formula.
@If(Price<100;@Success;@Failure("Price too large"))

---

@Sum

Adds a set of numbers or number lists.

**Syntax**

@Sum( numbers )

**Parameters**

numbers

Numbers or number lists. As many numbers or number lists as you want to sum.

**Return value**

result

Number. The sum of all the numbers, including members of number lists.

**Usage**

Make sure the fields you send as parameters contain a number value -- Notes/Domino interprets empty
number fields as the null string.

Negative numbers in lists must be enclosed in parentheses.
**Language cross-reference**

Addition operator of LotusScript language

**Examples: @Sum**

1. This example returns 3.
   \[ @Sum( 1 \text{ : } 2 ) \]
2. This example returns 11.
   \[ @Sum( -1 \text{ : } 2 \text{ ; } -10 \text{ : } 20 ) \]
3. This example sets the Total field to 50 if numPersons is a number field containing 5; 10; 15; 20.
   \[ @SetField("Total";@Sum(numPersons)) \]
4. This example looks at the Transactions view in the current database, whose first column contains number values indicating the amount of a transaction. The formula sums the transactions and places the total in the result field on the current document.
   \[
   \text{FIELD result:=result;}
   \text{r:=@DbColumn("";"";"";"Transactions";1 );}
   \text{@SetField("result";@Sum( r ))}
   \]
5. This example displays a view in a dialog box. The first column in the view contains a product name, the second contains its price. After the user selects one or more products in the dialog box, the formula displays the total cost of the selected items.
   \[
   \text{amounts:=@PickList([Custom];@DbName ;"Products";"Choose products";"Please select the products you want to order";2 );}
   \text{total:=@Sum( @TextToNumber( amounts ) );}
   \text{@Prompt([Ok];"Total";"The total cost of these products is "+@Text(total))}
   \]

**@Tan**

Given an angle in radians, returns the tangent of the angle. In a right triangle the tangent of an acute angle is the ratio of the length of the opposite side to the length of the adjacent side.

**Syntax**

\[ @Tan(\text{angle}) \]

**Parameters**

\[ \text{angle} \]

Number. Any angle, expressed in radians.

**Return value**

\[ \text{tangent} \]

Number. The tangent of \text{angle}.

**Language cross-reference**

Tan function in LotusScript language

**Examples: @Tan**

This example returns 1.
\[ @Tan( \text{@Pi/4} ) \]
@TemplateVersion

Returns the version number of the master template upon which the database design is based.

Note: This @function is new with Release 6.5.

Syntax
@TemplateVersion

Return value
versionNumber

Text. The version number, "6.0.3" as an example.

Usage
This @function applies when "Inherit design from master template" is checked on the Design tab of the Database Properties. The template version number can be seen there.

You cannot use this function in Web applications.

@Text

Converts any value to a text string.

Syntax
@Text( value ; format–string )

Parameters
value

Number, time-date, rich text, or text. The value you want to convert to text.

Note: Conversion of rich text is new with Release 6.

format-string

Text. Optional. Up to four format-strings (see table below). These determine how the text is returned. If the value is already a text data type, the format–string is ignored.

Return value
textValue

Text. The value you specified, converted to text. If you used any format-strings, they are applied.

@Text with time–date components

There are four separate categories of time–date, format–string components. You can include up to four components, but only one from each category.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>Month, day and year</td>
</tr>
<tr>
<td>D1</td>
<td>Month and day, year if it is not the current year</td>
</tr>
<tr>
<td>Symbol</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>D2</td>
<td>Month and day</td>
</tr>
<tr>
<td>D3</td>
<td>Month and year</td>
</tr>
<tr>
<td>T0</td>
<td>Hour, minute, and second</td>
</tr>
<tr>
<td>T1</td>
<td>Hour and minute</td>
</tr>
<tr>
<td>Z0</td>
<td>Always convert time to this zone</td>
</tr>
<tr>
<td>Z1</td>
<td>Display zone only when it is not this zone</td>
</tr>
<tr>
<td>Z2</td>
<td>Display zone always</td>
</tr>
<tr>
<td>S0</td>
<td>Date only</td>
</tr>
<tr>
<td>S1</td>
<td>Time only</td>
</tr>
<tr>
<td>S2</td>
<td>Date and time</td>
</tr>
<tr>
<td>S3</td>
<td>Date, time, Today, or Yesterday</td>
</tr>
<tr>
<td>Sx</td>
<td>Use when you cannot predict the exact format of the value being passed, but you know that it is either a time, a date, or both.</td>
</tr>
</tbody>
</table>

@Text with number values

For number values, compose a format–string by combining any of the following components into a string.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>General format (significant digits only)</td>
</tr>
<tr>
<td>F</td>
<td>Fixed format (set number of decimal places)</td>
</tr>
<tr>
<td>S</td>
<td>Scientific format (E notation)</td>
</tr>
<tr>
<td>C</td>
<td>Currency format (two decimal places)</td>
</tr>
<tr>
<td>,</td>
<td>Punctuated at thousands (using U.S. format)</td>
</tr>
<tr>
<td>%</td>
<td>Percentage format</td>
</tr>
<tr>
<td>()</td>
<td>Parentheses around negative numbers</td>
</tr>
<tr>
<td>number</td>
<td>Number of digits of precision</td>
</tr>
</tbody>
</table>

Usage

Once a number value is converted to text, you will not be able to use the number for arithmetic calculations.

Rich text conversion loses attachments and all formatting except tabs and spaces. When rich text is converted in a document, the document must be saved before the conversion becomes visible.

Rich text conversion does not work in column formulas. Use @Abstract to convert the contents of a rich text field to plain text. Then reference the plain text field in the view. For example, if you add the following code to a hidden computed field called plainText, you can then set the default value of the view column to "plainText" to display the contents of the RTField:

```
@Abstract([TextOnly];15360;"";"RTField")
```

Use caution if using @Text to convert numbers or dates in a column. In databases hosted by a server, the numbers and dates always display using the format settings of the hosting server's operating system. Also, if the date or number format settings of either the client accessing the database or server hosting the database change, you may need to entirely rebuild the view.
Language cross-reference
CStr function of LotusScript language

GetFormattedText method in LotusScript NotesRichTextItem class

getFormattedText method in Java RichTextItem class

Examples: @Text
1. This example returns 123.45.
   @Text(123.45)
2. This example returns $800.00 if the value in the Sales field is 800.
   @Text(Sales;"C,2")
3. This example returns 8.00E+02.
   @Text(800;"S")
4. This example returns 04/11/93 10:43 AM.
   @Text(@Now)
5. This example returns 04/11.
   @Text(@Now;"D1S0")
6. This example returns 10:43:30 AM.
   @Text(@Now;"D1S1")
7. This example returns 04/93 10:43 AM.
   @Text(@Now;"D3T1")
8. This example returns the rich-text Body field stripped of attachments and formatting.
   @Text(Body)
9. To convert a number date (in the ShipDate field) into a written date, you can use the following code.
    If ShipDate contains [08/31/2002], the result is "August 31, 2002."
    @If( @IsTime(ShipDate); @Text(@Select(@Month(ShipDate); "January"; "February"; "March"; "April"; "May"; "June"; "July"; "August"; "September"; "October"; "November"; "December") + " " + @Text(@Day(ShipDate)) + ", " + @Text(@Year(ShipDate)); "No date given")

@TextToNumber
Converts a text string to a number, where possible.

Syntax
@TextToNumber( string )

Parameters

string

Text. The string you want to convert to a number. If the string contains both numbers and letters, it must begin with a number to be converted properly. For example, the string "12ABC" converts to 12, but "ABC12" produces an error.
Return value

number

Number. The string, converted to a number.

Usage

This function is useful for converting a number in a text field to a number that can be used for computation in a number field.

You can't use @TextToNumber to convert special text (such as that returned by @DocChildren or @DocDescendants) to a number.

@TextToNumber returns an error if you try to pass anything besides a string into it.

Language cross-reference

Val function of LotusScript language

Str function of LotusScript language

CInt function of LotusScript language

CLng function of LotusScript language

CSng function of LotusScript language

CDbl function of LotusScript language

Examples: @TextToNumber

1. This example returns 123 as a number.
   @TextToNumber("123")

2. This example returns @ERROR if the contents of the field named Cost cannot be converted to a number.
   @TextToNumber(Cost)

@TextToTime

Converts a text string to a time–date value, where possible.

Syntax

@TextToTime( string )

Parameters

string

Text. The string you want to convert to a time-date.

Return value

time-date

Time-date or time-date range. The string, converted to a time-date.
Usage
This function is useful for converting a date within a text field to a value that can be used for computation in a time–date field.

"Today", "Tomorrow", and "Yesterday" are the only legal strings to use to represent relative dates. The formula @TextToTime("Next week") returns a blank because the text string "Next week" cannot be converted to a time–date value.

@TextToTime returns an error if you try to pass anything besides a string into it, including a time-date value.

Language cross-reference
DateValue function of LotusScript language
TimeValue function of LotusScript language
CDat function of LotusScript language

Examples: @TextToTime
1. This example returns 8/10/90 2:40:00 AM.
   @TextToTime("8/10/90 2:40")
2. This example returns Today.
   @TextToTime("Today")
3. This example sets the value of the result field (a time-date field that allows multiple values) to the date range 04/16/96 - 08/18/96.
   FIELD result:=result;
   @SetField("result"; @TextToTime("04/16/96-08/18/96"))

@ThisName
Returns the name of the current field.

Note: This @function is new with Release 6.

Syntax
@ThisName

Return value
name

Text. The name of the current field.

Usage
This @function returns null outside a field formula.

This @function is useful in writing portable code. Use @ThisName to construct references to other fields (for example, in @GetField) that have similar names.

Language cross-reference
CurrentField property of LotusScript NotesUIDocument class
Examples: @ThisName
1. Assume a form has fields named Total_1, Quantity_1, Cost_1, Total_2, Quantity_2, Cost_2, and so on. The Total fields are computed using the following formula for the value. The same formula can be used in every Total field.

\[
\text{Suffix} := \text{Right}(\text{ThisName}; " "); \\
\text{QuantityFld} := "\text{Quantity_}" + \text{Suffix}; \\
\text{CostFld} := "\text{Cost_}" + \text{Suffix}; \\
\text{GetField}(\text{QuantityFld}) \times \text{GetField}(\text{CostFld})
\]

2. This formula makes it easier for a designer to check code on a form that may have several debugging @Prompt functions in several different fields because it identifies which field value is being displayed:

\[
\text{result} := \text{Round}(2.66); \\
\text{Prompt}([\text{Ok}]; \text{ThisName}; \text{Text(result)}); \\
\text{result} + 2
\]

This example returns 3 in the "roundNumber" dialog box and 5 in the roundNumber field when the form displays.

@ThisValue
Returns the value of the current field.

Note: This @function is new with Release 6.

Syntax
@ThisValue

Return value
\(value\)

The value of the current field.

Usage
This @function returns null outside a field formula.

This @function is useful in writing portable code. Use @ThisValue instead of the name of the current field.

Language cross-reference
FieldGetText method of LotusScript NotesUIDocument class

Examples: @ThisValue
1. This translation formula replaces all spaces with underscores in the current field.

\[
@\text{ReplaceSubstring}(\text{ThisValue}; " "; ";")
\]

2. This input validation formula for a listbox field checks whether the user selected more than one list option and asks them to if they have not:

\[
@\text{If}(\text{ThisValue} \neq "") \& (\text{Elements}(\text{ThisValue}) = 1); \text{Failure}(\text{You must select more than one choice}); \text{Success}
\]

@Time
Translates numbers for the various components of time and date; then returns the time–date value.
Syntax

@Time( hour; minute; second ) @Time( year ; month ; day ; hour ; minute ; second ) @Time( time-date )

Parameters

year
Number. The year.

month
Number. The month.

day
Number. The day.

hour
Number. The number of hours you want to appear in the resulting time.

minute
Number. The number of minutes you want to appear in the resulting time.

second
Number. The number of seconds you want to appear in the resulting time.

time-date
Time-date. For a time–date value such as @Now or [10/31/93 12:00:00], @Time removes the date portion of the value, leaving only the time.

Return value

truncatedTimeDate
Time-date. The time corresponding to the parameters you sent to @Time, minus any date components if the parameter is time-date.

Language cross-reference

CDat function of LotusScript language

DateNumber function of LotusScript language

Examples: @Time

1. This example returns 4/11/51 11:50:30 PM.
   @Time(1951;04;11;23;50;30)
2. This example returns 09:19:24 AM at 9:19:24 A.M on any day.
   @Time(@Now)
3. This example returns 09:19:24 AM if 9:19:24 A.M is the time the document was created.
   @Time(@Created)
@TimeMerge
Builds a time-date value from separate date, time, and time zone values.

Note: This @function is new with Release 6.

Syntax

@TimeMerge ( date ; time ; timeZone )

Parameters

date
Time-date value. The date you want to include in the new date-time value.

time
Time-date value. The time you want to include in the new date-time value.

timeZone
String. Optional. The canonical time zone value you want to apply to the new date-time value. You can use a Time zone field to create this value.

Return value

Time-date

A new time-date value made up of the date, time, and zone supplied as function parameters.

Examples: @TimeMerge

1. This code, when added to a hostpot button, displays 02/23/2002 05:45:00 AM in the Merged date dialog box if the field date contains 02/23/02 and the field time contains 17:45:00.

@Prompt([OK];"Merged date";@Text(@TimeMerge(date;time)))

2. This code, when added to a form action, displays 02/23/2002 05:45:00 AM in the Merged date dialog box if the field date contains 02/23/02 02:30:00 and the field time contains 03/23/03 05:45:00.

@Prompt([OK];"Merged date";@Text(@TimeMerge(date;time)))

3. This code, when added to a hotspot button, displays 07/04/2002 08:30:00 PM in the Merged date dialog box if the field date contains 07/04/02, the field time contains 13:30:00, and the field zone contains Z=11$DO=0$ZX=1$ZN=Samoa (which displays as GMT-11:00). The hour is adjusted to reflect the specified time zone.

@Prompt([OK];"Merged date";@Text(@TimeMerge(date;time;zone)))

@TimeToTextInZone

Converts a time-date value to a text string, incorporating time zone information.

Syntax

@TimeToTextInZone ( timeDate ; timeZone ; formatString )

Parameters

timeDate
Time-date value.
timeZone

Canonical time zone value. You can derive a time zone value using a Lotus Notes Time zone field.

formatString

Optional. String consisting of one or more of the following format specifiers:

<table>
<thead>
<tr>
<th>Format specifier</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>Year, month, and day</td>
</tr>
<tr>
<td>D1</td>
<td>Month and day, year if it is not the current year</td>
</tr>
<tr>
<td>D2</td>
<td>Month and day</td>
</tr>
<tr>
<td>D3</td>
<td>Month and year</td>
</tr>
<tr>
<td>T0</td>
<td>Hour, minute, and second</td>
</tr>
<tr>
<td>T1</td>
<td>Hour and minute</td>
</tr>
<tr>
<td>S0</td>
<td>Date only</td>
</tr>
<tr>
<td>S1</td>
<td>Time only</td>
</tr>
<tr>
<td>S2</td>
<td>Date and time</td>
</tr>
<tr>
<td>S3</td>
<td>Date, time, Today, or Yesterday</td>
</tr>
<tr>
<td>Sx</td>
<td>Use when you cannot predict the exact format of the value being passed, but you know that it is either a time, a date, or both.</td>
</tr>
</tbody>
</table>

You can include up to three specifiers, but only one that begins with D, one that begins with T, and one that begins with S.

Return value

string

The time-date value converted to a string.

Examples: @TimeToTextInZone

1. This code, when used in an action button on a form, applies the zone information of GMT-00:00 that a user selects from the list in the "There" Time zone field to the time-date of 02/26/2002 03:19 PM EST that results from an @Now formula in the "Here" Date/Time field. The "Time there" message box that appears displays "02/26/2002 08:19:00 PM."
   @Prompt([OK];"Time there";@TimeToTextInZone(Here;There))
2. This code, when added as a Column Value formula, displays "11:06 AM Today" in the view column if the Here field contains "02/26/2002 03:06 PM EST," the There field contains Z=9$DO=1$DL=4 1 1 10-1 1$ZX=3$ZN=Alaskan (which displays as GMT -09:00 Alaska), and the current date is 02/26/2002.
   @TimeToTextInZone(Here;There;"D2T1S3")

@TimeZoneToText

Converts a canonical time zone value to a human-readable text string.

Syntax

@TimeZoneToText( timeZone ; formatString )
Parameters

**timeZone**

Canonical time zone value. Use a Lotus Notes Time zone field to create a time zone value.

**formatString**

Optional. String consisting of one or more of the following format specifiers:

<table>
<thead>
<tr>
<th>Format specifier</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Short time zone string, for example:</td>
</tr>
<tr>
<td></td>
<td>&quot;GMT-08:00&quot;</td>
</tr>
<tr>
<td>A</td>
<td>Alias for local time zone. For example, if the zone is the same as the zone in which the system is running, returns:</td>
</tr>
<tr>
<td></td>
<td>&quot;Local time&quot;</td>
</tr>
</tbody>
</table>

Return value

**string**

The time-date value converted to a string. If you do not include a format specifier, a long time zone label is returned. For example:

"(GMT-08:00) Pacific Time (US & Canada);Tijuana"

Usage

This function is useful for displaying the contents of a Time zone field in a view. If you do not use this function, a Time zone field value displays in the view with a format similar to the following:

Z=9$DO=1$DL=4 1 1 10-1 1$ZX=1$ZN=Alaskan

Also use this function with the [@GetCurrentTimeZone](#) function to translate the time zone value it returns into a readable string.

Examples: [@TimeZoneToText](#)

1. This code, when added as the Column Value formula for a view, displays the contents of the Time zone field named Zone as "GMT-07:00" if the Zone field has the value Z=7$DO=0$ZX=6$ZN=US Mountain (which is selected in the Time zone field as GMT-07:00 Arizona).
   
   ```
   @TimeZoneToText(Zone;"S")
   ```

2. This code, when added as the Column Value formula for a view and accessed from a system running in the EST time zone, displays a document that has (GMT 00:00) Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London selected in its "Zone" Time zone field as "" and a document that has (GMT-05:00) Eastern Time (US & Canada) selected in its Zone field as "Local time."
   
   ```
   @TimeZoneToText(Zone;"SA")
   ```

[@Today](#)

Returns today’s date.

Syntax

[@Today](#)
**Return value**

*today*

Time-date. Today’s date.

**Usage**

This function is identical to the formula @Date(@Now). It is usually used in default value formulas to automatically enter the current date.

Using @Today in column or selection formulas may impact the efficiency of your application. It also causes the view refresh indicator to display constantly.

In a field formula, Lotus Notes/Domino takes the value for @Today from the client computer’s clock.

**Language cross-reference**

Now function of LotusScript language

Today function of LotusScript language

SetNow method of LotusScript NotesDateTime class

Today property of LotusScript NotesInternational class

setNow method of Java DateTime class

Today property of Java International class

**Examples:** @Today

1. This example returns 02/19/93 if today is February 19, 1993.
   
   @Today

2. This example sets the field named ReceivedDate to today’s date.

   FIELD ReceivedDate:=@Today

---

**@Tomorrow**

Returns the time–date value that corresponds to tomorrow’s date.

**Syntax**

@Tomorrow

**Return value**

*tomorrow*

Time-date. Tomorrow’s date.

**Usage**

Using @Tomorrow in column or selection formulas may impact the efficiency of your application. It also causes the view refresh indicator to display constantly.

In a field formula, Notes/Domino takes the value for @Tomorrow from the clock in the client computer.
Language cross-reference
Tomorrow property of LotusScript NotesInternational class

Tomorrow property of Java International class

Examples: @Tomorrow
1. This example returns 4/26/93 if today is April 25, 1993.
    @Tomorrow
2. This example sets the field named AnswerBack to tomorrow’s date.
    FIELD AnswerBack:=@Tomorrow

@ToNumber
Converts a value with a data type of text or number to a number value.

Note: This @function is new with Release 6.

Syntax
@ToNumber( value )

Parameters
value

Text or number. A value having any other data type returns the error, "The value cannot be converted to a Number."

Return value
number

The value converted to a number.

Usage
This function is useful for ensuring that a value has a number data type before using it in functions that require numbers as parameters.

Language cross-reference
Val function of LotusScript language
Str function of LotusScript language
CInt function of LotusScript language
CLng function of LotusScript language
CSng function of LotusScript language
CDbl function of LotusScript language
Examples: @ToNumber
This example converts the values in a text field, containing "20," and a number field containing 10, into numbers so that they can be added using the @Sum function, which requires two numbers. The formula returns 30.
@Sum(@ToNumber(numberField);@ToNumber(textField))

@ToTime
Converts a value with a data type of text or time to a date-time value.

Note: This @function is new with Release 6.

Syntax
@ToTime( value )

Parameters
value
Text or time. A value having any other data type returns the error, "The value cannot be converted to a Number."

Return value
time
The value converted to a time value.

Usage
This function is useful for ensuring that a value has a time data type before using it in functions that require time values as parameters.

Language cross-reference
DateValue function of LotusScript language

TimeValue function of LotusScript language

CDat function of LotusScript language

Examples: @ToTime
1. This code, when added to a field, converts the text value in the "date" field containing "08/29/01" into a time value and adds two days to the date. This function returns 08/31/01.
   @Adjust(@ToTime(date);0;0;2;0;0;0)
2. This example, when added to an action button, displays the date two days after the date selected by a user in the "request" date-time field.
   @Prompt([Ok];"Delivery";@Text(@Adjust(@ToTime(holiday);0;0;2;0;0;0)))

@Transform
Applies a formula to each element of a list and returns the results in a list.

Note: This @function is new with Release 6.
**Syntax**

\[ @\text{Transform}( \text{list} ; \text{variableName} ; \text{formula} ) \]

**Parameters**

\text{list}

Text, number, or time-date list. The list to be acted upon.

\text{variableName}

Text. The name of a variable. Use this variable in the formula to refer to the list element being acted upon.

\text{formula}

Valid formula that evaluates to a result. The remainder of @Transform after the second parameter is the formula that is applied to each element of the input list. The formula must return a value.

**Return value**

\text{list}

Text, number, or time-date. The result of the transformation on the input list. The first value returned by the formula determines the data type of the list. Subsequent return values must be of the same type.

**Usage**

An iteration of the formula can return a list, which adds multiple values to the return list.

@Transform returns an error if any iteration of the formula returns an error.

If an iteration of the formula returns @Nothing, no element is added to the return list.

**Language cross-reference**

ForAll statement of LotusScript language

**Examples: @Transform**

The following examples are translation formulas that transform the elements of the numeric multi-value field OriginalList. Assume that OriginalList contains the values 4, -4, and 16.

1. This formula returns a 3-element list whose values are 2, -2, and 4.
   \[ \text{@Transform(OriginalList; } "x"; \text{ @If(x } \geq \text{ 0; } \text{ @Sqrt(x); -@Sqrt(@Abs(x)))} \]

2. This formula returns the same as above. However, if OriginalList is null, this formula returns null rather than an error.
   \[ \text{@If(OriginalList } = \text{ @Nothing; @Nothing; } @\text{Transform(OriginalList; } "x"; \text{ @If(x } \geq \text{ 0; } \text{ @Sqrt(x); -@Sqrt(@Abs(x)))}) \]

3. This formula returns a 2-element list whose values are 2 and 4.
   \[ @\text{If(OriginalList } = \text{ @Nothing; @Nothing; } @\text{Transform(OriginalList; } "x"; \text{ @If(x } \geq \text{ 0; } \text{ @Sqrt(x); @Nothing}))}\]

4. This formula, when used in a hotspot button creates a field called originalCorrected that adds an asterisk to the beginning of each element in the "original" text list if it does not already have one.
   \[ \text{FIELD originalCorrected := @Transform(original; } "\text{var}"; \]
@Trim

Removes leading, trailing, and redundant spaces from a text string, or from each element of a text list.

Syntax

@Trim(string)

Parameters

string

Text or text list.

Return value

trimmedString

Text or text list. The string, with extra spaces removed.

Usage

If a text string is all spaces, @Trim returns an empty string (length of 0). If an element of a text list is all spaces, @Trim removes the element. If all elements of a text list are all spaces, @Trim returns an empty string.

Language cross-reference

Trim function of LotusScript language

Examples: @Trim

1. This example returns ROBERT SMITH.
   @Trim(@UpperCase("Robert Smith "))
2. This example returns ROBERT SMITH.
   @UpperCase(@Trim(" Robert  Smith"))
3. This example returns Just a quick reminder, if the original Topic field is "Just a quick reminder."
   @Trim(Topic)
4. This example returns Seattle;Toronto;Santiago;USA;Canada;Chile if the list of values contained in the City field consists of Seattle, Toronto, Santiago; the StateOrProvince field contains no values; and the Country field contains the list of values USA, Canada, Chile.
   @Trim(City:StateOrProvince:Country)
5. This example returns 45 if the content of the field Date is 8/29/89 16:30:45.
   @Trim(@Text(@Second(Date)))

@True

Returns the number 1. This function is equivalent to @Yes.

Syntax

@True
Return value

true

Number. The number 1.

Language cross-reference

Built-in constants of LotusScript language

Examples: @True

1. This example returns 1.
   @True
2. This example returns 1 if the value in the Dept field is greater than 100.
   @If(Dept>100;@True;@False)

@Unavailable

Deletes the value of an editable field.

Syntax

FIELD fieldName := @Unavailable

Usage

This function works in agent, view action, and toolbar button formulas.

If the field has a default value, the default value is reinstated after this function deletes the current value.

This function is the same as @DeleteField

Do not use this function to test to see if a field is unavailable. Use @IsUnavailable instead.

Language cross-reference

RemoveItem method of LotusScript NotesDocument class

Remove method of LotusScript NotesItem class

FieldClear method of LotusScript NotesUIDocument class

Clear method of LotusScript NotesUIDocument class

removeItem method of Java Document class

remove method of Java Item class

Examples: @Unavailable

This formula creates a field named NewDate and sets it to today’s date, then removes the field named OldDate from the document.

FIELD NewDate:=@Today
FIELD OldDate:=@Unavailable;
@UndeleteDocument
In a database with "Allow soft deletions" selected, this command restores a deleted document.

Note: This @function is new with Release 5.

Syntax
@UndeleteDocument

Usage
This @function can be used in toolbar button, hotspot, action, and agent formulas.

To allow soft -- that is, delayed -- deletions, go to the Advanced tab of database properties, check "Allow soft deletions," and specify an integer value for "Soft delete expire time in hours." Soft-deleted documents appear to be deleted but are held in the database for the specified number of hours before actual deletion.

To see the soft-deleted documents, create a view of type "Shared, contains deleted documents." To restore a soft-deleted document, run @UndeleteDocument on it before the "Soft delete expire time in hours" expires.

Examples: @UndeleteDocument
This is the formula for an action in a view of type "Shared, contains deleted documents." The user can go to this view, see the documents that are soft-deleted, and run this action on selected documents to restore them. The database must "Allow soft deletions" and specify "Soft delete expire time in hours."

@Unique

Without a parameter, returns a random, unique text value. With a parameter, removes duplicate values from a text list by returning only the first occurrence of each member of the list.

Syntax
@Unique @Unique( textlist )

Parameters
textlist

Text list. Any text list.

Return value
Without a parameter:

uniqueValue

Text. A random, unique text value.

With a parameter:

uniqueList

Text list. The text list, with duplicate values removed.
Usage
@Unique is case-sensitive.

Language cross-reference
ArrayUnique function of LotusScript language

Examples: @Unique
1. This example returns red; green; blue.
   @Unique("red":"green":"blue":"green":"red")
2. This example returns red; green; blue; Green.
   @Unique("red":"green":"blue":"Green":"red")

@UpdateFormulaContext
Updates the context of a formula to the Notes client window currently being accessed by the code. For example, if the code accesses a new form called "Response" by using @Command([Compose];"Response", @UpdateFormulaContext switches the context of the formula to this new form. Any subsequent functions in the code execute in the context of the Response document, not the current document.

Note: This function in new with Release 6.

Syntax
@UpdateFormulaContext

Usage
You can use @UpdateFormulaContext to extract values from or set values in external documents. You can even access document- and database-specific information using functions such as @DbName, @DbTitle, @Created, @DocumentUniqueID, @GetDocField, @GetField, @GetProfileDocument.

This function is only valid in the Notes client; it is not supported in Web applications. @UpdateFormulaContext is only valid in formulas that interact with the user, such as in agents that have no target documents, events, toolbar buttons, hotspot buttons, and actions. It does not work in formulas in which commands cannot be used.

Examples: @UpdateFormulaContext
1. The following code, when used in a view action, creates a response document to the currently selected document then populates its fname and lname fields with the values of the fname and lname fields in the current document:
   tempfname := fname;
   templname := lname;
   @Command([Compose];"Response");
   @UpdateFormulaContext;
   FIELD fname := tempfname;
   FIELD lname := templname
2. The following code, when used in a view action that contains documents that have the fields "CreatedDate," which displays the document's creation date and "nextCreated," an editable text field, opens the previous document in the view and adds the creation date of the current document into its "nextCreated" field:
   tempDate := @GetDocField(@DocumentUniqueID;"CreatedDate");
   @Command([NavPrev]);
@UpperCase
Converts the lowercase letters in the specified string to uppercase.

Syntax
@UpperCase(string)

Parameters
string
Text. The string you want to convert to uppercase.

Return value
uppercaseString
Text. The string converted to uppercase letters.

Usage
This function is useful when you want to search for a particular value and cannot predict whether it will appear in lowercase, uppercase, or a combination of the two. You can also use it as an input translation formula to convert a field’s contents to uppercase.

Language cross-reference
UCase function of LotusScript language
StrConv function of LotusScript language

Examples: @UpperCase
1. This example returns ROBERT T. SMITH.
   @UpperCase("Robert T. Smith")
2. This example returns MA if the State field contains "ma," "Ma," or "MA."
   @UpperCase(State)
3. This example returns FLETCHER if "William Fletcher" is the name associated with the current UserID. @UpperCase is used in conjunction with @Right to find and convert only the user’s last name.
   @UpperCase(@Right(@UserName;" "))
   If the user id is a hierarchical id, the following code returns FLETCHER:
   @UpperCase(@Right(@Name([CN]; @UserName); " "))

@URLDecode
Decodes a URL string into regular text.

Note: This function is new with Release 6.

Syntax
@URLDecode(decodeType; token)
Parameters

decodeType

Text. The type of encoding you want to use to translate the token. You can specify either a string argument or a MIME character set.

String arguments:
- "Domino" -- Decodes the token using the standard character set used by the Lotus Domino Web server. This keyword is equivalent to the "UTF-8" MIME character set.
- "Platform" -- Decodes the token using the current system’s native character set.

MIME character set:

Decodes the hexadecimal digits that represent the code value into octets, then converts the specified character sets into LBMCS. The supported MIME character sets are:
- "ISO-8859-1" -- The ISO’s (International Standards Organization) 8-bit, single-byte-coded graphic character set for European languages.
- "Shift_JIS" -- The character set for the Japanese language.

Example:

```
@URLDecode("Domino";encode)
```

@URLEncode

Encodes a string into a URL-safe format.

Syntax

```
@URLEncode( encodingFormat ; token )
```

Parameters

encodingFormat

Text. The type of encoding you want to use to translate the token. You can specify either a string argument or a MIME character set.

String arguments:
- "Domino" -- Encodes the token in the standard character set used by the Lotus Domino Web server. This keyword is equivalent to the "UTF-8" MIME character set.

Examples: @URLDecode

1. This code, when used as the default value for a field, decodes the URL-formatted string in the encode field. It returns "Employee/My Database" if the encode field contains "Employee%2FMy%20Database.nsf."

```
@URLDecode("Domino";encode)
```

Chapter 6. Formula Language @Functions A-Z  445
"Platform" -- Encodes the token using the current system's native character set.

MIME character set:

Converts non-ASCII characters into the specified character set and encodes the characters into %XX format, where XX is a hexadecimal digit representing the encoded value. Some examples include:

- "ISO-8859-1" -- The ISO's (International Standards Organization) 8-bit, single-byte-coded graphic character set for European languages.
- "Shift_JIS" -- The character set for the Japanese language.

**token**

Text or text list. URL string(s) to be encoded.

**Return value**

*encodedURLString*

Text or text list. Returns the URL string(s) encoded in the specified encoding format.

**Usage**

Do not use @URLEncode to encode an entire URL string. For example, @URLEncode("Domino","http://www.ibm.com/") returns "http%3A%2Fwww.ibm.com%2F", which would not link successfully to the desired website.

**Examples: @URLEncode**

1. This formula returns "By%20Date" as the encoded URL.
   
   @URLEncode("Domino","By Date")

2. This formula returns "Support%20%E0%20la%20client%E8le" as the encoded URL.
   
   @URLEncode("ISO-8859-1","Support à la clientèle")

3. This formula returns "Support%20%C3A0%20la%20client%C3%A8le" as the encoded URL.
   
   @URLEncode("UTF-8","Support à la clientèle")

**@URLGetHeader**

Returns specific Hypertext Transfer Protocol (HTTP) header information from the Uniform Resource Locator (URL). A URL is a text string used for identifying and addressing a Web page.

**Syntax**

@URLGetHeader( urlstring; headerstring; webusername; webpassword; proxywebusername; proxywebpassword )

**Parameters**

*urlstring*

Text. The URL for the Web page you want to open, for example, http://www.acme.com/.

*headerstring*

Enter a header string to return the desired URL header value. The acceptable header strings are documented in the HTTP specification (available at locations on the Internet, such as http://www.w3.org/) and are subject to change based on updated versions of the specification.
webusername$

Text. Optional. Some Internet servers require you to obtain a user name and password before you can access their pages. This parameter allows you to enter the user name that you previously obtained from the authenticated Internet server.

webpassword$

Text. Optional. Some Internet servers require you to obtain a user name and password before you can access their pages. This parameter allows you to enter the password that you previously obtained from the authenticated Internet server.

proxyusername$

Text. Optional. Some proxy servers require that you specify a user name in order to connect through them. This parameter allows you to enter the user name for the proxy server. See your administrator for the username required by the proxy.

proxypassword$

Text. Optional. Some proxy servers require that you specify a password in order to connect through them. This parameter allows you to enter the user name for the proxy server. See your administrator for the password required by the proxy.

Return value

headervaluestring

Text. Returns the header value that you requested. If a null value is returned, the header value that you requested was not found in the header of the Web page.

Usage

The @URLGetHeader function should only be used in the context of either the Server Web Navigator or Personal Web Navigator database.

Examples: @URLGetHeader

1. This example returns the last date that the www.acme.com Web page was modified.
   @URLGetHeader ("http://www.acme.com/"; "Last-modified")

2. This example returns the name of the Web server software where the www.acme.com Web page resides.
   @URLGetHeader ("http://www.acme.com/"; "Server")

@URLHistory

Used for navigating, saving, and reloading a Uniform Resource Locator (URL) history list. The URL history list keeps track of all the Web pages you have visited. The history list is used for the Next and Previous buttons and for the Web Tours.

Syntax

@URLHistory( [ command ] )
Parameters
[ command ]

Keyword. The name of the @URLHistory command you want to use:
- [NEXT]
  Moves to the next URL in the history list.
- [PREV]
  Moves to the previous URL in the history list.
- [RELOAD]
  Reloads the current history list from the Web Tour document.
- [SHOW]
  Displays the History dialog box.
- [SAVE]
  Saves the history list into a new Web Tour document, which a user can reload later to follow that history.

Usage
The @URLHistory function works from the Notes/Domino workstation only and should only be used with either the Server Web Navigator or Personal Web Navigator database.

Examples: @URLHistory
Below are examples of each command you want to specify.

[NEXT]
This example moves to the next URL in the history list.
@URLHistory([NEXT])

[PREV]
This example moves to the previous URL in the history list.
@URLHistory([PREV])

[SHOW]
This example displays the History dialog box.
@URLHistory([SHOW])

[SAVE]
This example saves the history list into a new Web Tour document that a user can reload later to follow that history.
@URLHistory([SAVE])

[RELOAD]
This example reloads the history list from the Web Tour document.
@URLHistory([RELOAD])

@URLOpen
Retrieves a World Wide Web page specified by its URL.
Syntax

@URLOpen

@URLOpen( urlstring )

@URLOpen( urlstring ; [ reloadflag ] )

@URLOpen( urlstring ; [URLLIST] )

@URLOpen( urlstring ; [ reloadflag ];[URLLIST] )

@URLOpen( urlstring ; [ reloadflag ];[URLLIST]; charset )

@URLOpen( urlstring ; [ reloadflag ];[URLLIST]; charset ; webusername )

@URLOpen( urlstring ; [ reloadflag ];[URLLIST]; charset ; webusername ; webpassword )

@URLOpen( urlstring ; [ reloadflag ];[URLLIST]; charset ; webusername ; webpassword ; proxywebusername )

@URLOpen( urlstring ; [ reloadflag ];[URLLIST]; charset ; webusername ; webpassword ; proxywebusername ; proxywebpassword )

Parameters

urlstring

Text. Optional. The URL for the Web page you want to open, for example, http://www.acme.com/. This parameter may also include comma-separated arguments to be passed by Domino to the javascript window.open command.

[ reloadflag ]

Keyword. Optional.

RELOAD. Reloads the page from its Internet server.

RELOADIFMODIFIED. Reloads the page only if it has been modified on its Internet server.

[URLLIST]

Keyword. Optional. Web pages can contain URL links to other Web pages. This keyword specifies that the Web Navigator should save the URLs in a field called URLLinks{n} in the Notes/Domino document. (The Web Navigator creates a new URLLinks{n} field each time the field size reaches 64K. For example, the first URLLinks field would be URLLinks1, the second would be URLLinks2, and so on.)

If you save the URLs, you can use them in agents; for example, you could create an agent that opens Web pages in the Web Navigator database and then loads all the Web pages saved in each of the URLLinks{n} field(s).

CAUTION:
Saving URLS in the URLLinks{n} field(s) may affect performance.

[RELOAD] : [URLLIST]

Keywords. Optional. Specify both keywords to force a reload of the Web page and save the URLs in the URLLinks{n} field in the Notes/Domino document.
charset

Text. Optional. Enter the MIME character set (for example, ISO-2022-JP for Japanese or ISO-8859-1 for United States) that you want the Web Navigator to use when processing the Web page. Only use this parameter when the Web Navigator detects the MIME character set of the URL contents incorrectly.

webusername

Text. Optional. Some Internet servers require you to obtain a user name before you can access their pages. This parameter allows you to enter the user name that you previously obtained from the Internet server.

webpassword

Text. Optional. Some Internet servers require you to obtain a password before you can access their pages. This parameter allows you to enter the password that you previously obtained from the Internet server.

proxywebusername

Text. Optional. Some proxy servers require that you specify a user name in order to connect through them. This parameter allows you to enter the user name for the proxy server. See your administrator for the user name required by the proxy.

proxywebpassword

Text. Optional. Some proxy servers require that you specify a password in order to connect through them. This parameter allows you to enter the password for the proxy server. See your administrator for the password required by the proxy.

Usage

The @URLOpen function works from both the Notes/Domino workstation and server.

The user name and password parameters work only with the Notes Web Navigator. Other browsers always prompt for authentication.

For use on the server, you need to specify at least one parameter with the function; using the function without any parameters will attempt to display the URL Open dialog box which cannot be done from the server. If you want to use any of the parameters that follow the Reload and URLList keywords without specifying values for either of the keywords, enter a zero (0) in place of the keyword value(s). For example, @URLOpen("http://www.ibm.com";0;"myusername";"mypassword").

When a Notes browser triggers the @URLOpen function, it displays the retrieved Web page in a new window. When the @URLOpen function is used on a form or page that is accessed by a non-Notes browser, Domino generates a javascript window.open command with the following syntax:

```
window.open([sURL [, sName [, sFeatures [, bReplace]]]])
```

To display the retrieved Web page in a new window, pass the values for sName and sFeatures (if desired) as comma-separated arguments within the urlstring. For example, @URLOpen("http://www.ibm.com","NEW"). Be sure to use double quotes at the beginning and end of the urlstring parameter, and single quotes before and after each comma separating the arguments to be passed to window.open. Do not include any spaces.

To open another design element from the current Notes database in a Web application, use the @WebDbName function to properly encode the database name.
See the topic "Domino URL Commands" in the Application Development with Domino Designer guide for a list of the URL commands you can use to open design elements in a browser.

**Language cross-reference**

URLOpen method of LotusScript NotesUIWorkspace class

GetDocumentByURL method of LotusScript NotesDatabase class

getDocumentByURL method of Java Database class

**Examples: @URLOpen**

1. This example displays the URL Open dialog box that allows a user to enter the URL.

```
@URLOpen
```

2. This example opens the www.acme.com Web page from the database if it is found there. If the page is not found in the database, it is retrieved from the Web, loaded into the database, and then opened.

```
@URLOpen("http://www.acme.com")
```

3. This example retrieves the www.acme.com Web page from the Web, loads it into the database, and then opens it.

```
@URLOpen("http://www.acme.com"; 1)
```

4. The following code, when added to an action on the "Purchasing" Web application form, opens the "CustomerInfo" Notes form, which resides in the same database:

```
@URLOpen(@WebDbName + "/CustomerInfo?OpenForm")
```

5. The following code, in a document viewed from the Web, will open the www.acme.com Web page in the same window (_self) as that document.

```
@URLOpen("http://www.acme.com")
```

6. The following code, in a document viewed from the Web, will open the www.acme.com Web page in a new window (_blank). Note that the window will not have any sFeatures assigned by previous javascript commands.

```
@URLOpen("http://www.acme.com','_blank")
```

7. The following code, in a document viewed from the Web, will open the www.acme.com Web page in a new window (_blank). All sFeatures will be inherited from the calling window.

```
@URLOpen("http://www.acme.com','_blank','")
```

8. The following code, in a document viewed from the Web, will open the www.acme.com Web page in a new window (NEW). The window will not inherit any sFeatures.

```
@URLOpen("http://www.acme.com','NEW")
```

9. In a document viewed from the Web, clicking on the following hyperlink will open a new window (mywindow) displaying the www.yahoo.com Web page.

```
<a href="javascript: mywin = window.open('http://www.yahoo.com','mywindow');mywin.focus()">yahoo</a>
```

The following code will open the www.acme.com Web page in the mywindow window. All sFeatures will be inherited from the mywindow calling window.

```
@URLOpen("http://www.acme.com','mywindow','")
```

**@UrlQueryString**

In a Web application, returns the current URL command and parameters, or the value of one of the parameters.

**Note:** This function is new with Release 6.
Syntax
@UrlQueryString( parameterName )

Parameters

parameterName

Text. Optional. The name of a parameter in the URL command.

Return value

query

Text or text list.

• If the parameter is not specified, the return value is the URL command name (first list element) followed by the parameters (name, equal sign, value).

• If the parameter is specified, the return value is the value of the parameter or null if the parameter does not exist.

Usage

@UrlQueryString is useful in formulas that run in the context of a browser.

The Notes client always returns null for this formula.

Examples: @UrlQueryString

For these examples, the URL command is:
http://www.acme.com/marketing.nsf?OpenForm&ID=986574&Category=Golf

1. This example:
   @UrlQueryString
   returns the list:
   • OpenForm
   • ID=986574
   • Category=Golf

2. This example:
   @UrlQueryString("Category")
   returns the text:
   • Golf

@UserAccess

Given a server and file name, indicates the current user’s level of access to the database.

Note: If you used @UserAccess in Release 4, it is automatically converted to @V4UserAccess in Release 5 or later to preserve the functionality of your formulas. If you change those formulas to use @UserAccess, be sure to recompile them under Release 5. If you use @UserAccess in Release 5, a database created in Release 4 will not recognize the formula until you upgrade that database to Release 5. If the formula will be evaluated in Release 4, use @V4UserAccess.

Note: The AccessPrivilege keyword option is new with Release 6.

Syntax

@UserAccess( server : file ; [ accessPrivilege ] )
Parameters

server

Text. The name of the server. Use an empty string ("") to indicate the local computer.

file

Text. The path and file name of the database. Specify the database's path and file name using the appropriate format for the operating system.

[accessPrivilege]

Keyword. Optional. Specify one of the following keywords to return a user’s access level or test for a specific database privilege, instead of returning a list containing all of the user’s access information:

• [ACCESSLEVEL] returns a number from 1 to 6 that indicates the user’s access level to the database.

<table>
<thead>
<tr>
<th>Level</th>
<th>User’s access level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depositor</td>
</tr>
<tr>
<td>2</td>
<td>Reader</td>
</tr>
<tr>
<td>3</td>
<td>Author</td>
</tr>
<tr>
<td>4</td>
<td>Editor</td>
</tr>
<tr>
<td>5</td>
<td>Designer</td>
</tr>
<tr>
<td>6</td>
<td>Manager</td>
</tr>
</tbody>
</table>

The following return 1 (True) if the user has the specified privilege and 0 (False) if the user does not. These privileges are assigned in the Access Control List for the database.

• [CREATEDOCUMENTS]
• [DELETEDOCUMENTS]
• [CREATEPERSONALAGENTS]
• [CREATEPERSONALFOLDERSANDVIEWS]
• [CREATELOTUSSCRIPTJAVAAGENTS]
• [CREATESHARED FOLDERSANDVIEWS]
• [READPUBLICDOCUMENTS]
• [WRITEPUBLICDOCUMENTS]
• [REPLICATEORCOPYDOCUMENTS]

Return value

If you specify one or more keywords, returns a text value or a text list containing the following values:

• The [AccessLevel] keyword returns a value of 1 through 6.
• The other keywords return a value of 1 or 0.

If you specify no keywords, returns a text list of values for the following keywords:

• [AccessLevel] : [CreateDocuments] : [DeleteDocuments] : [CreatePersonalAgents] :
[CreatePersonalFoldersAndViews] : [CreateSharedFoldersAndViews] : [CreateLotusScriptJavaAgents] :
[ReadPublicDocuments]: [WritePublicDocuments]

@UserAccess does not test for access to the ReplicateOrCopyDocuments privilege by default.

Tip: If the multi-value separator for the field containing the formula is a semicolon, the values in the returned text list are separated by semicolons instead of colons.
Usage

On a local database without "Enforce a consistent Access Control List," @UserAccess without the second parameter always returns 6; 1; 1; 1; 1; 1; 1. If the current user has No Access to the database, Lotus Notes/Domino displays a message: "You are not authorized to perform that operation."

This function does not work in column or selection formulas, or in agents that run on a server (mail and scheduled agents). Hence it does not work with the Evaluate statement.

Language cross-reference

QueryAccess method of LotusScript NotesDatabase class

CurrentAccessLevel property of LotusScript NotesDatabase class

queryAccess method of Java Database class

CurrentAccessLevel property of Java Database class

Examples: @UserAccess

1. This formula returns the text list 3: 1: 1: 1: 0 if the user has Author access, permission to create documents, delete documents, create private agents, create personal views and folders, but does not have permission to create shared views and folders in the NUN.NSF database in the DISCUSS directory on server Gaborone.

   @UserAccess( "Gaborone" : "discuss\nun.nsf" )

2. This formula, when added to a form action button, creates a new document using the MyOpinion form if the current user has the privilege to create documents in the current (nun.nsf) database.

   @If(@UserAccess( "" : "discuss\nun.nsf" ; [CREATEDDOCUMENTS]) = "1";@Command([Compose];"MyOpinion");@Prompt([OK];
   "Access denied";"Sorry, you do not have permission to create documents in this database.")

3. This formula returns the text list 6: 1: 1: 1: 1: 1: 1: 1 if the user has Manager access and permission to create and delete documents, create private agents, create personal and shared views and folders, create LotusScript and/or Java agents, read and write public documents in the current database. The text list displays as 6; 1; 1; 1; 1; 1; 1; 1 if the multi-value separator for the field containing this formula is a semicolon.

   @UserAccess( @dbName )

@UserName

Returns the current user name.

If the user name is hierarchical, @UserName returns it in canonical format (including the CN, OU, O, and C identifiers). To return the name in abbreviated format (omitting those identifiers), use @V3UserName.

Notes

- If you used @UserName in Release 3, it is automatically converted to @V3UserName in Release 4 or later to preserve the functionality of your formulas. If you change those formulas to use @UserName, be sure to recompile them under Release 4 or later. If you use @UserName in Release 4 or later, a database created in Release 3 will not recognize the formula until you upgrade that database. If the formula will be evaluated in Release 3, use @V3UserName.

- With Release 5, @UserName returns the alternate name as well as the primary name which is associated with the ID.

Syntax

@UserName ( index )
Parameters

index

Note: This parameter is new with Release 5.

Number. Optional. Indicating the index of user names. 0 is for primary name and 1 is for the alternate name. If this parameter is omitted, @UserName returns the primary name.

Return value

name

Text. The primary or alternate user name.

Usage

When a formula runs on a server, the agent signer is considered the current user. Using @UserName on a local database or in a private view in a server-based database returns the user’s name. You should not use @UserName in a public view, doing so produces unpredictable results. Also, if the field that you are referencing changes, you will get unpredictable results because the index has to be rebuilt to accommodate the new information.

One use for @UserName is to display only those documents relevant to the current user. For example, your Service Request database could use @UserName in the private view named Assignments to display each technician’s assignments, weeding out everyone else’s:

SELECT @UserName=AssignedTo

However, the user can still design a different private view that retrieves all documents, so don’t depend on @UserName as a security mechanism.

For an alternative way to display only documents relevant to the current user, see “To show a single category in an embedded view.”

If you are using Release 5 and have an alternate name as well as a primary name, it is best to store the alternate name in the document as author information when using the extended feature of @UserName.

Language cross-reference

UserName property of LotusScript NotesSession class

UserName property of Java Session class

Examples: @UserName

1. This example returns CN=Robert T. Katsushima/OU=JPN/O=Acme if this is the name associated with the current user ID.
   @UserName(0)
2. This example returns Robert T. Katsushima.
   @Name([CN];@UserName)
3. This example returns CN=Rob Katsushima/OU=JPN/P=Acme if this is the first alternate name associated with the current user ID.
   @UserName(1)
4. This example returns Fletcher if William Fletcher is the name associated with the current User ID.
   @Right(@UserName;" ")
5. This example returns FLETCHER if William Fletcher is the name associated with the current User ID.
@UpperCase(@Right(@UserName;" "))

If the user id is a hierarchical id, the following code returns FLETCHER:
@UpperCase(@Right(@Name([CN];@UserName);" "))

6. This example returns the name in canonical format as shown below. Given this hierarchical user ID:
CN=Mary Tsen/OU=Illustration/OU=Documentation/OU=Development/OU=R&O/O=WorkSavers/C=US. To return the name in abbreviated format (omitting the CN, OU, O, and C identifiers), use @V3UserName.

@UserName

@UserNameLanguage

Returns language tags associated with the user ID.

Note: This @function is new with Release 5.

Syntax

@UserNameLanguage( index )

Parameters

index

Number. Indicates the index of user names. 0 is for primary name and 1 is for alternate name. Numbers greater than 1 are not used but reserved for future use.

Return value

namelanguage

Text. Language tag for the alternate user name. If the user does not have the alternate name, @UserNameLanguage returns an empty string (""). Also, this function returns an empty string for the primary name.

Usage

The alternate name is expected to be used for a user’s native language name.

Generally the native language name contains non-ASCII characters and cannot be displayed correctly without some proper fonts. The return value from @UserNameLanguage is used as reference of the native language.

@UserNameLanguage can be used as a default value formula to store the author’s alternate language tag in their document as well as their primary name and alternate name. While referring to the language tag, the Domino application can switch the display name on the document between the primary name and the alternate name.

See @Locale for a list of language codes.

Language cross-reference

Language property of LotusScript NotesName class

Language property of Java Name class
Examples: @UserNameLanguage

1. The following example returns "ja" if you have a Japanese name for your alternate name.
   @UserNameLanguage(1)
2. The following example returns an empty string (""") because the primary name has no language tag associated.
   @UserNameLanguage(0)

@UserNamesList

For a database on a server or a local database with "Enforce a consistent Access Control List across all replicas" in effect, @UserNamesList returns a text list containing the following information for the current user:

- Common name
- All hierarchical names (fully distinguished) that include the user name; for example, CN=My Name/OU=My Org Unit/O=My Org, plus */OU=My Org Unit/O=My Org, */O=My Org, and *
- Any roles associated with the user in the ACL
- All groups to which the user belongs (only if the database is on a server)

Note: This @Function is new with Release 5.

Syntax

@UserNamesList

Return value

names

Text list. Each list item is a name or role as specified above. Returns an empty string (""") if the current database is local and "Enforce a consistent Access Control List across all replicas" is not in effect, and the database is not replicated with the server database at least once.

Usage

This function does not work in column, selection, mail agent, or scheduled agent formulas.

Choose File - Database - Access Control, Advanced to set "Enforce a consistent Access Control List across all replicas."

@UserRoles returns a subset of the information returned by @UserNamesList.

Examples: @UserNamesList

This subform formula selects a different subform depending on whether the user is a member of the Marketing team or not. This formula works if the database containing it is on a server.

@If(@IsMember("Marketing Team"; @UserNamesList);
   "Marketing Head";
   "Generic Head")

@UserPrivileges

Returns a text list of the current user’s privileges. This function returns only the position of the privilege in the privilege list, not the name of the privilege.

Note: This @function is obsolete in Release 3. Use @UserRoles instead.
Syntax
@UserPrivileges

Return value
privileges

Text or text list.

Usage
This function does not work in scheduled agent formulas.

You cannot use this function in Web applications.

Language cross-reference
NotesACLEntry class
ACLEntry class

Examples: @UserPrivileges
1. A database has five privileges. User Mary Tsen has been assigned Privileges 2 and 3. This example returns the text list 2:3 (which displays as 2;3 if the multi-value separator for the field containing the formula is semicolon).

   @UserPrivileges

2. This form formula causes the Marketing Report form to be used if the current user has been assigned the first privilege in the list (regardless of what it is called); otherwise, the Main Topic form is used.

   @If(@UserPrivileges = "1"; "Marketing Report"; "Main Topic")

@UserRoles

For a database on a server or a local database with "Enforce a consistent Access Control List across all replicas" in effect, returns a list of roles that the current user has. Roles are defined in a database’s access control list.

Syntax
@UserRoles

Return value
roles

Text list. Each item in the list is the name of a role that the current user has in the current database. The role names are enclosed in brackets. Returns an empty string ("") if the current database is local and "Enforce a consistent Access Control List across all replicas" is not in effect.

Usage
This function does not work in column, selection, mail agent, or scheduled agent formulas.

Choose File - Database - Access Control, Advanced to set "Enforce a consistent Access Control List across all replicas."

@UserRoles appends $$WebClient to the list of roles when a Web user opens a database.
@UserRoles returns a subset of the information returned by @UserNamesList.

Language cross-reference
Roles property of LotusScript NotesACLEntry class

Roles property of JavaACLEntry class

Examples: @UserRoles
1. This example displays the roles assigned to the current user. The roles are displayed in brackets.
   @UserRoles
2. This code, if added to the New Document action button of a database that has the Enforce a consistent ACL across all replicas checkbox selected on the Advanced tab of the ACL Properties box, opens the Manager form if the [Manager] role is assigned to the current user; otherwise it open the Employee form in a Notes application.
   @Command([Compose];"";@If(@IsMember("[Manager]";@UserRoles);"Manager";"Employee"))
3. This subform formula selects a different subform depending on whether the user is a Web client or not. The WebClient role is a role that is automatically created by Lotus Notes/Domino; it does not require the surrounding brackets, but does require the leading double dollar signs.
   @If(@IsMember("$WebClient";@UserRoles);"WebSubform";"NotesSubform")

@V2If
This function performs an @If operation; the syntax is the same as for @If.

Syntax
@V2If( condition1 ; action1 ; condition2 ; action2 ; condition99 ; action99 ; else_action )

Usage
Use @V2If when you expect your application to be used with Lotus Notes Release 2.x. If the application will only be used with Lotus Notes Release 3 or later, you should use @If. The @If function in Release 3 was redesigned to work in conjunction with the new @functions first available in Release 3, such as @Prompt. Due to these changes, releases of Lotus Notes earlier than Release 3 cannot evaluate @If correctly, and return an error message.

Note: In applications created with Lotus Notes prior to Release 4, the @If function is automatically renamed to @V2If during the upgrade to Release 4.

Language cross-reference
If...Then...Else statement of LotusScript language
If...GoTo statement of LotusScript language
If...Then...Elseif statement of LotusScript language

@V3UserName
Returns the current user name or server name. Using @V3UserName on a local database or in a private view in a server-based database returns the user’s name.

If the user name is hierarchical, @V3UserName returns the name in abbreviated format (omitting the CN, OU, O, and C identifiers). To return the name in canonical format, use @UserName.
Note: If you used @UserName in Release 3 of Notes, it is automatically converted to @V3UserName in Release 4 and later to preserve the functionality of your formulas. If you change those formulas to use @UserName, be sure to recompile them. If you use @UserName in Release 4 or later, a database created in Release 3 does not recognize the formula until you upgrade that database. If the formula will be evaluated in Release 3, use @V3UserName.

Syntax
@V3UserName

Return value
name

Text. The current user name or server name.

Usage
When a formula runs on a server, the server is considered the current user, so @V3UserName returns the name of the server. We do not recommend using @V3UserName in a public view. Doing so produces unpredictable results.

One use for @V3UserName is to display only those documents relevant to the current user. For example, your Service Request database could use @V3UserName in the private view named Assignments to display each technician’s assignments, weeding out everyone else’s:

SELECT @V3UserName=AssignedTo

However, the user can still design a different private view that retrieves all documents, so don’t depend on @V3UserName as a security mechanism.

Language cross-reference
UserName property of LotusScript NotesSession class

UserName property of Java Session class

Examples: @V3UserName
1. @V3UserName returns Robert T. Smith if this is the name associated with the current user ID and returns Robert T. Smith/LA/Deli if this is the hierarchical name associated with the user ID.
2. @Right(@V3UserName; " ") returns Fletcher if William Fletcher is the name associated with the current user ID.
   If the user ID is hierarchical, the following code returns Fletcher:
   @Right(@Name([CN]; @V3UserName); " ")
3. @UpperCase(@Right(@V3UserName; " ")) returns FLETCHER if William Fletcher is the name associated with the current user ID.
   If the user ID is hierarchical, the following code returns FLETCHER:
   @UpperCase(@Right(@Name([CN]; @V3UserName); " "))
4. Given this hierarchical user ID:
   CN=Mary Tsen/OU=Illustration/OU=Documentation/OU=Development/OU=R&D/O=WorkSavers/C=US
   @V3UserName returns the name in abbreviated format:
   Mary Tsen/Illustration/Documentation/Development/R&D/WorkSavers/US
   To return the name in canonical format (using the CN, OU, O, and C identifiers), use @UserName.
@V4UserAccess

Given a server and file name, indicates the current user’s level of access to the database.

**Note:** This @function is new with Release 5. If you used @UserAccess in Release 4, it is automatically converted to @V4UserAccess in Release 5 or later to preserve the functionality of your formulas. With Release 5 and later, more user access information is returned by @UserAccess. If you change those formulas to use @UserAccess, be sure to recompile them under the later release. If you use @UserAccess in Release 5 or later, a database created in Release 4 does not recognize the formula until you upgrade it. If the formula will be evaluated in Release 4, use @V4UserAccess.

**Syntax**

@V4UserAccess( server : file )

**Parameters**

server

Text. The name of the server. Use an empty string (""") to indicate the local computer.

file

Text. The path and file name of the database. Specify the database’s path and file name using the appropriate format for the operating system.

**Return value**

level ; create ; delete

Text list.

- **level** is a number from 1 to 6 that indicates the user’s access level to the database.

<table>
<thead>
<tr>
<th>Level</th>
<th>User’s access level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Depositor</td>
</tr>
<tr>
<td>2</td>
<td>Reader</td>
</tr>
<tr>
<td>3</td>
<td>Author</td>
</tr>
<tr>
<td>4</td>
<td>Editor</td>
</tr>
<tr>
<td>5</td>
<td>Designer</td>
</tr>
<tr>
<td>6</td>
<td>Manager</td>
</tr>
</tbody>
</table>

- **create** is a number that returns 1 (True) if the user can create documents in the database, and 0 (False) if not.

- **delete** is a number that returns 1 (True) if the user can delete documents from the database, and 0 (False) if not.

On a local database without "Enforce a consistent Access Control List," @V4UserAccess always returns 6; 1; 1. If the current user has No Access to the database, Lotus Notes/Domino displays a message: “You are not authorized to perform that operation.”

**Usage**

This function does not work in column or selection formulas, or in agents that run on a server (mail and scheduled agents).
**Language cross-reference**
QueryAccess method of LotusScript NotesDatabase class

CurrentAccessLevel property of LotusScript NotesDatabase class

queryAccess method of Java Database class

CurrentAccessLevel property of Java Database class

**Examples: @V4UserAccess**

1. This formula returns 4; 1; 1 if the user has Editor access, permission to create documents, and permission to delete documents, in a database with the path of DSource\lookup.nsf on server Galactica/Space/Federation.
   @V4UserAccess( "Galactica//Space//Federation" ; "dsource\lookup.nsf" )

2. This formula returns 6;1;1, despite the user's access level and permissions, since the customer.nsf database is running on the local server. Or if the user has No Access to the database, "You are not authorized to perform that operation" displays instead.
   @V4UserAccess("\Lotus\Notes\Data\customer.nsf")

3. This formula returns 6; 1; 0 if the user has Manager access, permission to create documents, and no permission to delete documents in the current database if the database is running on a server other than the local server.
   @V4UserAccess( 0@DbName )

**@ValidateInternetAddress**

Validates an Internet address based on the RFC 822 or RFC 821 Address Format Syntax.

**Note:** This @function is new with Release 5.

**Syntax**

```
@ValidateInternetAddress( [ addressFormat ] ; Address )
```

**Parameters**

[ addressFormat ]

Keyword. Specifies the formatting with which to validate an Internet address. Can be one of the following keywords:

[ADDRESS821]
Requests input address be validated based on RFC821 Address Format Syntax.

SStreitfeld@gazette.com

[ADDRESS822]
Requests input address be validated based on RFC822 Address Format Syntax.

"Streitfeld, Sara (Miami)" <SStreitfeld@gazette.com>

Address

Input address string
**Return value**

- If validation is successful, the NULL string is returned.
- If validation fails, an error message string is returned to the user specific to the failure. More error message strings will be added in the future as necessary.

**Possible error messages**

"Invalid Input Parameter"

Invalid parameters to @function - @ValidateInternetAddress.

"Invalid RFC821 syntax, no Phrase required."

When a phrase is present in an address requiring an RFC821 syntax.

"Invalid Phrase or character found."

Phrase part of 822 address invalid.

"Invalid Quoted String or mismatched quotes found."

Quoted string is invalid within the address.

"Invalid comment or mismatched parenthesis found."

Embedded (comment(s)) within address is invalid.

"Invalid or missing Domain."

Invalid or missing Domain part of Address.

"Invalid LocalPart or character found."

Invalid LocalPart specified.

**Usage**

@ValidateInternetAddress is currently used in location records to validate Internet address fields as well as in mail forms. This function is most useful in field validation formulas where users are asked to input their Internet address or in computed fields where Internet addresses are inherited.

**Note:** Multi-byte, or 8-bit characters, are allowed in the Phrase part of an RFC 822 format Internet address. They are not allowed anywhere else. Also, the Group syntax (i.e. several Internet addresses combined into one group name, such as "Customers") is not supported in the validator.

**Examples: @ValidateInternetAddress**

You have designed a form asking the user to input an Internet address. The user enters a standard RFC 821 format Internet address SSReitfeld@gazette.com in the editable field User_Address.

If you enter the field validation formula

```plaintext
validateAddress := @ValidateInternetAddress([Address821]; User_Address);
@If(validateAddress != ""; @Failure(validateAddress); @Success)
```

the validation formula returns the NULL string indicating a successful validation.
However if you enter
"Streitfeld, Sara (Miami)" <Streitfeld@gazette.com>
the validation formula returns the following error message:
"Invalid RFC821 syntax, no Phrase required."

@VerifyPassword
Compares two passwords.

Note: This @function is new with Release 6.

Syntax
@VerifyPassword( password ; password )

Parameters
password
Text. This can be a text expression or a password field name.

Return value
flag
Boolean.
• Returns 1 (True) if the passwords are equivalent.
• Returns 0 (False) if the passwords are not equivalent.

Usage
Use this function to verify which password format, @Password or @HashPassword, was used to encode a password field.

@Language cross-reference
VerifyPassword method of LotusScript NotesSession class
verifyPassword method of Java Session class

Examples: @VerifyPassword
1. This example returns true:
   @VerifyPassword("tolstoy";@HashPassword("tolstoy"))
2. This example returns false because the hashed string contains an upper-case T:
   @VerifyPassword("tolstoy";@HashPassword("Tolstoy"))
3. If the access field is a password field containing the string, "He++llo", this code returns true:
   @VerifyPassword(access;@Password(access))
4. This code returns false because the @HashPassword and @Password functions use different formats to encode the contents of the access field:
   @VerifyPassword(@HashPassword(access);@Password(access))

@Version
Returns the release number of the Notes/Domino software you’re running.
Syntax
@Version

Return value

versionNumber

String. The release number.

Usage

In column, selection, mail agent, and scheduled agent formulas, @Version returns the release number of the Notes/Domino server or workstation containing the database. In all other formulas, @Version returns the release number of the Notes/Domino workstation running the formula.

The following table maps the numbers returned by @Version to each Notes/Domino version.

<table>
<thead>
<tr>
<th>Number Returned by @Version</th>
<th>Corresponding Lotus Notes/Domino version</th>
</tr>
</thead>
<tbody>
<tr>
<td>114</td>
<td>Notes 3.x</td>
</tr>
<tr>
<td>136</td>
<td>Notes 4.0, 4.0x</td>
</tr>
<tr>
<td>138</td>
<td>Notes 4.1, 4.1x</td>
</tr>
<tr>
<td>145</td>
<td>Notes 4.5, 4.5x</td>
</tr>
<tr>
<td>147</td>
<td>Notes 4.6</td>
</tr>
<tr>
<td>166</td>
<td>Notes 5.0, 5.0x</td>
</tr>
<tr>
<td>190</td>
<td>Notes 6.0, 6.0.1</td>
</tr>
<tr>
<td>191</td>
<td>Notes 6.0.2</td>
</tr>
<tr>
<td>194</td>
<td>Notes 6.0.3, 6.5</td>
</tr>
<tr>
<td>256</td>
<td>Notes 7.0</td>
</tr>
</tbody>
</table>

Note the following:

- @Version returns the same number for all releases of Notes 3.x.
- @Version doesn’t distinguish between the maintenance releases of Notes 4.x.

Language cross-reference

NotesBuildVersion property of LotusScript NotesSession class

NotesVersion property of LotusScript NotesSession class

NotesVersion property of Java Session class

@ViewShowThisUnread

Changes a view to show only unread documents, or to show read and unread documents.

Note: This @function is new with Release 6.5.

Syntax

@ViewShowThisUnread( unreadOnly )
Parameters
unreadOnly

Text.
• The value "1" shows only unread documents.
• The value "0" (or any value but "1") shows read and unread documents.

Return value
flag

Number. Returns 1 (True).

Usage
This @function is intended for use in view actions.

This @function does not work on the web.

@ViewTitle

Returns the current view’s name. If there are aliases and synonyms, they are returned in a text list.

Syntax
@ViewTitle

Return value
title

Text or text list.

Usage
This function works in toolbar button, hotspot, or form action formulas, if the formula opens to a view using an @command such as FileOpenDatabase. It can be used in hide-when formulas for view action bars, but not for other hide-when formulas. Returns the name of the view that was last accessed when used in field, form action, section editor, or window title formulas or null if no view has been accessed. It does not work in column, selection, mail agent, paste agent, or scheduled agent formulas.

Language cross-reference

Name property of LotusScript NotesView class

ViewName property of LotusScript NotesUIView class

Name property of Java View class

Examples:  @ViewTitle

1. This example returns Main View if that is the title of the current view.
   @ViewTitle
2. This example returns "Main View":"By Date" if the view name is Main View\By Date.
   @ViewTitle
3. This example returns MAIN VIEW if the title of the current view is "main view" in any combination of uppercase and lowercase letters.
@UpperCase(@ViewTitle)

@WebDbName

Returns the name of the current database encoded for URL inclusion.

Note: This @function is new with Release 6.

Syntax
@WebDbName

Return value
databaseName

Text. The URL encoded name of the database.

Usage

The return value can be placed as is in a URL command.

URL encoding changes most special characters to the text %xx where xx is a hexadecimal number representing the value of the character. In particular, spaces are changed to %20.

A backslash (\) is changed to a forward slash (/) rather than encoded. Double backslashes (\\) are removed. Dashes (-) are passed through as is.

The file extension starting with the period is not encoded.

This function is most effective when used in Web applications. When executed from the Notes client, with @URLOpen, for example, specify the host name before this function or the URL command will not execute properly:
@URLOpen("//hostname/" + @WebDbName + "/viewname?OpenView")

Language cross-reference

GetURLHeaderInfo method of LotusScript NotesDatabase class

getURLHeaderInfo method of Java Database class

Examples: @WebDbName

In an application accessed from the Web, this action opens "View A" in the current database. Note that "View+A" could also be written as "View%20A" in the formula.
@URLOpen(@WebDbName + "/View+A?OpenView")

@Weekday

Computes the day of the week and returns a number that identifies the day.

Syntax
@Weekday( time–date )
Parameters

*time-date*

Time-date. The date having the weekday value you want.

Return value

*weekdayNumber*

Number. Weekday numbers are 1 through 7, with Sunday = 1, Monday = 2, and so on.

Language cross-reference

Weekday function of LotusScript language

Examples: *@Weekday*

1. This example returns 5.
   
   `@Weekday([9/29/88])`

2. This example returns 2 if the date in the response field happens to fall on a Monday.
   
   `@Weekday(ResponseDate)`

3. This example returns the string Working on the Weekend if the contents of the field named ResponseDate is 7 (Saturday) or 1 (Sunday); otherwise, it returns the date the document was created as a text string.
   
   `@If(@Weekday(ResponseDate) = 7 | @Weekday(ResponseDate) = 1;"Working on the Weekend";@Text(@Created))`

@While

Executes one or more statements iteratively while a condition remains true. Checks the condition before executing the statements.

Note: This @function is new with Release 6.

Syntax

`@While( condition ; statement ; ... )`

Parameters

*condition*

Expression that returns a value of True (1) or False (0).

*statement*

A formula language statement. The maximum number of statements you can include is 254.

Return value

*true*

True (1) unless an error occurs during execution of the condition. An "unexpected data type" error occurs if the conditional expression results in a non-numeric value.

Usage

@While evaluates the condition. If the condition is True (1), @While executes the statements then evaluates the condition again. If the condition is False (0), @While terminates.
Tip: If you are looping through a field containing a list, be sure the Allow multiple values check box is selected in the Field Properties box for the list field.

For other iterative statements, see \@DoWhile and \@For.

Language cross-reference
While statement of LotusScript language

Examples: \@While
This agent displays the elements of the Categories field one at a time.
\begin{verbatim}
n := 1;
\@While(n <= @Elements(Categories));
@Prompt([OK]; "Category " + @Text(n); Categories[n]);
n := n + 1)
\end{verbatim}

\@Wide
Converts half-pitch alphanumerical characters (single-byte characters -- SBCS) in the specified string to full-pitch alphanumerical characters (double-byte characters -- DBCS). This function works in Japanese, Korean, Simplified Chinese, and traditional Chinese environments. In the Japanese environment, this function can convert half-pitch Katakana as well.

Note: This \@function is new with Release 5.

Syntax
\@Wide( string )

Parameters
string

Text. The string you want to convert to double-byte characters.

Return value
returnstring

Text. The string converted to double-byte characters.

Usage
This function can be used in input translation formulas to convert a field’s contents to double-byte characters or in computed field formulas to save space for displaying a string.

Language cross-reference
StrConv function of LotusScript language

Examples: \@Wide
1. This input translation formula returns "Tokyo" as a full-pitch character, if the Location field contains a half-pitch character expression of "Tokyo."
\@Wide(Location)
2. This computed field formula returns "New York" as a full-pitch character, to save space for displaying the string.
@Word

Returns the specified word from a text string. A "word" is defined as the part of a string that is delimited by the defined separator character. For example, if you specify a space (" ") as the separator, then a word is any series of characters preceded by and followed by a space (or the beginning or end of the string).

@Word( string ; separator ; number )

Syntax

string

Text or text list. The string you want to scan.

separator

Text. The character that you want used to delimit a word in the string.

number

Number. A position indicating which word you want returned from string. A positive number refers to the position of the word starting from the beginning where 1 is the first word. A negative number refers to the position of the word starting from the end where -1 is the last word.

Return value

word

Text or text list. The word that holds the position specified by the number in the string; for example, if number is 3, @Word returns the third word in the string. If a text list is used, @Word returns (in list format) a word from each list that holds the specified position. Returns an empty string if number is out of range, except that 0 is equivalent to 1.

Language cross-reference

StrToken function of LotusScript language

Examples: @Word

1. This example returns Collins,
   @Word("Larson, Collins, and Jensen"; " " ; 2)
2. This example returns Collins:Marketing,
   @Word("Larson, Collins, and Jensen":"Sales, Marketing, and Administration";" ";2)
3. This example returns M.; here, the specified separator is the comma. The string contains 3 words:
   Larson, James, and M.
   @Word("Larson,James,M.";",";3)
4. This example returns Larson if James Larson is the name associated with the current user ID. It returns M. if James M. Larson is the name associated with the current user ID.
   @Word(@Username; " ";2)
5. This example returns Larson if James Larson is the name associated with the current user ID. It also returns Larson if James M. Larson is the name associated with the current user ID.
   @Word(@Username; " ";-1)
@Year

Extracts and returns the year from the specified time–date value.

Syntax

@Year( time–date )

Parameters

time–date

Time-date. The time-date of the year you want.

Return value

year

Number. The year of time–date. @Year returns the year relative to the time zone in which the date was generated. Returns -1 if the time-date provided contains only a time value and not a date.

Language cross-reference

Year function of LotusScript language

Examples: @Year

This example returns 1995.
@Year([9/29/95])

@Yes

Returns the value 1.

Syntax

@Yes

Return value

yes

Number. The number 1.

Usage

This function is equivalent to @True.

Language cross-reference

Built-in constants of LotusScript language

Examples: @Yes

1. This example returns 1.
@Yes

2. This example returns 1 if the value in the Cost field is greater than 100.
@If(Cost>100;@Yes;@No)
@Yesterday

Returns the time–date value which corresponds to yesterday’s date.

Syntax
@Yesterday

Return value
_yesterday_

Time-date. Yesterday’s date.

Usage
Using @Yesterday in column or selection formulas may impact the efficiency of your application. It also causes the view refresh indicator to display constantly.

In a field formula, Lotus Notes/Domino takes the value for @Yesterday from the clock in the client computer.

Language cross-reference
Yesterday property of LotusScript NotesInternational class

Yesterday property of Java International class

Examples: @Yesterday
1. This example returns 12/31/92 if today is January 1, 1993.  
   @Yesterday
2. This example returns 8/16/93 if today is August 17, 1993.  
   @Yesterday

@Zone

Returns the time zone setting of the current computer or of a time-date value, and indicates if daylight-saving time is observed.

The time zone is represented as the number of hours that must be added to the time-date to convert it to Greenwich Mean Time.

Syntax
@Zone @Zone( timeDate )

Parameters
_timeDate_

Time-date. Optional. The time-date whose zone you want to know. You must specify both a date and a time; otherwise, @Zone returns 0.

Return value
_zoneNumber . dstFlag_

Number. The time zone, followed by a period, followed by a flag indicating daylight-saving time.
- For time zones east of GMT, `zoneNumber` is negative.
- For time zones west of GMT, `zoneNumber` is positive.
- When you use `@Zone` with no parameter, and daylight-saving time is being observed on the current computer, `dstFlag` is 1. If daylight-saving time is not being observed, only the `zoneNumber` is returned.
- When you use `@Zone` with a parameter, and the specified date falls within the daylight-saving time boundary, `dstFlag` is 1. If the date does not fall within daylight-saving time, only the `zoneNumber` is returned.

**Usage**

When used without a parameter, `@Zone` returns the zone and daylight-saving time setting of the current computer.

When used with the parameter `currentTimeDate`, `@Zone` returns the zone and daylight-saving time setting of `currentTimeDate`.

**Time zones that are not full-hour increments from GMT**

For time zones that are not a full hour increment from GMT, the return value is:

```
mnhh . dstFlag
```

- `mm` is the minutes component of the time relative to GMT.
- `hh` is the hours component of the time relative to GMT
- `dstFlag` is .1 if daylight-saving time is being observed. Otherwise, only the `mmhh` is returned.

For example, on a computer with a time zone setting eleven and a half hours west of GMT, with daylight-saving time disabled, `@Zone` returns: 3011

On a computer with a time zone setting ten and three-quarter hours west of GMT, with daylight-saving time enabled, `@Zone` returns: 4510.1

On a computer with a time zone setting nine and a half hours east of GMT, with daylight-saving time enabled, `@Zone` returns: -3009.1

**Language cross-reference**

TimeZone property of LotusScript NotesDateTime class

ZoneTime property of LotusScript NotesDateTime class

IsDST property of LotusScript NotesDateTime class

Time Zone property of LotusScript NotesInternational class

TimeZoneFmt property of LotusScript NotesViewColumn class

TimeZone property of Java DateTime class

ZoneTime property of Java DateTime class

IsDST property of Java DateTime class

Time Zone property of Java International class
TimeZoneFmt property of Java ViewColumn class

Examples: @Zone

1. This example returns:
   - 5.1 for Eastern Standard Time and daylight saving time observed.
   - 5 for Eastern Standard Time and daylight saving time not observed.
   - 6 for Central Standard Time and daylight saving time not observed.
   - 7.1 for Mountain Standard Time and daylight saving time observed.
   - 8.1 for Pacific Standard Time and daylight saving time observed.

2. This example returns 5 if in the Eastern Standard time zone.
   @Zone([1/26/94 11:00 AM])

3. This example returns 5.1 if in the Eastern Standard time zone and daylight saving is observed, 5 if daylight saving time is not observed.
   @Zone([5/28/94 11:00 AM])
Chapter 7. Formula Language @Commands A-Z

This documentation shows the syntax and usage for all the @commands, in alphabetical order. It also includes examples, wherever appropriate.

For information on how to use @commands, see the following topics:
- Using @commands
- Working with @commands
- @Commands with ECL security

Using @Commands

An @command executes a Domino command. All of the standard menu commands can be executed using @commands. In addition, a number of specialized commands are available.

Syntax

```
@Command( [ commandName ] ; parameters )
```

Parameters

```
[ commandName ]
```

The name of the @command you want to perform.

```
parameters
```

Zero, one, or more parameters, depending on the @command you’re calling. Separate parameters with semicolons.

Return value

Number.
- 1 if the @command executes successfully
- 0 if the @command does not execute successfully

@Commands with ECL security

The following table lists the @commands affected by an execute control list (ECL). Those @commands do not execute on the workstation unless the marked ECL privileges are granted to the formula’s signer.

The ECL flags listed in the table are:
- Access to the file system (file)
- Access to current database (cur)
- Access to non-Notes databases (db)
- Access to external code (code)
- Access to external programs (prog)
- Ability to send mail (mail)
• Ability to export data (exp)

<table>
<thead>
<tr>
<th>@command</th>
<th>file</th>
<th>cur</th>
<th>db</th>
<th>code</th>
<th>prog</th>
<th>mail</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminSendMailTrace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentEnableDisable</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentRun</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentSetServerName</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentTestRun</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AttachmentDetachAll</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AttachmentLaunch</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DesignRefresh</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditClear</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditCopy</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditCut</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditDetach</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditInsertFileAttachment</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditInsertObject</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditOpenLink</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditPaste</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditPasteSpecial</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EditUntruncate</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EmptyTrash</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExchangeUnreadMarks</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileDatabaseCompact</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileExport</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileImport</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileOpenDatabase</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileOpenDbRepID</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FilePrint</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileSave</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileSaveNewVersion</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Folder</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FolderDocuments</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FolderMove</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FolderRename</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailForward</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailForwardAsAttachment</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailRequestCrossCert</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailRequestNewName</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailRequestNewPublicKey</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@command</td>
<td>file</td>
<td>cur</td>
<td>db</td>
<td>code</td>
<td>prog</td>
<td>mail</td>
<td>exp</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------</td>
<td>-----</td>
<td>----</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>MailSend</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailSendCertificateRequest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MailSendEncryptionKey</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MailSendPublicKey</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ObjectOpen</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenDocument</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveFromFolder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ReplicatorReplicateHigh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReplicatorReplicateNext</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReplicatorReplicateSelected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReplicatorReplicateWithServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReplicatorSendMail</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReplicatorSendReceiveMail</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReplicatorStart</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunAgent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>RunScheduledAgents</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SetCurrentLocation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ToolsCategorize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToolsReplicate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToolsRunBackgroundMacros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ToolsRunMacro</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ToolsScanUnreadSelected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ViewRefreshUnread</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AddBookmark @Command**

Adds a bookmark with the specified URL or current object.

**Note:** This @command is new with Release 5.

**Syntax**

@Command([AddBookmark]; urlstring)

@Command([AddBookmark]; urlstring; title)

@Command([AddBookmark]; urlstring; title; folder)

**Parameters**

urlstring

Text. Optional. The URL string for the database that you would like to bookmark. If you specify a null ("" string, this function takes the current object using the Domino URL scheme as the object to be bookmarked. You can specify the url string for a database by specifying its replica id, without the colon, and preceding it with the "Notes://" prefix. The Notes URL for a database displays in the Address bar at the top of the Notes client window if you click the down arrow button.
title

Text. Optional. The title that you would like to specify for the bookmark. If you specify a null ("") string, you get a default title.

folder

Text. The name of the folder where you would like to place the bookmark. If you specify a null ("") string, this function will select either an open bookmark page, or a default bookmark folder.

Usage

This command does not work on the Web.

Examples: AddBookmark

1. This example, when triggered from a hotspot button on a form, creates a bookmark, named "User Log," for the 873476CD:0070A010 replica of the UserLog database in the "Logs" folder on the current user's bookmark bar.

   @Command([AddBookmark];"Notes:///873476CD0070A010";"User Log";"Logs")

2. This example, when triggered from a hotspot button on a form, creates a bookmark, named "Feedback1," for the 641599VW:0030C007 replica of the Feedback database (which resides on the Acme/South/Notes server) in the "ProductOne Customers" folder on the current user's bookmark bar.

   @Command([AddBookmark];"Notes/Acme/641599VW0030C007";"Feedback1";"ProductOneCustomers")

AddDatabase @Command

Adds the specified database icon to the user's workspace, without opening the database.

Syntax

@Command([AddDatabase]; server : database ; bookmark )

Parameters

server

Text. The name of the server where the database is.

database

Text. The path and file name of the database.

bookmark

Note: This parameter is new with Release 5.

Text ("0" or "1"). Optional. Specify "1" to bring up the Add Bookmark dialog box. Here, you can select or create a folder in which the bookmark should be placed. If you specify "0" or omit this parameter, it will bookmark the database in the Databases folder.

Usage

This command does not work on the Web.

Language cross-reference

AddDatabase method of LotusScript NotesUIWorkspace class
Examples: AddDatabase
This formula adds TRADEMRK.NSF to the user’s workspace; this database is stored in the Document directory on the LEGAL server.
@Command([AddDatabase]; "LEGAL" : "Document\TRADEMRK.NSF")

AddDatabaseRepID @Command
Adds an icon to the desktop for the database specified by its replica ID.

Syntax
@Command([AddDatabaseRepID]; replicaID; serverHint; bookmark)

Parameters
replicaID
Text. The replica ID of the database to be added to the desktop.

serverHint
Text. Optional. The name of the server where the replica might reside. Notes checks this server for the replica before checking the other servers.

bookmark
Text ("0" or "1"). Optional. Specify "1" to bring up the Add Bookmark dialog box. Here, you can select or create a folder in which the bookmark should be placed. If you specify "0" or omit this parameter, it will bookmark the database in the Databases folder.

Note: This parameter is new with Release 5.

Usage
AddDatabaseRepID is similar to @Command([AddDatabase]) but uses a replica ID instead of a server/path name.

The database has to exist in the Notes Data directory on the server, otherwise it will not be found.

AddDatabaseRepID looks for the replica in the following order:
• Checks the user’s workspace
• Checks the server indicated in the serverHint
• Checks other servers known to the current session

This command does not work on the Web.

AddToIMContactList @Command
Adds a name or names to a personal group in the Instant Messaging Contact List.

Note: This @command is new with Release 6.5.

Syntax
@Command([AddToIMContactList]; names; personalGroup)
Parameters

names

Text list. The name or names to be added.

personalGroup

Text. Optional. The name of a personal group. If this parameter is omitted, the user is prompted with a list of personal group names.

AdminCertify @Command

Displays the Choose Certifier ID dialog box, where you can select a Certifier ID file. After selecting a Certifier ID and entering its password, you select the user or server ID to be certified.

Syntax

@Command( [AdminCertify] )

Usage

This command works almost anywhere in IBM Lotus Notes/Domino except from within a dialog box or on the Web.

Language cross-reference

RegisterNewCertifier method of LotusScript NotesRegistration class

registerNewCertifier method of Java Registration class

AdminCreateGroup @Command

Opens a Domino Directory and displays a blank Group form.

Note: This @command is obsolete in Release 6.

Syntax

@Command( [AdminCreateGroup] )

Usage

AdminCreateGroup is available only when the Domino Administrator is open.

When there is only one Domino Directory on the selected server, AdminCreateGroup opens it and displays a blank Group form so that you can add a new group to it. When there are multiple Domino Directories on the selected server, Notes/Domino displays a dialog box that allows you to select the Directory to open.

This command does not work on the Web.

AdminCrossCertifyIDFile @Command

Displays the Choose Certifier ID dialog box, where you can select a Certifier ID file. After you select a Certifier ID and enter its password, Notes displays the Choose ID to be Cross-Certified dialog box, which allows you to create a hierarchical cross certificate for an ID in another organization.
Syntax
@Command( [AdminCrossCertifyIDFile] )

Usage
This command does not work on the Web.

Language cross-reference
CrossCertify method of LotusScript NotesRegistration class
crossCertify method of Java Registration class

AdminCrossCertifyKey @Command
Displays the Choose Certifier ID dialog box, where you can select a Certifier ID file. After you select a Certifier ID and enter its password, Notes displays the Cross Certify Key dialog box, which allows you to create a cross certificate for an ID in another organization using the numeric key associated with that ID.

Syntax
@Command( [AdminCrossCertifyKey] )

Usage
This command does not work on the Web.

AdminDatabaseAnalysis @Command
Displays the Database Analysis dialog box, which provides information about the selected database on the selected server.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminDatabaseAnalysis] )

Usage
AdminDatabaseAnalysis is available only when the Domino Administrator is open.

This command does not work on the Web.

AdminDatabaseQuotas @Command
For the selected server, displays a list of the databases in which you can change the size limits.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminDatabaseQuotas] )

Usage
AdminDatabaseQuotas is available only when the Domino Administrator is open.

This command does not work on the Web.
Language cross-reference
SizeQuota property of LotusScript NotesDatabase class
SizeQuota property of Java Database class

AdminIDFileClearPassword @Command
Allows the administrator to delete the password associated with any user ID file, without having to first switch to that ID and make it active.

Syntax
@Command( [AdminIDFileClearPassword] )

Usage
This command works almost anywhere in Notes/Domino except from within a dialog box or on the Web.

AdminIDFileExamine @Command
Displays the Choose ID File to Examine dialog box. After the administrator selects an ID, Notes/Domino displays the ID Properties box, which contains information about security basics and your identity.

Syntax
@Command( [AdminIDFileExamine] )

Usage
This command works almost anywhere in Notes/Domino except from within a dialog box or on the Web.

AdminIDFileSetPassword @Command
Allows the administrator to assign a password to any user ID file, without having to first switch to that ID and make it active.

Syntax
@Command( [AdminIDFileSetPassword] )

Usage
This command can be used almost anywhere in Notes/Domino except from within a dialog box or on the Web. This command is particularly useful for changing the password on a Certifier ID.

Administration @Command
Opens the Domino Administrator if the Domino Administrator package is installed on the local machine of the user executing the command.

Syntax
@Command( [Administration] )

Usage
This command works anywhere in Notes/Domino except from within a dialog box or on the Web.
AdminNewOrganization @Command
Displays the Register Organization Certifier dialog box, where an administrator can create a hierarchical Certifier ID for an organization. After the administrator enters a name and password for the new Certifier ID, Notes asks the user where to save the ID file and then creates the ID.

Syntax
@Command( [AdminNewOrganization] )

Usage
This command works anywhere in Notes/Domino except from within a dialog box or on the Web.

Language cross-reference
RegisterNewCertifier method of LotusScript NotesRegistration class
registerNewCertifier method of Java Registration class

AdminNewOrgUnit @Command
Prompts for the Certifier ID password and then displays the Register Organizational Unit Certifier dialog box, where the administrator can create a hierarchical Certifier ID for an organizational unit.

Syntax
@Command( [AdminNewOrgUnit] )

Usage
This command works anywhere in Notes/Domino except from within a dialog box or on the Web.

Language cross-reference
OrgUnit property of LotusScript NotesRegistration class
OrgUnit property of Java Registration class

AdminOpenAddressBook @Command
Opens a Domino Directory on the selected server.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminOpenAddressBook] )

Usage
AdminOpenAddressBook is available only when the Domino Administrator is open.

When there is only one Domino Directory on the selected server, AdminOpenAddressBook opens it. When there are multiple Directories on the selected server, Notes/Domino displays a dialog box that allows you to select the Directory to open.

This command does not work on the Web.
**Language cross-reference**
AddressBooks property of LotusScript NotesSession class

AddressBooks property of Java Session class

---

**AdminOpenCatalog @Command**
Opens the database catalog (CATALOG.NSF) on the selected server.

**Syntax**
@Command( [AdminOpenCatalog] )

**Usage**
AdminOpenCatalog is available only when the Domino Administrator is open.

This command does not work on the Web.

---

**Language cross-reference**
IsDirectoryCatalog property of LotusScript NotesDatabase class

---

**AdminOpenCertLog @Command**
Opens the certification log (CERTLOG.NSF) on the selected server.

**Note:** This @command is obsolete in Release 6.

**Syntax**
@Command( [AdminOpenCertLog] )

**Usage**
AdminOpenCertlog is available only when the Domino Administrator is open.

In order for AdminOpenCertLog to operate successfully, there must be a copy of a database named certlog.nsf on the selected server.

This command does not work on the Web.

---

**AdminOpenGroupsView @Command**
Opens a Domino Directory on the selected server and displays its Groups view.

**Note:** This @command is obsolete in Release 6.

**Syntax**
@Command( [AdminOpenGroupsView] )

**Usage**
AdminOpenGroupsView is available only when the Domino Administrator is open.

When there is only one Domino Directory on the selected server, AdminOpenGroupsView opens it and displays its Groups view. When there are multiple directories on the selected server, Notes/Domino displays a dialog box that allows you to select the directory to open.
This command does not work on the Web.

**Language cross-reference**
GetView method of LotusScript NotesDatabase class
getView method of Java Database class

---

**AdminOpenServerLog @Command**
Opens the server log (LOG.NSF) on the selected server.

**Note:** This @command is obsolete in Release 6.

**Syntax**
@Command( [AdminOpenServerLog] )

**Usage**
AdminOpenServerLog is available only when the Domino Administrator is open.

This command does not work on the Web.

**Language cross-reference**
OpenDatabase method of LotusScript NotesUIWorkspace class

---

**AdminOpenServersView @Command**
Opens a Domino Directory on the selected server and displays its Servers view.

**Note:** This @command is obsolete in Release 6.

**Syntax**
@Command( [AdminOpenServersView] )

**Usage**
When there is only one Domino Directory on the selected server, AdminOpenServersView opens it and displays its Servers view. When there are multiple Domino directories on the selected server, Notes/Domino displays a dialog box that allows you to select the directory to open.

This command does not work on the Web.

**Language cross-reference**
GetView method of LotusScript NotesDatabase class
getView method of Java Database class

---

**AdminOpenStatistics @Command**
Opens the statistics reporting database (STATREP5.NSF) on the selected server.

**Note:** This @command is obsolete in Release 6.
Syntax
@Command( [AdminOpenStatistics] )

Usage
AdminOpenStatistics is available only when the Domino Administrator is open.

For AdminOpenStatistics to operate successfully, there has to be a database named STATREP5.NSF on the selected server. Domino creates this database the first time it starts the REPORT server event.

This command does not work on the Web.

Language cross-reference
OpenDatabase method of LotusScript NotesUIWorkspace class

AdminOpenUsersView @Command
Opens a Domino Directory on the selected server and displays its People view.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminOpenUsersView] )

Usage
AdminOpenUsersView is available only when the Domino Administrator is open.

When there is only one Domino Directory on the selected server, AdminOpenUsersView opens it and displays its People view. When there are multiple Domino directories on the selected server, Notes/Domino displays a dialog box that allows you to select the directory to open.

This command does not work on the Web.

Language cross-reference
GetView method of LotusScript NotesDatabase class

cgetView method of Java Database class

AdminOutgoingMail @Command
Displays the contents of the selected server’s MAIL.BOX file.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminOutgoingMail] )

Usage
AdminOutgoingMail is available only when the Domino Administrator is open.

This command does not work on the Web.
AdminRegisterFromFile @Command
Displays a series of dialog boxes for certifying multiple new users from a text file.

Syntax
@Command( [AdminRegisterFromFile] )

Usage
This command works almost anywhere in Notes/Domino except from within a dialog box or on the Web.

AdminRegisterServer @Command
Displays a series of dialog boxes for creating a new server ID.

Syntax
@Command( [AdminRegisterServer] )

Usage
This command works anywhere in Notes/Domino except from within a dialog box or on the Web.

Language cross-reference
RegisterNewServer method of LotusScript NotesRegistration class
registerNewServer method of Java Registration class

AdminRegisterUser @Command
Displays a series of dialog boxes for certifying new users.

Syntax
@Command( [AdminRegisterUser] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box or on the Web.

Language cross-reference
RegisterNewUser method of LotusScript NotesRegistration class
registerNewUser method of Java Registration class

AdminRemoteConsole @Command
Displays the Remote Server Console dialog box, where an administrator can enter server console commands from a workstation.

Syntax
@Command( [AdminRemoteConsole] )

Usage
This command works anywhere in Notes/Domino except from within a dialog box or on the Web.
AdminSendMailTrace @Command
Displays the Mail Path Tracing dialog box, which allows you to send a message to a location on the mail router and receive a trace message in return.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminSendMailTrace] )

Usage
AdminSendMailTrace is available only when the Domino Administrator is open.
You use AdminSendMailTrace to determine the source of a mail delivery failure or to see if it is possible to deliver mail to a specific address.
This command does not work on the Web.

AdminStatisticsConfig @Command
Opens the server events database (EVENTS4.NSF) on the selected server and displays its Servers to Monitor view.

Note: This @command is obsolete in Release 6.

Syntax
@Command( [AdminStatisticsConfig] )

Usage
AdminStatisticsConfig is available only when the Domino Administrator is open.
In order for AdminStatisticsConfig to operate successfully, there must be a database named EVENTS4.NSF on the selected server. Notes/Domino creates this database the first time it starts the EVENTS server task.
This command does not work on the Web.

Language cross-reference
GetView method of LotusScript NotesDatabase class
getView method of Java Database class

AdminTraceConnection @Command
Displays the Trace Connections dialog box, which allows you to test network connections through a passthru server.

Syntax
@Command( [AdminTraceConnection] )

Usage
This command does not work on the Web.
AgentEdit @Command
Opens the Agent Properties box for the currently selected agent.

Syntax
@Command( [AgentEdit] )

Usage
An agent must be selected in the Agents view of a database.
This command does not work on the Web.

Language cross-reference
GetAgent method of LotusScript NotesDatabase class
getAgent method of Java Database class

AgentEnableDisable @Command
Enables or disables the specified agent.

Syntax
@Command( [AgentEnableDisable]; agentName; enableState )

Parameters
agentName
Text. Optional. The name of a scheduled agent defined for the currently selected database. If you omit this parameter, AgentEnableDisable applies to the currently selected agent.

enableState
Number ("1" or "0"). Optional. A value of "1" specifies that the agent is to be enabled. A value of "0" specifies that the agent is to be disabled. If you omit this parameter, AgentEnableDisable changes the agent’s current state from enabled to disabled or from disabled to enabled.

Usage
You can omit both parameters when an Agents window has focus and a scheduled agent is selected. Otherwise, agentName is required.
This command does not work on the Web.

Language cross-reference
IsEnabled property of LotusScript NotesAgent class
IsEnabled property of Java Agent class

AgentLog @Command
Displays the log for the currently selected agent. The log contains information about when the agent last ran, what actions it performed, and when it finished running.
Syntax
@Command( [AgentLog] )

Usage
An agent must be selected in the Agents view of a database. The agent must have run at least once.

This command does not work on the Web.

Language cross-reference
LastRun property of LotusScript NotesAgent class
LastRun property of Java Agent class

AgentRun @Command
Runs the currently selected agent and then displays its log.

Syntax
@Command( [AgentRun] )

Usage
An agent must be selected in the Agents view of a database.

This command does not work on the Web.

Language cross-reference
Run method of LotusScript NotesAgent class
run method of Java Agent class

AgentSetServerName @Command
Specifies a scheduled agent to run on a particular server.

Syntax
@Command( [AgentSetServerName] ; agentName ; serverName )

Parameters
agentName
Text. The name of an existing scheduled agent in the selected database.

serverName
Text. Optional. The name of the server on which you want agentName to run. If you omit this parameter, Notes/Domino displays the Choose Server To Run On dialog box when AgentSetServerName executes.

Usage
This command does not work on the Web.
Language cross-reference
RunOnServer method of LotusScript NotesAgent class

runOnServer method of Java Agent class

AgentTestRun @Command
Displays a log for the currently selected agent, describing what actions the agent will perform when run.

Syntax
@Command( [AgentTestRun] )

Usage
An agent must be selected in the Agents view of a database.

This command does not work on the Web.

Language cross-reference
Comment property of LotusScript NotesAgent class

Comment property of Java Agent class

AttachmentDetachAll @Command
Displays the Save Attachments To dialog box, where you select a location for the current document’s attachments.

Syntax
@Command( [AttachmentDetachAll] )

Usage
• A document must be open in Read or Edit mode.
• A form or subform must be open in Design mode.

The document must have two or more attachments and at least one must be selected.

This @command does not work when used in a hotspot button. The hotspot button takes focus away from the attachment that must be selected to be detached. Use an action button instead.

This @command does not work on the Web. To detach an attachment from a form via a Web browser, a user can click the attachment’s icon.

Language cross-reference
ExtractFile method of LotusScript NotesEmbeddedObject class

extractFile method of Java EmbeddedObject class

AttachmentLaunch @Command
Launches the selected attachment by opening the application in which it was created, if possible.
Syntax
@Command( [AttachmentLaunch] )

Usage
- A document must be open in Read or Edit mode.
- A form or subform must be open in Design mode.
- An attachment must be selected.

This @command does not work in Web applications. If a rich-text field on a form contains an attachment on the Web and you click the attachment’s icon, Notes either launches the attachment (TXT files, for example) or prompts the user to detach the file by saving it to disk (NSF files, for example).

This @command does not work when used in a hotspot button. The hotspot button takes focus away from the attachment that must be selected to be launched. Use in an action button instead.

If the user’s machine does not have the program that runs the attachment being launched, the following error is generated, “Sorry, an application to open this document cannot be found.”

Language cross-reference
Shell function of LotusScript language

Examples: @Command([AttachmentLaunch])
This formula, when added to a "Launch" action button on a form, launches the Notepad file, LICENSE.TXT, which is attached to a rich-text field on the form. First, you click the LICENSE.TXT icon to select it, then you click the “Launch” button. The program NOTEPAD.EXE launches, displaying the LICENSE.TXT file.
@Command([AttachmentLaunch])

AttachmentProperties @Command
Displays the Properties box for the current attachment.

Syntax
@Command( [AttachmentProperties] )

Usage
- A document must be open in Read or Edit mode.
- A form or subform must be open in Design mode.
- An attachment must be selected.

This @command does not work when used in a hotspot button. The hotspot button takes focus away from the attachment that must be selected for the Properties box to display. Use in an action button instead.

This command does not work on the Web.

AttachmentView @Command
Launches the Attachment Viewer, which lets you view the contents of the selected attachment without opening the application in which it was created.
Syntax
@Command( [AttachmentView] )

Usage
• A document must be open in Read or Edit mode.
• A form or subform must be open in Design mode.
• An attachment must be selected.

This @command does not work when used in a hotspot button. The hotspot button takes focus away from the attachment that must be selected to be viewed. Use in an action button instead.

This command does not work on the Web.

CalendarFormat @Command
Changes the Calendar view to display one day, two days, one week, two weeks, one month, or one year.

Syntax
@Command( [CalendarFormat]; format )

Parameters
format

Text. Optional. Specify one of the following:
• "1" specifies the one day display.

Note: Option "1" is new with Release 5.
• "2" specifies the two day display.
• "7" specifies the one week display.
• "14" specifies the two week display.
• "30" specifies the one month display.
• "365" specifies the one year display. This option is only valid in Web applications.

Usage
With no parameters, CalendarFormat cycles to the next calendar display option, in this order: One Day, Two Days, One Week, Two Weeks, One Month. With parameters, CalendarFormat changes to the designated display.

A Calendar view must be open.

CalendarGoTo @Command
Goes to a particular date in a Calendar view.

Syntax
@Command( [CalendarGoTo]; timedate )

Parameters
time-date
Time-date. Optional. Enclose the desired date in square brackets, as in [12/09/96]. Any @functions that return a time-date value, such as @Now, @Today, @Yesterday, and @Tomorrow can also be used as the date parameter.

Usage
With no parameters, CalendarGoTo displays the View Calendar GoTo dialog box. With parameters, CalendarGoTo moves the view focus to the requested date.

A Calendar view must be open.

You can use this command with Web applications.

**CheckCalendar @Command**

Pops up a dialog box containing a one-day calendar view. The current database must contain a calendar view for this command to function properly.

**Syntax**

```plaintext
@Command([CheckCalendar]; timedate)
```

**Parameters**

*timedate*

Time-date. Optional. Specify the date you want to view in the one-day calendar that displays in the dialog box. If no date is supplied, today’s date displays by default.

**Usage**

You can add this command to action button code in a mail memo to enable recipients of meeting invitations to check their calendars without having to toggle between their Calendar and Inbox views; the recipient can click and drag the dialog box.

This command does not work on the Web.

**Examples: CheckCalendar**

1. The following code, when added to an action button on a Notice form, which is a form that notifies meeting invitees of the time and location of a meeting, displays a dialog box containing a one-day view of the invitee’s calendar. Notes determines the day to display in the view based on the dates specified in the StartDate, DueDate, or NewStartDate fields, in that order. It displays the calendar information for the first date it encounters; if none of the fields contains a date, it displays today’s date:

   ```plaintext
   DateToDisplay := @If(@IsAvailable(StartDate);StartDate;
   @IsAvailable(DueDate);DueDate;@IsAvailable(NewStartDate);NewStartDate;
   @Today);
   @Command([CheckCalendar]; DateToDisplay)
   ```

**ChooseFolders @Command**

Displays the Folders dialog box, which allows you to select a folder in which to file the current document.

**Syntax**

```plaintext
@Command([ChooseFolders])
```
Usage
A document must be open in Edit mode.

This command does not work on the Web.

Language cross-reference
Folder method of LotusScript NotesUIWorkspace class

Clear @Command
Performs the menu command Edit – Delete.

Note: This command is new with Release 6.

Syntax
@Command( [Clear] )

Usage
This command executes immediately. Use the EditClear @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

In a view, folder, or a document in Read mode in a Notes application, marks the currently selected document for deletion. If you include functions that access or assign values to fields or properties in that document after this @command in a formula, they are ignored.

In a document in Edit mode, deletes the highlighted data (text, tables, graphics, links, file attachments, or objects).

In Web applications, only use this command on a form to delete the entire current document. It cannot be used to delete highlighted data on a form in Edit mode; if executed on a form, it deletes the entire document. You cannot use this command to mark selected documents in a view for deletion. Use the MoveToTrash @Command instead. To customize the "Deleted" confirmation page returned by the server, create a form named "$ReturnDocumentDeleted." See Customizing "Form processed" confirmation for the Web in the Application Development with Domino Designer guide for details.

In Notes applications, when this command is called on a form, subform, view, or folder in Design mode, it deletes the highlighted data, fields, or columns.

On the workspace, removes the selected icon (without permanently deleting the database from disk).

It is most convenient to use toolbar buttons to invoke this command.

Language cross-reference
Clear method in LotusScript NotesUIDocument class

DeleteDocument method of LotusScript NotesUIDocument class

DeleteDocument method of LotusScript NotesDocument Collection class

deleteDocument method of Java DocumentCollection class
Examples: Clear

1. The following code, when included in a form Delete action, deletes the current document and returns to the Locations view:
   
   ```
   @Command([Clear]);
   @Command([OpenView];"Locations")
   ```

CloseWindow @Command

Closes the current Notes window. If the document or design element in that window has not been saved, Notes prompts the user to save it before closing.

Note: This command is new with Release 6.

Syntax

```
@Command( [CloseWindow] )
```

Usage

This command executes immediately. Use the `FileCloseWindow @command` to execute after all @functions. See the [Order of evaluation for formula statements](#) topic for more details.

CloseWindow does not close the Notes workspace window.

When using this command on a form in Notes, you can prevent the user from being prompted to save any changes.

You can use this command with Web applications, as long as you enable the "Allow Javascript on the Web" setting on the Basics tab of the Database Properties box. Precede this command with `@Command([FileSave])` to simulate a Submit button.

Language cross-reference

Close method of LotusScript NotesUIView class

Examples: @Command([CloseWindow])

1. This code, when added to an action button on a form and accessed from a Web application, saves and closes the current document and opens the Results view. Following CloseWindow with the OpenView @command prevents the default Form Processed page from displaying and instead brings the user to the specified view.
   
   ```
   @Command([FileSave]);
   @Command([CloseWindow]);
   @Command([OpenView];"Results")
   ```

Compose @Command

Creates a new, blank document.

Syntax

```
@Command([Compose]; server : database ; form ; width ; height )
```

Parameters

```
server : database
```
Text list. The name of the server and database where you want to create the document. Null for server means the local Domino or Notes directory. Null for server and database means the current database.

form

Text. The name of the form you want to use when creating the new document.

width : height

Number list. Optional. The width and height, in inches, of the window for the document you compose. If you omit this parameter, or use zero for either value, you create the window at the default size (usually the size that the last user set).

Note: The width/height parameter has no effect in Release 5 and later.

Usage

To use this command in Web applications, use the following syntax:

@Command([Compose]; form)

When you compose a response document, make sure a database is open and a document is already selected at the view level. See ComposeWithReference for composing response documents with references to the main document.

When the command is used to compose main documents, the target database does not have to be open. This command adds a database icon to the workspace.

If the command is used in a view action, the form formula of the view will override the form specified in the @command. To avoid this problem add the following line to the form formula of the view:

@If(@IsNewDoc; @Return(Form); "")

For information about form formulas, see Form Formulas in "Programming Overview."

The width/height parameter does not apply in MDI mode when the window is maximized. When restored, the window returns to the size you specify. The measurement in inches matches the ruler bar in the editor, so that you can use the ruler bar to guide you in sizing the window. When you specify the width and height, you center the window in the enclosing Notes window (for MDI mode) or in the operating desktop (for Mac and SDI mode).

Language cross-reference

CreateDocument method in LotusScript NotesDatabase class

createDocument method in Java Document class

Examples: Compose

1. The following formula composes a new MainTopic document in the current database.
   @Command([Compose]; ""; "MainTopic")

2. The following formula composes a new Client Information document in the REPS.NSF database in the Westchester directory of the `sales/nyoffice` server.
   @Command([Compose]; "SALES//nyoffice": "Westchester\REPS.NSF"; "Client Information")

3. The following formula composes a mail memo in the user’s own Mail database.
   @Command([Compose]; @MailDbName; "Memo")
4. The following formula composes a response document based on the currently selected document, in a window four inches wide and seven inches high.
   \@Command([Compose];"";"Response"; 4:7)
5. The following code, when added to the "Next" hotspot button in form1 and triggered from the Web, saves form1 and opens form2 in edit mode.
   \@Command([FileSave]);
   \@Command([CloseWindow]);
   \@Command([Compose];"form2")

**ComposeWithReference @Command**

Creates a response document containing a reference to the main document.

**Note:** This @command is new with Release 6.

**Syntax**

\@Command([ComposeWithReference]; server : database ; form ; flags )

**Parameters**

*server : database*

Text list. The name of the server and database where you want to create the document. Null for server means the local Domino or Notes directory; must be null on the Web. Null for server and database means the current database.

*form*

Text. The name of the form you want to use when creating the new document. This form must contain a rich text or rich text lite field named "Body."

*flags*

Number. Optional. One or more of the following reference attributes. Combine attributes by adding them. Defaults to flags 3 plus 4. You must supply this parameter in Web applications.

- 0 -- includes Body in a collapsible section (flags 3 plus 4, the default). Cannot be combined with other flags. Does not work on Web.
- 1 -- Includes a reference to the main document. Cannot be combined with other flags. Does not work on Web.
- 2 -- Includes no history. Cannot be combined with other flags.
- 3 -- Includes the Body of the main document. Can be combined with the flags that follow.
- 4 -- Puts the Body of the main document in a collapsible section. Does not work on Web.
- 8 -- Includes Body as an Internet-style copy of the main document, with a "So-and-so wrote on ..." header and each line prefixed by a greater-than sign. Requires flag value 2. Implicitly applies flag 16. Does not work on Web.
- 16 -- Removes attachments, images, and other large objects from the reference copy, replacing them (except on the Web) with text statements in brackets.
- 32 -- For databases that contain a $ForwardSep subform (as exists in the Release 6 mail template), prefixes the reference copy with a forward separator. Does not work on Web.

**Usage**

When composing a response document, make sure a database is open and a document is already selected at the view level.
If this command is used in a view action, the form formula of the view will override the form specified in the @command. To avoid this problem add the following line to the form formula of the view:
@If(@IsNewDoc; @Return(Form); "")

For information about form formulas, see [Form Formulas] in "Programming Overview."

**Language cross-reference**

MakeResponse method of LotusScript NotesDocument class

makeResponse method of Java Document class

**Examples: ComposeWithReference**

1. This example, which could be a form or view action, creates a response document based on the Reply form containing a link to the main document.
   @Command([ComposeWithReference]; ";"; "Reply"; 1)

2. This example creates a response document with no copy of or reference to the main document.
   @Command([ComposeWithReference]; ";"; "Reply"; 2)

3. This example creates a response document containing a copy of the main document.
   @Command([ComposeWithReference]; ";"; "Reply"; 3)

4. These three examples are equivalent. Each creates a response document containing a copy of the main document in a collapsible section.
   @Command([ComposeWithReference]; ";"; "Reply"; 3 + 4)
   @Command([ComposeWithReference]; ";"; "Reply"; 0)
   @Command([ComposeWithReference]; ";"; "Reply")

5. This example creates a response document containing a copy of the main document in Internet style.
   @Command([ComposeWithReference]; ";"; "Reply"; 3 + 8)

6. This example creates a response document containing a copy of the main document in a collapsible section in Internet style.
   @Command([ComposeWithReference]; ";"; "Reply"; 3 + 4 + 8)

7. This example creates a response document containing a copy of the main document with attachments stripped.
   @Command([ComposeWithReference]; ";"; "Reply"; 3 + 16)

8. This example creates a response document containing a copy of the main document in a collapsible section with attachments stripped.
   @Command([ComposeWithReference]; ";"; "Reply"; 3 + 4 + 16)

9. This example, when used in an action button in a database containing a $ForwardSep subform, composes a forwarded mail memo referencing the main document.
   @Command([ComposeWithReference]; ";"; "Memo"; 3 + 32)

10. This example composes a forwarded mail memo in the current database. All attachments are stripped before the main document is copied to the new memo.
    @Command([ComposeWithReference]; ";"; "Memo"; 3 + 16 + 32)

11. This example composes a forwarded mail memo in the current database. The new document contains an Internet-style quoted copy of the reference document. All attachments, images and other large objects are stripped before the main document is copied to the new one.
    @Command([ComposeWithReference]; ";"; "Memo"; 3 + 8 + 32)

12. This example, when triggered from an action button on a form, checks if the current document has been saved. If not, it saves it, then opens the "responseDoc" response document, including the body of the main document in a collapsible section.
    @If(@IsNewDoc; @Do(@Command([FileSave]); "")
    @Command([ComposeWithReference];"";"responseDoc")
CreateAction @Command

Creates a new action and opens the design pane, where you can edit the action.

Syntax
@Command([CreateAction])

Usage
• A form or subform must be open in Design mode
  or
• A view or folder must be open in Design mode

This command does not work on the Web.

CreateAgent @Command

Creates a new agent in the current database and opens the Agent Properties box, where you can name and define the agent.

Syntax
@Command([CreateAgent])

Usage
A database must be open or selected on the workspace. The user must have at least Designer access to the database or have permission in the ACL to create private agents.

This command does not work on the Web.

CreateControlledAccessSection @Command

Creates a controlled access section on a form or subform. Unlike a regular section, a controlled access section has a formula to determine who can edit it.

Syntax
@Command([CreateControlledAccessSection])

Usage
A form or subform must be open in Design mode and the text you want in the section must be selected.

This command does not work on the Web.

CreateEllipse @Command

Lets you create an ellipse in a navigator. After you invoke the command, the cursor changes to a crosshair when you start to drag the mouse in the design area. You create an ellipse by dragging the mouse until the ellipse is the size you want.

Syntax
@Command([CreateEllipse])
Usage
A navigator must be open in Design mode.

This command does not work on the Web.

CreateFolder @Command
Displays the Create Folder dialog box, which lets you choose a location for a new folder and create it.

Syntax
@Command( [CreateFolder] )

Usage
A database must be open or selected on the workspace, and the user must have at least Designer access to the database, or have permission in the ACL to create personal folders.

This command does not work on the Web.

Language cross-reference
EnableFolder method of LotusScript NotesDatabase class
enableFolder method of Java Database class
PutInFolder method of LotusScript NotesDocument class
putInFolder method of Java Document class

CreateForm @Command
Creates a new, blank form in a database.

Syntax
@Command( [CreateForm] )

Usage
A database must be open or selected on the workspace, and the user must have at least Designer access to the database.

This command does not work on the Web.

CreateLayoutRegion @Command
Creates a new layout region on a form or subform.

Syntax
@Command( [CreateLayoutRegion] )

Usage
A form or subform must be open in Design mode.

This command does not work on the Web.
CreateNavigator  @Command
Creates a new, blank navigator in a database.

Syntax
@Command( [CreateNavigator] )

Usage
A database must be open or selected on the workspace, and the user must have at least designer access to the database.

This command does not work on the Web.

Language cross-reference
CreateViewNav method of LotusScript NotesView class
createViewNav method of Java View class

CreatePolygon  @Command
Lets you create a polygon in a navigator. After you invoke the command, the cursor changes to a crosshair when you start to drag the mouse in the design area. You create a polygon by clicking the mouse each time you want to start a new side. Double-click when you are done.

Syntax
@Command( [CreatePolygon] )

Usage
A navigator must be open in Design mode.

This command does not work on the Web.

CreatePolyline  @Command
Lets you create a polyline in a navigator. After you invoke the command, the cursor changes to a crosshair when you start to drag the mouse in the design area. You create a polyline by clicking the mouse each time you want to start a new line. Double-click when you are done.

Syntax
@Command( [CreatePolyline] )

Usage
A navigator must be open in Design mode.

This command does not work on the Web.

CreateRectangle  @Command
Lets you create a rectangle in a navigator. After you invoke the command, the cursor changes to a crosshair when you start to drag the mouse in the design area. You create a rectangle by dragging the mouse until the rectangle is the size you want.
CreateRectangularHotspot @Command

Lets you create a rectangular hotspot in a navigator. After you invoke the command, the cursor changes to a crosshair when you start to drag the mouse in the design area. You create a rectangular hotspot by dragging the mouse until the rectangle is the size you want.

Syntax
@Command( [CreateRectangularHotspot] )

Usage
A navigator must be open in Design mode.

This command does not work on the Web.

CreateSection @Command

Creates a section.

Syntax
@Command( [CreateSection] )

Usage
A document must be open in Edit mode
or
A form or subform must be open in Design mode

The text you want in the section must be selected. If you do not select any text, a blank section will be created.

This command does not work on the Web.

CreateSubform @Command

Creates a new, blank subform in a database.

Syntax
@Command( [CreateSubform] )

Usage
A database must be open or selected on the workspace, and the user must have at least Designer access to the database.

This command does not work on the Web.
CreateTextbox @Command

Lets you create a textbox in a navigator. After you invoke the command, the cursor changes to a crosshair when you start to drag the mouse in the design area. You create a textbox by dragging the mouse until the textbox is the size you want.

Syntax
@Command( [CreateTextbox] )

Usage
A navigator must be open in Design mode.

This command does not work on the Web.

CreateView @Command

Displays the Create View dialog box, which lets you choose a location for a new view and create it.

Syntax
@Command( [CreateView] )

Usage
A database must be open or selected on the workspace, and the user must have at least Designer access to the database.

This command does not work on the Web.

Language cross-reference
CreateView method of LotusScript NotesDatabase class

DatabaseDelete @Command

Permanently deletes the current database file from the hard disk where it is stored.

Note: This command is new with Release 6.

Syntax
@Command( [DatabaseDelete] )

Usage
This command executes immediately. Use the FileDatabaseDelete @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

A database icon must be selected, but the database cannot be open. The user must have Manager access in order to delete the database.

This command does not work on the Web.

Note: To remove the database icon from the user’s workspace without deleting the database, use @Command([FileDatabaseRemove])
Language cross-reference
Remove method of LotusScript NotesDatabase class

remove method of Java Database class

DatabaseReplSettings @Command
Displays the Replication Settings dialog box for the current database.

Syntax
@Command( [DatabaseReplSettings] )

Usage
A database must be open or selected on the workspace.

This command does not work on the Web.

DebugLotusScript @Command
Puts Designer into debug mode, so that all LotusScript scripts run in the debugger. The command is a toggle; selecting it again takes Notes out of debug mode.

Syntax
@Command( [DebugLotusScript] )

Usage
This command works almost anywhere in Notes/Domino except from within a dialog box or on the Web.

Once Designer is in debug mode and the Script Debugger window is open, you cannot select this @command again to terminate debugging. Use the Close Debugger button on the Script Debugger window instead.

Examples: @Command([DebugLotusScript])
This formula, when added to a hotspot button labeled Debug on a document, starts the script debugger. Once an event that has code associated with it is triggered, the Script Debugger window displays.
@Command([DebugLotusScript])

For instance, if a document with the Debug hotspot button has LotusScript code in its postmodechange event, when you open the document from a view (in read mode), click Debug, then double-click the document background to change it to edit mode, the Script Debugger window displays.

Tip: If you want to close the window and stop debugging, click the Close Debugger button in the Script Debugger window. If you decide to take Designer out of debug mode before the Debugger window displays, click the Debug hotspot button again to turn the debugger off.

DesignDocumentInfo @Command
Displays the Properties box for the current document.

Syntax
@Command( [DesignDocumentInfo] )
Usage

- The user must be at the view level with a document highlighted or
- A document must be open in read or Edit mode

If multiple documents have been selected, the Properties box displays for the highlighted document.

This command does not work on the Web.

**DesignFormAttributes @Command**

Displays the Properties box for the current form, subform, or page.

**Syntax**

@Command( [DesignFormAttributes] )

**Usage**

A form, subform, or page must be open in Design mode. It’s most convenient to use a toolbar button to invoke this command.

This command does not work on the Web.

**DesignFormFieldDef @Command**

Displays the Properties box for the currently selected field.

**Syntax**

@Command( [DesignFormFieldDef] )

**Usage**

A form or subform must be open in Design mode, and a field must be selected. It’s most convenient to use a toolbar button to invoke this command.

This command does not work on the Web.

**DesignFormNewField @Command**

Creates a new field on a form or subform.

**Syntax**

@Command( [DesignFormNewField] )

**Usage**

A form or subform must be open in Design mode and there must be no fields selected. It’s most convenient to use a toolbar button to invoke this command.

This command does not work on the Web.

**DesignForms @Command**

Displays the Design - Forms view of the current database.
Syntax
@Command( [DesignForms] )

Usage
The user must have at least Designer access to the database.
This command does not work on the Web.

DesignFormShareField @Command
Turns a single–use field into a shared field.

Syntax
@Command( [DesignFormShareField] )

Usage
A form or subform must be open in Design mode and a field must be selected. It’s most convenient to use a toolbar button to invoke this command.
This command does not work on the Web.

DesignFormUseField @Command
Displays the Insert Shared Field dialog box, where the user can select a shared field to place on the current form or subform.

Syntax
@Command( [DesignFormUseField] )

Usage
A form or subform must be open in Design mode and there must be no fields selected. It’s most convenient to use a toolbar button to invoke this command.
This command does not work on the Web.

DesignFormWindowTitle @Command
Displays the design pane and sets the Event edit control to Window Title, so you can define a formula for a form’s window title.

Syntax
@Command( [DesignFormWindowTitle] )

Usage
A form must be open in Design mode. It’s most convenient to use a toolbar button to invoke this command.
This command does not work on the Web.

DesignHelpAboutDocument @Command
Displays the About document in Edit mode for the current database.
Syntax
@Command([DesignHelpAboutDocument])

Usage
A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

DesignHelpUsingDocument @Command
Displays the Using document in Edit mode for the current database.

Syntax
@Command([DesignHelpUsingDocument])

Usage
A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

DesignIcon @Command
Displays the Design Icon dialog box, where you can edit the icon for the currently selected database.

Syntax
@Command([DesignIcon])

Usage
A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

DesignMacros @Command
Displays the Agents and Release 3 macros in the current database.

Syntax
@Command([DesignMacros])

Usage
A database must be open or selected on the workspace.

This command does not work on the Web.

DesignRefresh @Command
Displays the Refresh Database Design dialog box, where the user can choose a server that contains the design template for the current database.
Syntax
@Command( [DesignRefresh] )

Usage
A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

DesignReplace @Command
Displays the Replace Database Design dialog box, where the user can choose a design template to replace that of the current database.

Syntax
@Command( [DesignReplace] )

Usage
A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

DesignSharedFields @Command
Displays the Design - Shared Fields view in the current database.

Syntax
@Command( [DesignSharedFields] )

Usage
A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

DesignSynopsis @Command
Displays the Design Synopsis dialog box so the user can select the types of information to be included in the synopsis.

Syntax
@Command( [DesignSynopsis] )

Usage
This command does not work on the Web.

DesignViewAppendColumn @Command
Creates a column in a view or folder, positioned after the last column.
**Syntax**
@Command( [DesignViewAppendColumn] )

**Usage**
A view or folder must be open in Design mode.

This command does not work on the Web.

**Language cross-reference**
CreateColumn method of LotusScript NotesView class

---

**DesignViewAttributes @Command**
Displays the Properties box for the current view or folder.

**Syntax**
@Command( [DesignViewAttributes] )

**Usage**
A view or folder must be open in Design mode.

This command does not work on the Web.

---

**DesignViewColumnDef @Command**
Displays the Properties box for the currently selected column.

**Syntax**
@Command( [DesignViewColumnDef] )

**Usage**
A view or folder must be open in Design mode, and a column must be selected.

This command does not work on the Web.

---

**DesignViewEditActions @Command**
Displays or hides the action pane.

**Syntax**
@Command( [DesignViewEditActions] )

**Usage**
A view or folder must be open in Design mode. If the action pane is visible, DesignViewEditActions hides it; and if the action pane is not visible, DesignViewEditActions displays it.

This command does not work on the Web.
**DesignViewFormFormula @Command**

Displays the Design Form Formula dialog box, where the user can define a formula that determines which form is used to display documents opened from a particular view or folder.

**Syntax**

@Command( [DesignViewFormFormula] )

**Usage**

A view or folder must be open in Design mode.

This command does not work on the Web.

---

**DesignViewNewColumn @Command**

Creates a new column before the currently selected column in a view or folder.

**Syntax**

@Command( [DesignViewNewColumn] )

**Usage**

A view or folder must be open in Design mode. It’s most convenient to use a toolbar button to invoke this command.

This command does not work on the Web.

**Language cross-reference**

CreateColumn method of LotusScript NotesView class

---

**DesignViews @Command**

Displays the Design - Views view in the current database.

**Syntax**

@Command( [DesignViews] )

**Usage**

A database must be open or selected on the workspace and the user must have at least Designer access to the database.

This command does not work on the Web.

---

**DesignViewSelectFormula @Command**

Displays the design pane and sets the Define control to View Selection, which allows you to define a selection formula to determine which documents are displayed in a view.

**Syntax**

@Command( [DesignViewSelectFormula] )
**Usage**
A view must be open in Design mode.

This command does not work on the Web.

**Language cross-reference**
SelectionFormula property of LotusScript NotesView

**DialingRules @Command**
Displays the Dialing Rules dialog box, which allows you to define dialing rules for a modem in a Location document in a Domino Directory.

**Syntax**
@Command( [DialingRules] )

**Usage**
The Location document must be open and should have focus for this @command to work.

This command does not work on the Web.

**Directories @Command**
Displays the Directories dialog box which allows you to search for an address, view detailed information about an address entry, and add an entry to your Domino Directory with the Directories dialog box.

**Note:** This @command is new with Release 5.0.1

**Syntax**
@Command( [Directories] )

**Usage**
This command does not work on the Web.

**DiscoverFolders @Command**
Displays the "Folders containing current document" dialog box.

**Note:** This @command is new with Release 7.

**Syntax**
@Command( [DiscoverFolders] )

**Usage**
This command does not work on the Web.

This command works only in a design element containing an embedded outline with the "Maintain folder unread information" property selected. Elsewhere in the design a document must be selected. Otherwise, the error "Cannot execute the specified command" occurs.
The dialog box displays the names of the folders containing the document. The user can open a folder or remove the document from a folder. The dialog box is replaced by the message box "Document does not exist in any of the folders" if that is the case.

**EditBottom @Command**

Moves the insertion point to the bottom of a document or form.

**Syntax**

@Command( [EditBottom] )

**Usage**

- On a form in Design mode, EditBottom moves the insertion point to the bottom of the form as if the user had pressed ctrl+end.
- In a document in Edit mode, EditBottom moves the insertion point to the last editable field or button on the document.
- In a document in Read mode, EditBottom has no effect.
- You can precede this command with either:
  - @Command([EditDocument]; "1") to put the document into edit mode
  - @IsDocBeingEdited function to test if the document is in edit mode

This command does not work on the Web.

**Language cross-reference**

GotoBottom method in LotusScript NotesUIDocument class

**EditButton @Command**

Displays the design pane and the Properties box for the selected button.

**Syntax**

@Command( [EditButton] )

**Usage**

A document must be open in Edit mode and a button must be selected.

This command does not work on the Web.

**EditClear @Command**

Performs the menu command Edit – Delete.

**Syntax**

@Command( [EditClear] )

**Usage**

- This command executes after all @functions. Use @Command([Clear]) to execute immediately. See the Order of evaluation for formula statements topic for more details.
- In a view, folder, or a document in Read mode in a Notes application, marks the currently selected document for deletion.
In a document in Edit mode, deletes the highlighted data (text, tables, graphics, links, file attachments, or objects).

In Web applications, only use this command on a form to delete the entire current document. It cannot be used to delete highlighted data on a form in Edit mode; if executed on a form, it deletes the entire document. You cannot use this command to mark selected documents in a view for deletion. Use the MoveToTrash @Command instead. To customize the "Deleted" confirmation page returned by the server, create a form named "$ReturnDocumentDeleted." See Customizing "Form processed" confirmation for the Web in the Application Development with Domino Designer guide for details.

Using this command followed by @Command([EditGotoField]) produces an error.

In Notes applications, when this command is called on a form, subform, view, or folder in Design mode, deletes the highlighted data, fields, or columns.

On the workspace, removes the selected icon (without permanently deleting the database from disk).

This command executes only after the entire formula has been evaluated regardless of whether @Command or @PostedCommand is used.

It is most convenient to use a toolbar button to invoke this command.

Language cross-reference
Clear method in LotusScript NotesUIDocument class

DeleteDocument method of LotusScript NotesUIDocument class

DeleteDocument method of LotusScript NotesDocument Collection class
deleteDocument method of Java DocumentCollection class

Examples: EditClear

1. The following example, when used in the DeleteField action of a form in a Notes application, deletes the content of whichever field has focus when the DeleteField action button is pressed in edit mode:
   @Command([EditGotoField]; @ThisName);
   @Command([EditSelectAll]);
   @Command([EditClear])

2. The following example, when used in the Delete action of a form, deletes the current document opened in read mode on the Web. It then displays a customized form, if you created a form with the name $ReturnDocumentDeleted, or displays the default "Deleted" confirmation page.
   @Command([EditClear])

3. The following example, when used in a view action of a Notes application, deletes the documents selected in the view.
   @Command([EditClear])

EditCopy @Command
Performs the menu command Edit – Copy.

Syntax
@Command( [EditCopy] )

Usage

• In a view or folder, copies the selected documents to the Clipboard.
• In a document in Read or Edit mode, copies the highlighted data to the Clipboard.
• On a form, subform, view, or folder in Design mode, copies the highlighted data, fields, or columns to the Clipboard.
• You can use a toolbar button to invoke this command.

This command does not work on the Web.

**Language cross-reference**
Copy method in LotusScript NotesUIDocument class

GetSelectedText method in LotusScript NotesUIDocument class

**Examples: EditCopy**
This form action copies the selected text in a document that is in read or edit mode. If no text is selected, the entire document is copied.

```@Command([EditCopy])```

---

**EditCut @Command**
Performs the menu command Edit – Cut.

**Syntax**
```@Command([EditCut])```

**Usage**
• In a view or folder, deletes the selected documents and places them on the Clipboard.
• In a document in Edit mode, deletes the highlighted data and places it on the Clipboard.
• In a document in Read mode, EditCut has no effect.
• On a form, subform, view, or folder in Design mode, deletes the highlighted data, fields, or columns and places them on the Clipboard.
• It is most convenient to use a toolbar button to invoke this command.

This command does not work on the Web.

**Language cross-reference**
Cut method in LotusScript NotesUIDocument class

---

**EditDeselectAll @Command**
Performs the menu command Edit – Deselect All.

**Syntax**
```@Command([EditDeselectAll])```

**Usage**
• In a view or folder, deselects all selected documents.
• In a document in Read or Edit mode, deselects all highlighted data.
• On a form, subform, view, or folder in Design mode, deselects all highlighted data, fields, and columns.
• On the workspace, deselects all selected databases.

This command does not work on the Web.
**Language cross-reference**
DeselectAll method in LotusScript NotesUIDocument class
DeselectAll method in LotusScript NotesUIView class

---

**EditDetach @Command**
Detaches a file attachment to a location you specify.

**Syntax**
@Command( [EditDetach] )

or

@Command( [EditDetach] ; sourcefile ; targetfile )

**Parameters**

*sourcefile*
Text. The name of the attachment you want to detach.

*targetfile*
Text. A path and file name indicating where you want to put the file. Include the complete path specification (appropriate to the user’s operating system).

**Usage**
With no parameters, displays the Save Attachment dialog box for the current attachment. A document must be open in Read or Edit mode and the attachment must be selected.

This command is useful in action buttons; it cannot be added to a hotspot button. It does not work on the Web.

With both parameters, detaches the specified *sourcefile* and stores it using the *targetfile* path and file name without displaying the Save Attachment dialog box. A document must be open in Read or Edit mode, but since you are specifying which attachment to detach, the attachment does not have to be selected.

---

**Language cross-reference**
ExtractFile method of LotusScript NotesEmbeddedObject class

extractFile method of Java EmbeddedObject class

---

**Examples: EditDetach**

1. This formula displays the Save Attachment dialog box.
   @Command([EditDetach])

2. This formula detaches the 1–2–3 for Macintosh worksheet "Budget96" from the open document and stores it in the folder called "Worksheets" on "Macintosh HD," the user’s Macintosh workstation.
   @Command([EditDetach];"BUDGET96";"Macintosh HD:Worksheets:Budget96")
**EditDocument @Command**

Places the current document into the mode you specify. If you don’t specify a mode, toggles between Read and Edit mode.

**Syntax**

@Command( [EditDocument] )

or

@Command( [EditDocument] ; mode; previewpane )

**Parameters**

*mode*

Number. Specify “1” to place the document in Edit mode or "0" to place it in Read mode.

*previewpane*

Number. Specify “1” to edit the document in the preview pane.

**Usage**

In a view or folder, this @command opens the highlighted document in the specified mode. If the *mode* is omitted, Edit mode is assumed.

When you use this command with Web applications, it edits the current document. Use this command only on forms for the Web. The parameters for this function do not work on the Web.

On the Web, this command does not work in view actions. To open a document in Edit mode from a view, use @Command([OpenDocument]; "1") preceded by the OpenView @command.

When you edit a document using the @command([EditDocument]), the hidden attributes within rich-text fields are not honored. The hidden attributes are honored when the document is opened in Read mode with @Command([OpenDocument]).

**Language cross-reference**

EditDocument method of LotusScript NotesUIWorkspace class

**Examples: EditDocument**

1. This formula opens the current document in Edit mode.
   
   @Command( [EditDocument]; "1" )

2. This formula toggles the currently open document from Read mode to Edit mode, or vice versa. At the view level, opens the document in Edit mode.
   
   @Command( [EditDocument] )

3. This formula opens the current document into Edit mode in the preview pane if you are viewing from the preview pane. This feature is used in Domino.Action.
   
   @Command( [EditDocument]; "1"; "1" )

**EditDown @Command**

Moves the insertion point in a document down by the number of lines you specify. If you don’t specify a number, moves the insertion point down one line.
Syntax
@Command( [EditDown] )

or

@Command( [EditDown]; count )

Parameters

count

Number. Optional. Specifies the number of lines to move down.

Usage

- On a form or subform in Design mode, moves the insertion point down one line or count lines, as if the user had pressed down.
- In a document in Edit mode, moves the insertion point down within the current field, or if there are no more lines in the current field, in the next editable field (which must be below, not to the right of, the current field).
- In a document in Read mode, has no effect.

This command does not work on the Web.

Language cross-reference

GoToNextField method of LotusScript NotesUIDocument class

Examples: EditDown

This formula moves the insertion point down five lines.

@Command( [EditDown]; "5" )

EditEncryptionKeys @Command

Displays the Properties box for the current document, where you can edit its encryption keys.

Syntax

@Command( [EditEncryptionKeys] )

Usage

- In a view, the user can assign encryption keys to one or more selected documents.
- In a document in Read or Edit mode, the user can assign encryption keys to the current document.

This command does not work on the Web.

EditFind @Command

Performs the menu command Edit – Find/Replace.

Syntax

@Command( [EditFind] )
**Usage**

- In a view, displays the Find dialog box, whether the database is full-text indexed or not.
- In a document, displays the Find and Replace dialog box.

This command does not work on the Web.

**Language cross-reference**
Replace function of LotusScript language

FindString method in LotusScript NotesUIDocument class

FieldContains method of LotusScript NotesUIDocument class

---

**EditFindInPreview @Command**

Performs the menu command Edit - Find/Replace without moving the focus to the editing window. This allows searches to take place when the focus is on the view or folder panes.

**Syntax**

@Command( [EditFindInPreview] )

**Usage**

- When the preview pane is opened, displays the Find or Find/Replace dialog box (depending on whether the user is editing the document in the preview pane or not).
- When the preview pane is not opened, displays the Find dialog box (just as @Command([EditFind]) does).

This command does not work on the Web.

---

**EditFindNext @Command**

Performs the menu command Edit – Find Next.

**Syntax**

@Command( [EditFindNext] )

**Usage**

The following happens when EditFindNext operates on the result of a full-text search:

- In a view, highlights the next document that contains the search word or phrase.
- In a document, highlights the next occurrence of the search word or phrase.

When EditFindNext is not preceded by a full-text search, Notes displays the Find dialog box.

This command does not work on the Web.

---

**EditGotoField @Command**

In a document in Edit mode, places the insertion point in a field you specify.

**Syntax**

@Command( [EditGotoField] ; fieldName )
Parameters

fieldName

Text. The name of the field where you want to place the insertion point. The field must be editable.

Usage

A document must be open in Edit mode.

This command does not work on the Web.

Examples: EditGoToField command

1. This formula, when added to the “Apply font” hotspot button, applies the font a user selects from the “fonts” Dialog list field (which derives its list of fonts by choosing to Use formula for choices and entering @FontList as the formula) to the text the user enters or highlights in the "Body" Rich Text field. If no font was selected from the dialog list, an error message displays telling the user to select one.

@Command([EditGoToField];"Body");
@Command([EditSelectAll]);
result := @Command([TextSetFontSize];size);
@If(@IsError(result);Prompt([Ok];"Error encountered";"You must enter a font size first");result)

EditHeaderFooter @Command

Displays the Properties box for the current document or form, which allows you to set headers, footers, and other print attributes.

Syntax

@Command( [EditHeaderFooter] )

Usage

- A document must be selected from the view or opened in Read mode
  or
- A form or subform must be open in Design mode.

This command does not work on the Web.

EditHorizScrollbar @Command

Toggles the horizontal scroll bar in a document.

Note: EditHorizScrollbar is not supported under OS/2 or on the Macintosh.

Syntax

@Command( [EditHorizScrollbar] )

Usage

A document must be open.

This command does not work on the Web.
**Language cross-reference**
HorzScrollBar property of LotusScript NotesUIDocument class

---

**EditIndent @Command**
Indents a paragraph 1/4 inch. The entire paragraph is indented, as well as any paragraphs that are subsequently typed below it, until the user disables indenting.

**Syntax**
@Command( [EditIndent] )

**Usage**
- A form or subform must be open in Design mode
  or
- A document must be open in Edit mode, with the insertion point in a rich text field.

This command does not work on the Web.

---

**EditIndentFirstLine @Command**
Indents the first line of a paragraph 1/4 inch. The first line of the current paragraph is indented, as well as first lines of subsequently typed paragraphs, until the user disables indenting. (A carriage return defines a new paragraph.)

**Syntax**
@Command( [EditIndentFirstLine] )

**Usage**
- A form or subform must be open in Design mode
  or
- A document must be open in Edit mode, with the insertion point in a rich text field.

This command does not work on the Web.

---

**EditInsertButton @Command**
Creates a new button and displays the design pane, where the user can define a formula, simple action(s), LotusScript, or JavaScript for the button.

**Syntax**
@Command( [EditInsertButton] )
Usage

- A document must be open in Edit mode with the insertion point in a rich text field or
- A form or subform must be open in Design mode, with nothing selected.

This command does not work on the Web.

**EditInsertFileAttachment @Command**

Attaches a file to a document.

**Syntax**

@Command( [EditInsertFileAttachment] )

or

@Command( [EditInsertFileAttachment]; file; compress )

**Parameters**

*file*

Text. Optional. The name of the file you want to attach. Be sure to include the complete path specification (appropriate to the user's operating system).

*compress*

Number. Optional. Specify "1" if you want to compress the attachment. Specify "0" if you do not.

Without a *file* parameter, displays the Create Attachment dialog box. If a *file* is specified, attaches that file to the document without opening the Create Attachment dialog box.

Usage

A document must be open in Edit mode with the insertion point in a rich text field.

This command does not work on the Web.

**Language cross-reference**

EmbedObject method of LotusScript NotesRichTextItem class

embedObject method of Java RichTextItem class

**Examples: EditInsertFileAttachment**

1. This formula displays the Create Attachment dialog box.
   
   @Command( [EditInsertFileAttachment] )

2. This formula attaches the 1–2–3 for Macintosh worksheet "Budget96" to the document.
   
   @Command( [EditInsertFileAttachment]; "Worksheets:Budget96" )

**EditInsertObject @Command**

Inserts an object into a document, form, or subform.

**Note:** EditInsertObject is not supported by OS/2 and UNIX, or the Macintosh.
Syntax
@Command( [EditInsertObject] )

or

@Command( [EditInsertObject] ; object )

Parameters

object

Text. Optional. The name of the object you want to insert.

If an object name is included, Notes assumes it represents a registered OLE object and will attempt to insert a copy of it into the document or form. If the object name is omitted, Notes displays the Insert Object dialog box.

Usage

• A document must be open in Edit mode with the insertion point in a rich text field
  or
• A form or subform must be open in Design mode.

This command does not work on the Web.

Language cross-reference

EmbedObject method of LotusScript NotesRichTextItem class
embedObject method of Java RichTextItem class

Examples: EditInsertObject
This formula inserts a Word Pro OLE object into the document or form.
@Command( [EditInsertObject] ; "WordPro Document" )

EditInsertPageBreak @Command

Inserts a forced page break into a document, form, subform, or page.

Syntax

@Command( [EditInsertPageBreak] )

Usage

• A document must be open in Edit mode
  or
• A form, subform, or page must be open in Design mode.

This command does not work on the Web.

Language cross-reference

AddPageBreak method of LotusScript NotesRichTextItem class
Pagination property of LotusScript NotesRichTextParagraphStyle class
addPageBreak method of Java RichTextItem class

Pagination property of Java RichTextParagraphStyle class

**EditInsertPopup @Command**

Creates a hotspot that displays text.

**Syntax**

@Command( [EditInsertPopup] )

**Usage**

- A document must be open in Edit mode with text selected in a rich text field
  or
- A form, subform, or page must be open in Design mode and text must be selected.

This command does not work on the Web.

**EditInsertTable @Command**

Displays the Create Table dialog box, where the user can specify the number of rows and columns in a new table.

**Syntax**

@Command( [EditInsertTable] )

**Usage**

- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode, with nothing selected.

This command does not work on the Web.

**Language cross-reference**

AppendTable method of LotusScript NotesRichTextItem class

**EditInsertText @Command**

Inserts the specified string at the current cursor position.

**Syntax**

@Command( [EditInsertText] ; string )

**Parameters**

* string

Text. The string you want to insert.
Usage

- A document must be open in Edit mode with the insertion point in a text or rich text field or
- A form, subform, or page must be open in Design mode.

This command does not work on the Web.

Language cross-reference

GotoField method in LotusScript NotesUIDocument class

InsertText method in LotusScript NotesUIDocument class

Examples: EditInsertText

This formula inserts the text "Jones, Casey M." at the current position.

@Command( [EditInsertText]; "Jones, Casey M." )

EditLeft @Command

Moves the insertion point in a document, form, subform, or page to the left by the number of spaces you specify. If you don’t specify a number, moves the insertion point one space to the left.

Syntax

@Command( [EditLeft]; count )

Parameters

count

Number. Optional. The number of spaces you want to move.

Usage

- On a form, subform, or page in Design mode, moves the insertion point one or count spaces to the left, as if the user had pressed the left arrow key.
- In a document in Edit mode, moves the insertion point one or count spaces to the left within the current field, or if there are no more spaces in the current field, in the previous editable field (which may be to the left of, or above, the current field).
- In a document in Read mode, has no effect.

This command does not work on the Web.

Language cross-reference

GoToPrevField method of LotusScript NotesUIDocument class

Examples: EditLeft

This formula moves the insertion point four spaces to the left.

@Command( [EditLeft]; "4" )

EditLinks @Command

Displays the External Links dialog box.
Syntax
@Command( [EditLinks] )

Usage
• A document must be open in Edit mode
  or
• A form or subform must be open in Design mode
• The document, form, or subform must contain at least one DDE or OLE link.

This command does not work on the Web.

Language cross-reference
GetObject method of LotusScript NotesUIDocument class

EditLocations @Command
Opens the Personal Address Book to the Locations view.

Syntax
@Command( [EditLocations] )

Usage
This command does not work on the Web.

EditMakeDocLink @Command
Creates a link to the current document and copies it to the Clipboard. The user can paste the link into any rich text field.

Syntax
@Command( [EditMakeDocLink] )

Usage
• A document must be selected in a view
  or
• A document must be open in Read or Edit mode. A new document must be saved before you can create a link to it using this command.

After this command is triggered, the status bar displays, "DocLink copied to clipboard. Use Paste to insert it into a document."

This command does not work on the Web.

Language cross-reference
AppendDocLink method of LotusScript NotesRichTextItem class
appendDocLink method of Java RichTextItem class
Examples: @Command([EditMakeDocLink])

This formula, when added to a hotspot button on a form, creates a document link of the current document and copies it to the clipboard. If the document has new, it first saves it before creating the link.

@If(@IsNewDoc;@Do(@Command([FileSave]);@Command([EditMakeDocLink]));@Command([EditMakeDocLink]))

---

**EditNextField @Command**

In a document in Edit mode, moves the insertion point to the next editable field in the document, working left to right, top to bottom.

**Syntax**

@Command( [EditNextField] )

**Usage**

A document must be open in Edit mode.

This command does not work on the Web.

**Language cross-reference**

GoToNextField method of LotusScript NotesUIDocument class

---

**EditOpenLink @Command**

Opens the selected link.

**Syntax**

@Command( [EditOpenLink] )

**Usage**

This command does not work on the Web.

**Language cross-reference**

Activate method of LotusScript NotesEmbeddedObject class

activate method of Java EmbeddedObject class

---

**EditPaste @Command**

Performs the menu command Edit – Paste.

**Syntax**

@Command( [EditPaste] )

**Usage**

- In a view, pastes in documents that were cut or copied from another database.
- In a document in Edit mode or a form or subform in Design mode, pastes the contents of the Clipboard into the document, form, or subform. If the data was copied from a rich text field but is pasted into a non–rich text field, some information may be lost.
- In a document in Read mode, or if the Clipboard is empty, has no effect.
This command does not work on the Web.

**Language cross-reference**
Paste method of LotusScript NotesUIDocument class

---

**EditPasteSpecial @Command**
Displays the Paste Special dialog box.

**Syntax**
@Command([EditPasteSpecial])

**Usage**
- A document must be open in Edit mode with the insertion point in a rich text field or
- A form or subform must be open in Design mode.
- If the clipboard is empty, this command has no effect.

This command does not work on the Web.

---

**EditPhoneNumbers @Command**
Displays the Server/Connections view of the Personal Address Book.

**Syntax**
@Command([EditPhoneNumbers])

**Usage**
This command does not work on the Web.

---

**EditPrevField @Command**
In a document in Edit mode, moves the insertion point to the previous editable field in the document, working right to left, bottom to top.

**Syntax**
@Command([EditPrevField])

**Usage**
A document must be open in Edit mode.

This command does not work on the Web.

---

**Language cross-reference**
GoToPrevField method of LotusScript NotesUIDocument class

---

**EditProfile @Command**
Opens a new or existing profile document in Edit mode.
Syntax
@Command([EditProfile]; formname ; uniqueKey )

Parameters

formname
Text. The form upon which the profile is based. Must exist in the database.

uniqueKey
Text. Optional. The unique key that identifies the profile document.

Usage

This command executes after all functions. Use @Command([EditProfileDocument]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

You can access profile documents quickly and use them to store information that you don’t want in user documents and to share information across scripts within a database.

Only one profile of a given form can exist per database per key. If you create a profile without a key, Notes assumes it’s the only profile document of that form in the database. You need at least author access to create a profile that applies to an entire database.

Documents saved with EditProfile are hidden.

This function does not work in Web applications. Use @SetProfileField to create a profile document in a Web application.

Language cross-reference

EditProfile method in LotusScript NotesUIWorkspace class

GetProfileDocument method in LotusScript NotesDatabase class

Examples: EditProfile

1. This formula opens Mary Tsen’s Interest Profile in a discussion database.
   @Command([EditProfile];"Interest Profile";"Mary Tsen/Worksavers/US")

2. This formula creates a Calendar Profile document for the current user or opens the user’s existing Calendar Profile.
   @Command([EditProfile];"CalendarProfile";@UserName)

3. This formula creates a new Archive Profile document for the current database or opens the Archive Profile in Edit mode if it already exists.
   @Command([EditProfile];"Archive Profile")

4. This formula when used in the Domino Directory opens the Server\Setup Profile document.
   @Command([EditProfile];"Profile")

EditProfileDocument @Command

Creates a new or opens an existing profile document in Edit mode.

Note: This @command is new with Release 6.
Syntax
@Command( [EditProfileDocument] ; formname ; uniqueKey )

Parameters
formname
Text. The form upon which the profile is based. Must exist in the database.

uniqueKey
Text. Optional. The unique key that identifies the profile.

Usage
This command executes immediately. Use the EditProfile @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

You can access profile documents quickly and use them to store information that you don’t want in user documents and to share information across scripts within a database.

Only one profile of a given form can exist per database per key. If you create a profile without a key, Notes assumes it’s the only profile document of that form in the database. You need at least author access to create a profile that applies to an entire database.

Documents saved with EditProfileDocument are hidden.

This function does not work in Web applications. Use @SetProfileField to create a profile document in a Web application.

Language cross-reference
EditProfile method in LotusScript NotesUIWorkspace class

GetProfileDocument method in LotusScript NotesDatabase class

EditQuoteSelection @Command
Makes selected text look like an Internet-style reply by prefixing each line with a greater-than sign and removing attachments and other objects. This sets each line of the message to a default length, preventing it from wrapping in unexpected places when sent to non-Notes users.

Note: This @command is new with Release 6.

Syntax
@Command( [EditQuoteSelection] )

Usage
A document must be open in Edit mode and text must be selected.

The greater-than sign is the default prefix. You can specify a different prefix with the environment (notes.ini) variable QuotePrefix.

The default length for wrapping each line is 70. You can specify a different length with the environment (notes.ini) variable QuoteLineLength.
This command does not work on the Web.

**Examples: EditQuoteSelection**
This form action selects and quotes all the text in the current field.
@Command([EditSelectAll]);
@Command([EditQuoteSelection])

**EditResizePicture @Command**
Displays the proportions of the selected graphic at the bottom of the screen. The user can size the picture by dragging the mouse in the appropriate direction.

**Syntax**
@Command([EditResizePicture])

**Usage**
A document must be open in Edit mode and a graphic must be selected.

This command does not work on the Web.

**EditRestoreDocument @Command**
Performs the menu command Edit - Restore.

**Note:** This function is new with Release 6.

**Syntax**
@Command([EditRestoreDocument])

**Usage**
In a soft deleted document, restores the document to the view or folder it was deleted from.

In a soft deletions view, restores the selected documents to the folders or views from which they were deleted.

To create a soft deletions view:
1. Select File - Database - Properties from the menu bar.
2. On the Advanced tab of the Database Properties box, select Allow soft deletions.
3. From the View design list in Lotus Domino Designer, select the New View button.
4. In the Create View dialog box, enter a name for the view, then choose "Shared, contains deleted documents" as the View type and click Ok.
   This creates a view that holds any documents deleted from other views or folders in the database. To permanently remove a document from the database, you must delete it from this view.

This command does not work on the Web.

**Examples: @Command([EditRestoreDocument])**
1. This code, when used in the Restore action button on a form, enables you to return a document accessed from a soft deletions view called "DeletedDocs" to the view or folder that it was deleted from.
   @Command([EditRestoreDocument])
You may want to add a Hide When formula to the Restore action button so it only displays on the form when the current document is a soft deleted document. Select the Hide action if formula is true check box and enter the following code in the formula window.

@ViewTitle !="DeletedDocs"

2. This code, when added to the Restore action button in a soft deletions view, returns the currently selected documents to the respective views and folders from which they were deleted.

@Command([EditRestoreDocument])

## EditRight @Command

Moves the insertion point in a document to the right by the number of spaces you specify. If you don’t specify a number, moves the insertion point one space to the right.

### Syntax

@Command( [EditRight] ; count )

### Parameters

*count*

Number. Optional. The number of spaces you want to move.

### Usage

- On a form or subform in Design mode, EditRight moves the insertion point one or *count* spaces to the right, as if the user had pressed RIGHT.
- In a document in Edit mode, EditRight moves the insertion point one or *count* spaces to the right within the current field, or if there are no more spaces in the current field, in the next editable field (which may be to the right of, or below, the current field).
- In a document in Read mode, EditRight has no effect.

This command does not work on the Web.

### Language cross-reference

GoToNextField method of LotusScript NotesUIDocument class

### Examples: EditRight

This formula moves the insertion point three spaces to the right.

@Command( [EditRight] ; "3" )

## EditSelectAll @Command

Performs the menu command Edit – Select All.

### Syntax

@Command( [EditSelectAll] )

### Usage

- In a view or folder, all documents are selected.
- In a document in Read mode, all data on the document is selected, including field labels.
- In a document in Edit mode, all data in the current field is selected.
- On a form or subform, selects everything on the form or subform except layout regions.
• On the workspace, all databases on the current tab are selected.
• In a view, folder, or navigator in Design mode, EditSelectAll is invalid.
• This command does not work on the Web.

**Language cross-reference**
SelectAll method of LotusScript NotesUIDocument class

**Examples: EditSelectAll command**
1. This formula, when added to the "Set font size" hotspot button, first sets focus to the "Body" Rich Text field, then applies the font size the user specifies in the "size" text field to the text in the Body field. If no size was selected, an error displays telling the user to enter one.

```lss
@Command([EditGoToField];"Body");
@Command([EditSelectAll]);
result := @Command([TextSetFontSize];size);
@If(@IsError(result);Prompt([Ok];"Error encountered";"You must enter a font size first");result)
```

**EditSelectByDate @Command**
Displays the Select by Date dialog box, where the user indicates which documents should be selected in the view or folder, according to the date they were created or last modified.

**Syntax**
@Command( [EditSelectByDate] )

**Usage**
A view or folder must be open.

This command does not work on the Web.

**Language cross-reference**
UnprocessedSearch method of LotusScript NotesDatabase class
Search method of LotusScript NotesDatabase class
unprocessedSearch method of Java AgentContext class
search method of Java Database class

**EditShowHideHiddenChars @Command**
Toggles the display of the hidden characters (such as spaces, tabs, and carriage returns) in a document, form, or subform.

**Syntax**
@Command( [EditShowHideHiddenChars] ; showOrHide )

**Parameters**
showOrHide

Number. Optional. Specify “1” if you want to show hidden characters, “0” if you want to hide them. If you omit this parameter, EditShowHideHiddenChars toggles the current state of the hidden characters.
Usage
A document must be open in Edit mode or a form or subform must be open in Design mode. Hidden characters are shown only in rich text fields.

This command does not work on the Web.

Language cross-reference
HiddenChars property of LotusScript NotesUIDocument class

EditTableDeleteRowColumn @Command
Displays the Delete Row/Column dialog box, where the user can select the number of rows or columns to delete from the current table.

Syntax
@Command( [EditTableDeleteRowColumn] )

Usage
• A document must be open in Edit mode
  or
• A form, subform, or page must be open in Design mode
• The cursor must be in a table.

This command does not work on the Web.

Language cross-reference
RemoveRow method of LotusScript NotesRichTextTable class
DeleteRow method of ODBCResultSet class

EditTableFormat @Command
Displays the Properties box for the selected table.

Syntax
@Command( [EditTableFormat] )

Usage
• A document must be open in Edit mode
  or
• A form, subform, or page must be open in Design mode
• The cursor must be in a table.

This command does not work on the Web.

Language cross-reference
AppendStyle method of LotusScript NotesRichTextItem class
appendStyle method of Java RichTextItem class

534 Programming Guide, Volume 1: Overview and Formula Language
EditTableInsertRowColumn @Command
Displays the Insert Row/Column dialog box, where the user can select the number of rows or columns to insert into the table.

Syntax
@Command( [EditTableInsertRowColumn] )

Usage
- A document must be open in Edit mode
- A form, subform, or page must be open in Design mode
- The insertion point must be in a table.

This command does not work on the Web.

Language cross-reference
AddRow method of LotusScript NotesRichTextTable class
AddRow method of ODBCResultSet class

EditTop @Command
Moves the insertion point to the top of a document or form.

Syntax
@Command( [EditTop] )

Usage
- On a form in Design mode, EditTop moves the insertion point to the top of the form as if the user had pressed ctrl+home.
- In a document in Edit mode, EditTop moves the insertion point to the first editable field on the document.
- In a document in Read mode, EditTop has no effect.
- You can precede this command with either:
  - @Command([EditDocument]; "1") to put the document into edit mode
  - @IsDocBeingEdited function to test if the document is in edit mode

This command does not work on the Web.

Language cross-reference
GotoTop method in LotusScript NotesUIDocument class

EditUndo @Command
Performs the menu command Edit – Undo, which undoes the previous reversible operation.

Syntax
@Command( [EditUndo] )
Usage
For buttons and actions, Edit Undo works only when the user is at the view level; use it to remove the deletion flag from a document that is marked deleted.

This command does not work on the Web.

EditUntruncate @Command
Retrieves the full version of a truncated document.

Syntax
@Command( [EditUntruncate] )

Usage
In a database which has undergone replication with the option to receive only summary (truncated) documents, EditUntruncate retrieves full versions of the truncated documents that you select.

This command does not work on the Web.

EditUp @Command
Moves the insertion point in a document up by the number of lines you specify. If you don’t specify a number, moves the insertion point up one line.

Syntax
@Command( [EditUp] ; count )

Parameters

\textit{count}

Number. Optional. Specifies the number of lines to move up.

Usage
• On a form in Design mode, moves the insertion point up one or \textit{count} lines, as if the user had pressed up.
• In a document in Edit mode, moves the insertion point up one line within the current field, or if there are no more lines in the current field, in the previous editable field (which must be above, not to the right of, the current field).
• In a document in Read mode, has no effect.
• This command does not work on the Web.

Language cross-reference
GoToPrevField method of LotusScript NotesUIDocument class

Examples: EditUp
This formula moves the insertion point up two lines.
@Command( [EditUp];"2" )
EmptyTrash @Command

Deletes documents marked for deletion in a database and refreshes the view.

Note: This @command is new with Release 5.

Syntax

@Command( [EmptyTrash] )

Usage

This command permanently deletes any documents currently marked for deletion (use MoveToTrash @Command to mark documents for deletion).

You can use this command with Web applications. The View applet is also programmable via this @command.

Language cross-reference

ViewRefresh method of LotusScript NotesUIWorkspace class

Examples: @Command([EmptyTrash])

This code, when added as the formula for the Delete action in a view, permanently deletes all the documents currently marked for deletion in the view.

@Command([EmptyTrash])

ExchangeUnreadMarks @Command

For two selected database replicas, marks the documents as read in one replica that are marked as read in the other.

Syntax

@Command( [ExchangeUnreadMarks] )

Usage

Two database replicas must be selected whose icons are not stacked.

This command does not work on the Web.

Execute @Command

Launches an application.

Syntax

@Command( [Execute] ; application ; fileNames )

Parameters

application

Text. A path and file name specifying the application you want to open.

fileNames
Text or text list. One or more paths and file names specifying the file(s) you want to open in the application. You can specify more than one file, as long as the application can open multiple files at launch time.

**Usage**

Specify the paths and file names using the appropriate format for the operating system.

This command does not work on the Web.

**Language cross-reference**

Shell function of LotusScript language

**Examples: Execute**

1. This formula launches 1–2–3 for Windows.
   
   ```
   @Command( [Execute]; "C:\123W\PROGRAMS\123W.EXE" )
   ```

2. This formula launches 1–2–3 for Windows and loads the worksheet named SALES.WK4.
   
   ```
   @Command( [Execute]; "C:\123W\PROGRAMS\123W.EXE"; "C:\123W\WORK\SALES.WK4" )
   ```

**ExitNotes @Command**

Performs the menu command File – Exit (File – Quit on the Macintosh), which closes Notes/Domino and all its open windows.

**Note:** This @command is new with Release 6.

**Syntax**

```
@Command( [ExitNotes] )
```

**Usage**

This command executes immediately. Use the `FileExit @command` to execute after all @functions. See the "Order of evaluation for formula statements" topic for more details.

This can be used almost anywhere in Notes/Domino except from within a dialog box or on the Web. If an open document or design element has not been saved, Notes prompts the user to save it.

**Language cross-reference**

Close method of LotusScript NotesUIDatabase class

**FileCloseWindow @Command**

Closes the current Notes window. If the document or design element in that window has not been saved, Notes prompts the user to save it before closing.

**Syntax**

```
@Command( [FileCloseWindow] )
```

**Usage**

This command executes after all @functions. Use `@Command([CloseWindow])` to execute immediately. See the "Order of evaluation for formula statements" topic for more details.

FileCloseWindow does not close the Notes workspace window.
When using this command on a form in Notes, you can prevent the user from being prompted to save any changes.

You can use this command with Web applications, as long as you enable the "Allow Javascript on the Web" setting on the Basics tab of the Database Properties box. Precede this command with `@Command([FileSave])` to simulate a Submit button.

**Language cross-reference**
Close method of LotusScript NotesUIView class

---

**FileDatabaseACL @Command**
Displays the access control list for the current database.

**Syntax**
`@Command( [FileDatabaseACL] )`

**Usage**
A database must be open or selected on the workspace. Although any user with access greater than "No access" can view the ACL, only users with Manager access can edit the ACL.

This command does not work on the Web.

**Language cross-reference**
ACL property in LotusScript NotesDatabase class

---

**FileDatabaseCompact @Command**
Compacts the white space in a database, and at the same time, converts the old database to the new format.

**Syntax**
`@Command( [FileDatabaseCompact] )`

**Usage**
A database must be open, or selected on the workspace. No access level is required to compact a database.

This command does not work on the Web.

**Language cross-reference**
Compact method of LotusScript NotesDatabase class
compact method of Java Database class

**Examples: @Command([FileDatabaseCompact])**
This formula, when added as the code in an action button on a form in the TESTING.NSF database, compacts the TESTING.NSF database when you click the action button.

`@Command([FileDatabaseCompact])`
A message box displays indicating that the database is now being compacted. After you click the OK button, the status bar indicates when the database has been compacted by displaying the message, "Finished compacting testing."

---

**FileDatabaseCopy @Command**

Displays the Copy Database dialog box, where the user can copy the current database to a new location.

**Syntax**

```
@Command( [FileDatabaseCopy] )
```

**Usage**

A database must be open, or selected on the workspace. The user must have at least Reader access to the database.

This command does not work on the Web.

**Language cross-reference**

CreateCopy method of LotusScript NotesDatabase class

createCopy method of Java Database class

---

**FileDatabaseDelete @Command**

Permanently deletes the current database file from the hard disk where it is stored.

**Syntax**

```
@Command( [FileDatabaseDelete] )
```

**Usage**

This command executes after all @functions. Use `@Command([DatabaseDelete])` to execute immediately. See the [Order of evaluation for formula statements] topic for more details.

A database icon must be selected, but the database cannot be open. The user must have Manager access in order to delete the database.

**Note:** To remove the database icon from the user's workspace without deleting the database, use `@Command([FileDatabaseRemove])`.

This command does not work on the Web.

**Language cross-reference**

Remove method of LotusScript NotesDatabase class

remove method of Java Database class

---

**FileDatabaseInfo @Command**

Displays the Properties box for the current database.

**Syntax**

```
@Command( [FileDatabaseInfo] )
```
Usage
A database must be open, or selected on the workspace. Any user with at least Depositor access to the database can view the information.

This command does not work on the Web.

Language cross-reference
LotusScript NotesDatabase class
Java Database class

FileDatabaseRemove @Command
Removes the current database icon from the workspace.

Syntax
@Command( [FileDatabaseRemove] )

Usage
The database icon must be selected, but the database cannot be open.

This command does not work on the Web.

Note: To delete the database file from the hard disk where it’s stored, use FileDatabaseDelete or DatabaseDelete.

FileDatabaseUseServer @Command
Displays the Switch Servers dialog box, where the user can choose a server that contains a replica of the currently selected database.

Syntax
@Command( [FileDatabaseUseServer] )

Usage
The command must be run from the workspace with a database selected.

This command does not work on the Web.

Language cross-reference
OpenWithFailover method of LotusScript NotesDatabase class

FileExit @Command
Performs the menu command File – Exit (File – Quit on the Macintosh), which closes Notes/Domino and all its open windows.

Syntax
@Command( [FileExit] )
Usage
This command executes after all @functions. Use [@Command([ExitNotes]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

This can be used almost anywhere in Notes/Domino except from within a dialog box or on the Web. If an open document or design element has not been saved, Notes prompts the user to save it.

Language cross-reference
Close method of LotusScript NotesUIDatabase class

FileExport @Command
Exports a Notes/Domino document or view.

Syntax
@Command( [FileExport] )

or

@Command( [FileExport] ; fileType ; fileName )

Parameters
fileType
Text. The kind of file you want to export to. See list of file types, below.

fileName
Text. The name of the file you want to export to. Must be a complete path specification, including drive, directory, and file name.

If the fileType and fileName parameters are omitted, Notes/Domino displays the Export dialog box. If the parameters are included, Notes/Domino exports the view or currently opened document using the specified fileType and fileName.

Return value
Number. If the user clicks the Cancel button on the Export dialog box, the number 1 is returned.

Usage
FileExport can be used at the view level, and when a document is open in Read or Edit mode, according to what is being exported.

This command does not work on the Web.

The fileType must be one of those listed below. You do not have to spell the name exactly as shown; you have to include enough characters to uniquely identify the file type. In case of ambiguity, Notes/Domino will use the first file type in the list that matches your entry.

Windows File Types
Document Level ASCII Text CGM Image Microsoft RTF* TIFF 5.0 Image vCard 3.0**

View Level Lotus 1-2-3* Structured Text Tabular Text vCard 3.0**
Macintosh File Types

**Document Level ASCII Text Microsoft RTF* TIFF 5.0 Image CGM Image vCard 3.0**

*Note:* The Macintosh does not support document level export of the XTND file format (MacWrite II and Text).

**View Level** Lotus 1-2-3* Structured Text Tabular Text

*The formatting of the names of these file types changed with R5.0.5. If you installed (not upgraded to) R5.0.5 or later, pre-5.0.5 scripts that reference these file types using the old name formats may not work.

**New with Release 6.**

*Note:* With R4.0, Notes does not support exporting ANSI Metafile file types.

*Note:* With Release 6, Notes does not support exporting UNIX or Lotus Ami Pro file types.

*Note:* With Release 7, Notes does not support exporting Microsoft Word or WordPerfect file types.

**Examples: FileExport**

1. This formula exports a view to c:\temp.txt as tabular text.
   `@Command([FileExport]; "Tabular Text"; c:\temp.txt)`

2. This formula exports the current view to a 1–2–3 for Macintosh worksheet called Hardware and stores the file in the Lotus 1–2–3 folder on the user's workstation.
   `@Command([FileExport];"Lotus 1-2-3";"SBRAUN:Lotus 1–2–3:Hardware")`

3. This agent formula exports the current document to the testing.txt file on the C drive of the local machine when the user triggers the agent from the Action menu.
   `@Command([FileExport];"ASCII";"C:\testing.txt")`

   This agent has an event trigger of "Action view selection" and has "None" selected as its target.

**FileFullTextCreate @Command**

Displays the Full-Text Create Index dialog box, where the user can specify settings for the database's full-text index.

**Syntax**

`@Command([FileFullTextCreate])`

**Usage**

This @command takes no parameters.

A database must be open, or selected on the workspace, and the user must have at least Designer access.

This command does not work on the Web.

**Language cross-reference**

UpdateFTIndex method of LotusScript NotesDatabase class

updateFTIndex method of Java Database class
FileFullTextDelete @Command
Deletes a database’s full-text index.

Syntax
@Command( [FileFullTextDelete] )

Usage
A database must be open or selected on the workspace, and must have a full-text index already created. The user must have at least Designer access to the database.

This command does not work on the Web.

FileFullTextInfo @Command
Displays the Full Text tab of the current database’s Properties box.

Syntax
@Command( [FileFullTextInfo] )

Usage
A database must be open, or selected on the workspace.

This command does not work on the Web.

Language cross-reference
LastFTIndexed property of LotusScript NotesDatabase class
lastFTIndexed property of Java Database class

FileFullTextUpdate @Command
Updates full-text indexes for local databases or queues the update request for server–based databases.

Syntax
@Command( [FileFullTextUpdate] )

Usage
A database must be open, or selected on the workspace.

This command does not work on the Web.

Language cross-reference
UpdateFTIndex method of LotusScript NotesDatabase class
updateFTIndex method of Java Database class

FileImport @Command
Imports a file into a Notes/Domino document or view.
Syntax
@Command( [FileImport] ; fileType ; fileName )

Parameters

fileType
Text. The kind of file you want to import. See list of file types, below.

fileName
Text. The name of the file you want to import. Must be a complete path specification, including drive, directory, and file name.

If the fileType and fileName parameters are omitted, this displays the Import dialog box. If the parameters are included, Notes/Domino imports the specified file into the currently open document, or into a view.

Usage

FileImport can be used at the view level, and when a document is open in Edit mode, according to what is being imported. If the document is being edited, the insertion point must be in a rich text field.

This command does not work on the Web.

The fileType must be one of those listed below. You do not have to spell the name exactly as shown; you have to include enough characters to uniquely identify the file type. In case of ambiguity, Notes/Domino will use the first file type in the list that matches your entry.

Windows File Types


View Level Lotus 1-2-3* Structured Text Tabular Text vCard**

Macintosh File Types

Document Level Lotus 1-2-3* ASCII Text Microsoft RTF* TIFF 5.0 Image CGM Image GIF Image JPEG Image Binary with Text HTML File

Note: The Macintosh does not support document-level import of the XTND file format (MacWrite II and Text).

View Level Lotus 1-2-3* Structured Text Tabular Text vCard**

*The formatting of the names of these file types changed with R5.0.5. If you installed (not upgraded to) R5.0.5 or later, pre-5.0.5 scripts that reference these file types using the old name formats may not work.

**New with Release 6.

Note: With R4.0, Notes does not support importing ANSI Metafile file types.

Note: With Release 6, Notes does not support importing UNIX or Ami Pro file types.

Language cross-reference

Import method in LotusScript NotesUIDocument class
Examples: FileImport

1. This formula imports a GIF image from the Notes directory into the current document.
   @Command([FileImport]; "GIF Image"; "c:\\notes32\\sound.gif")

2. This formula imports a 1–2–3 for Macintosh worksheet called Hardware from the Lotus 1–2–3 folder on the user’s workstation into the current view.
   @Command([FileImport]; "Lotus 1-2-3"; "SBRAUN:Lotus 1–2–3:Hardware")

FileNewDatabase @Command

Displays the New Database dialog box, where the user can select a server, title, and file name for a new database.

Syntax
   @Command( [FileNewDatabase] )

Usage
This command does not work on the Web.

Language cross-reference
Create method of LotusScript NotesDatabase class

CreateFromTemplate method of LotusScript NotesDatabase class

createFromTemplate method of Java Database class

FileNewReplica @Command

Displays the New Replica dialog box, where the user can create a replica of the current database.

Syntax
   @Command( [FileNewReplica] )

Usage
This command does not work on the Web.

Language cross-reference
CreateReplica method of LotusScript NotesDatabase class

createReplica method of Java Database class

FileOpenDatabase @Command

Opens the specified database to the specified view, highlighting the first document whose value in the sort column matches the key. You specify a database using its server and file name.

Syntax
   @Command( [FileOpenDatabase]; server : database ; viewName ; key ; newinstance ; temporary )

or

   @Command( [FileOpenDatabase]; server : database ; navigator ; solo ; newinstance ; temporary )
Parameters

*server*

Text. The name of the server where the database resides.

*database*

Text. Optional. The path and file name (in the appropriate format for the operating system) of the database you want to open. If you omit the name of the database, and one is already open, Notes/Domino displays that database’s view in the topmost, or current, window. If no database is open but one is selected on the workspace, the selected database is opened. If no databases are open or selected, Notes/Domino displays the Open Database dialog box, so the user can select a database.

$viewName$

Text. Optional. The name of the view you want to open in the database. If you omit the name of the view, the database opens to its default view; or, if the user has opened the database before, to the last view used by that user.

*key*

Text. Optional. Indicates which document you want Notes to scroll to when it opens $viewName$. The *key* is a value that appears in the first sorted column of $viewName$. If you omit the key, no document is selected.

*newinstance*

Number. Optional. Specify “1” if you want the view to open in a new window, even if the database is already open in that view. If you omit this parameter, the new window is opened only if the database is opened in a new view.

*temporary*

Number. Optional. Specify “1” if you only want the database opened on a temporary basis for browsing, without adding the database to the user’s workspace. If you omit this parameter, the database is added to the user’s workspace.

*navigator*

Text. Optional. The name of a navigator defined for the database. On opening the database, Notes displays this navigator.

*solo*

Number. Optional. A value of “1” instructs Notes/Domino to open the navigator named by $navigator$ in its own window.

**Usage**

If you omit some of the parameters like $viewName$ and $key$, but still include the parameters that come after it, substitute “” for each parameter that you’re skipping, as in:

```
@Command([FileOpenDatabase]; "Sales":"PROBLEMS.NSF"; ""; ""; ""; "1")
```

You can use this command in Web applications, but the *server* argument must be specified as a null string (“”). When using the $view$ argument, you must follow the command with @Command [OpenDocument].
Language cross-reference

GetDatabase method of LotusScript NotesSession class

GetDocumentByKey method of LotusScript NotesView class

OpenDatabase method of LotusScript NotesUIWorkspace class

getDatabase method of Java Session class

getDocumentByKey method of Java View class

Examples: FileOpenDatabase

1. Notes/Domino opens the PROBLEMS.NSF database, which is stored on the Development server. If the database is already open, Notes/Domino brings that window to the foreground. If the database is not open, Notes/Domino opens it in a new window. Since no view was specified, the database opens either to the default view, or if the user has opened the database before, to the view last used by the user. If the database is not on the user’s workspace, it will be added automatically.

   @Command([FileOpenDatabase];"Development":"PROBLEMS.NSF")

2. Notes/Domino opens the PROBLEMS.NSF database on the Development server. Additionally, Notes/Domino opens a new window, displays the Open Problem Reports view, and highlights the first document containing "Printer problems" in the key field. Since the temporary parameter is used, the database icon is not added to the user’s workspace.

   @Command([FileOpenDatabase]; "Development":"PROBLEMS.NSF"; "Open Problem Reports"; "Printer problems"; "1"; "1")

3. This code, when added to the Open My Feedback hotspot button returns the All\By Employee view of the UserNotes.nsf database located in the feedback subdirectory of the Customers/ME/ACME server. The view, which contains documents grouped by category, where the categories are employee names appears with the category named after the current user highlighted and displayed in the first row of the view.

   @Command([FileOpenDatabase]; "Customer/ME/ACME":"feedback\UserNotes.nsf"; "All\By Employee"; @Name([CN];@UserName(0)); "1"; "1")

4. This formula, when added to the “Int’l” hotspot button on a document whose design is based on the USUser form, opens the Int’l view. If the Int’l view’s form formula is "IntlUser," the user can select the same document from this view and it will display using the IntlUser form. If you do not close the current USUser-based document first, it will redisplay based on the USUser form when selected from the Int’l view.

   @Command([CloseWindow]);
   @Command([FileDatabaseOpen];"AcmeServer/Central","Users.nsf","Int’l")

FileOpenDBRepID @Command

Opens the specified database to the specified view, highlighting the first document whose value in the sort column matches the key. You specify a database using its replica ID, and Notes/Domino searches the workspace and all servers available in the current session to find a replica.

Syntax

@Command( [FileOpenDBRepID] ; replicaID ; serverHint ; viewName ; key ; newInstance ; temporary )

or

@Command( [FileOpenDBRepID] ; replicaID ; serverHint ; navigator ; solo ; newInstance ; temporary )
Parameters

replicaID

Text. The replica ID of the database you want to open. You can see the replica ID for a database in its InfoBox.

serverHint

Text. Optional. The name of a server where the database might reside. Notes/Domino checks this server for the replica before checking other servers only if the database was not accessible from a workspace icon.

viewName

Text. Optional. The name of the view that you want to open in the database. If you omit the name of the view, the database opens to its default view, or if the user has opened the database before, to the last view used by that person.

key

Text. Optional. Indicates which document you want Notes/Domino to scroll to when it opens viewName. The key is a value that appears in the first sorted column of viewName. If you omit the key, no document is selected.

newInstance

Number. Optional. Specify "1" if you want the view to always open in a new window, even if there’s already a window open for the database. Specify an empty string ("") if you want the new window opened only when it’s actually needed.

temporary

Number. Optional. Specify "1" if you want the database opened on a temporary basis for browsing, without adding the database to the user’s workspace. Specify an empty string ("") if you want to add the database to the user’s workspace.

navigator

Text. Optional. The name of a navigator defined for the database. On opening the database, Notes displays this navigator.

solo

Number. Optional. A value of "1" instructs Notes to open the navigator named by navigator in its own window.

Usage

FileOpenDBRepID is useful whenever you want to write a formula for users working on several different replicas of the same database. Libraries, for example, use this command to open a database from the library.

FileOpenDBRepID looks for the replica in the following order:

- Checks the user’s workspace for an icon with the specified replica ID.
- Checks the server indicated in the serverHint for the specified replica ID.
Checks other home/mail servers specified in the location document for the specified replica ID.

This command does not work on the Web.

**Language cross-reference**

OpenByReplicaID method of LotusScript NotesDatabase class

**Examples: FileOpenDBRepID**

This formula opens the first database that Notes/Domino finds with the replica ID 85255F1E:004F2CEB, to the All by Category view.

```
@Command([FileOpenDBRepID]; "85255F1E:004F2CEB";
    ""; "All by Category"
)
```

**FilePageSetup @Command**

Displays the Page Setup dialog box (or, on the Macintosh, the File Print Margins dialog box), which allows you to specify print settings for the selected database.

**Syntax**

@Command( [FilePageSetup] )

**Usage**

A database must be selected or open.

This command does not work on the Web.

**Language cross-reference**

Print method in LotusScript NotesUIDocument class

Print method in LotusScript NotesUIView class

**FilePrint @Command**

Prints the currently open or selected document(s), or the current view.

**Syntax**

@Command( [FilePrint] )

or

@Command( [FilePrint] ; numCopies ; fromPage ; toPage ; ifDraft ; ifView ; formName ; breakType ; ifReset; startDate; endDate )

**Parameters**

*numCopies*  
Text. Optional. The string must evaluate to a number, or be empty. The number of copies you want to print. Specify an empty string ("") for one copy.

*fromPage*  

Text. Optional. The string must evaluate to a number, or be empty. The page of a document where you want to start printing. Specify an empty string (""") if you want to print all pages.

*toPage*

Text. Optional. The string must evaluate to a number, or be empty. The page of a document where you want to stop printing. Specify an empty string (""") if you want to print all pages.

*ifDraft*

Text. Optional. Either the word "draft", to indicate that you want draft quality printing; or an empty string (""), if you want regular quality printing.

*ifView*

Text. Optional. Either the word "printview", to indicate that you want to print the current view; or an empty string (""), if you want to print the selected document(s) in a view, not the view itself. This parameter is ignored if you're printing from an open document.

*formName*

Text. Optional. The name of the form you want to use to print the document. Specify an empty string (""") if you want to print the document using its current form. This parameter is ignored if you're printing from an open document.

*breakType*

Text. Optional. Either the word "pagebreak", the word "line", or an empty string (""). If you're printing multiple documents from a view, "pagebreak"indicates you want a page break between each document, "line" indicates you want a ruled line between each document, and an empty string indicates you want a blank line between each document. This parameter is ignored if you're printing from an open document.

*ifReset*

Text. Optional. Either the word "resetpages", or an empty string (""). If you're printing multiple documents from a view, then "resetpages" specifies that page numbering begins at 1 with each new document; and an empty string ("") specifies that page numbering begins at 1 with the first document and continues cumulatively. This parameter is ignored if you're printing from an open document.

*startDate*

Time-date. Optional. Used with the printview parameter when printing a calendar view; indicates the first date to be printed. Specify an empty string (""") if you want to start printing with the earliest date in the view.

*endDate*

Time-date. Optional. Used with the printview parameter when printing a calendar view; indicates the last date to be printed. Specify an empty string (""") if you want to end printing with the last date in the view.

**Usage**

With no parameters, FilePrint displays the File Print dialog box (on the Macintosh, the chosen printer’s dialog box). With parameters, FilePrint prints the current document(s) or view without displaying the dialog box.
In Notes Release 3, FilePrint displays the File Print dialog box whether you use parameters or not.

This @command does not work on the Web. Use the browser’s print button to print the current document or view. All hide-when formulas set for elements in the form or view are invoked by the browser.

**Language cross-reference**
Print method in LotusScript NotesUIDocument class

Print method in LotusScript NotesUIView class

**Examples: FilePrint @Command**
1. The following code, when added to an action button in a view, prints the entire view:
   ```lotusscript
   @Command([FilePrint];"";"";"";"";"printview")
   ```
2. This code, when added to an action button in a view, prompts users for the start and end dates of documents they want to print and then prints only documents with creation or modification dates within that specified range:
   ```lotusscript
   @Command([EditSelectByDate]);
   @Command([FilePrint];"";"";"";"";"pagebreak")
   ```
3. This code prints the selected documents from a view in landscape format. Add this code to an agent that has Action menu selection chosen as the trigger and None as the target.
   ```lotusscript
   @Command([FilePrint])
   When the Print View dialog box displays, select landscape as the orientation on the Page Setup tab, then press Ok.
   ```

**FilePrintSetup @Command**
Displays the Print Setup dialog box, which allows you to direct the current view or document to a printer or a file that you specify.

**Note:** [FilePrintSetup] is not supported on the Macintosh. To make setup changes, use [FilePrint]; to select different printers, use the Chooser.

**Syntax**
```lotusscript
@Command( [FilePrintSetup] )
```
Usage
A document must be open in Edit mode.

A form, subform, view, folder, agent, or navigator must be open in Design mode.

You can use this command with Web applications, as long as you enable the "Web access: Use JavaScript when generating pages" setting on the Basics tab of the Database Properties box. Follow this command with `@Command([FileCloseWindow])` or `@Command([CloseWindow])` to simulate a Submit button.

Language cross-reference
Save method of LotusScript NotesDocument class

Save method of LotusScript NotesUIDocument class

save method of Java Document class

Examples: FileSave
1. This code, when added to the "Next" action button in form1 and triggered from the Notes client, saves and closes form1 and opens form2 in edit mode.
   
   ```livescript
   dbname := @DbName;
   @Command([FileSave]);
   @Command([CloseWindow]);
   @Command([Compose]; dbname; "form2")
   ```

2. This code, when added to the Save action button in a form and triggered from Notes, saves the current document and opens the AllDocs view when the save is successful. If the save is not successful, it returns the message, "Save unsuccessful," to the status bar.
   
   ```livescript
   @If(@Command([FileSave]);@Do(@Command([CloseWindow]);@Command([OpenView];"AllDocs");@StatusBar ("Save unsuccessful"))
   ```

FileSaveNewVersion @Command
Saves a document as a new version.

Syntax
```livescript
@Command( [FileSaveNewVersion] )
```

Usage
A document must be open in Edit mode.

This command does not work on the Web.

Language cross-reference
SaveNewVersion method in LotusScript NotesUIDocument class

FindFreeTimeDialog @Command
Opens the Free Time dialog box to allow searches for available meeting times.

Syntax
```livescript
@Command( [FindFreeTimeDialog]; reqPeopleItems; optPeopleItems; reqRoomsItems ; optRoomsItems; reqResourcesItems; optResourcesItems; removedPeopleItems; startDateTime; endDateTime )
```
Parameters

\textit{reqPeopleItems}

Text or text list. Names of fields (items) in the current document that contain names of people required at the meeting.

\textit{optPeopleItems}

Text or text list. Optional. Names of fields (items) in the current document that contain names of people whose attendance is optional.

\textit{reqRoomsItems}

Text or text list. Optional. Names of fields (items) in the current document that contain names of rooms required for the meeting.

\textit{optRoomsItems}

Text or text list. Optional. Names of fields (items) in the current document that contain names of optional rooms.

\textit{reqResourcesItems}

Text or text list. Optional. Names of fields (items) in the current document that contain names of resources required for the meeting.

\textit{optResourcesItems}

Text or text list. Optional. Names of fields (items) in the current document that contain names of optional resources.

\textit{removedPeopleItems}

Text or text list. Optional. Names of fields (items) in the current document that contain names of people to remove from the attendance list.

\textit{startTime}

Text. Name of a Time field (item) in the current document that contains the start time for the meeting. You must specify the start date time.

\textit{endTime}

Text. Name of a Time field (item) in the current document that contains the end time for the meeting. You must specify the end date time.

Usage

- If you don’t need a parameter, use consecutive quotes (“”) to omit it.
- The meeting time should not span midnight.
- The user can adjust the values specified here through the dialog box.

This command does not work on the Web.
Language cross-reference
FindFreeTimeDialog method of LotusScript NotesUIDocument class

Examples: FindFreeTimeDialog
This @command opens the Free Time dialog box with required people from the From and SendTo fields in the current document, and times from the StartDateTime and EndDateTime fields.

@Command([ViewRefreshFields]);
@Command([FindFreeTimeDialog]; "From" : "SendTo"; "" ; "" ; "" ; "" ; "StartDateTime"; "EndDateTime")

Folder @Command
Moves or copies the selected document to a folder.

Syntax
@Command([Folder] ; folderName ; moveOrCopy)

Parameters
folderName
Text. Optional. The name of the folder to which you want to move or copy the selected document. If you omit this parameter, Notes/Domino displays the Move to Folder dialog box, where you can choose a folder.

moveOrCopy
Number ("1" or "0"). Optional. A value of "1" moves the document to the folder. A value of "0" copies the document to the folder. If you omit this parameter, Folder assumes a value of "0" (copy).

Usage
This command executes after all @functions. Use the FolderDocuments @Command to execute immediately. See the Order of evaluation for formula statements topic for more details.

If you don’t want to specify a folderName, but you want to specify a moveOrCopy, use a NULL string as shown below.
@Command([Folder];"";"1")

A saved document must be open or selected in a view.

Folder does not work for new documents. If multiple documents are selected in a view, they are all moved or copied to the folder.

@AddToFolder works just like @Command([Folder]; Foldername; MoveOrCopy) except it can move a document from another folder.

Note: The following feature is new with Release 5.

This @command works on the Web if “Use applet in the browser” is in effect for the implementing view or folder.
Language cross-reference
Folder method of LotusScript NotesUIWorkspace class
PutInFolder method of LotusScript NotesDocument class
putInFolder method of Java Document class

FolderCollapse @Command
For a folder or view containing nested folders or views, collapses the selected folder or view in the navigation pane.

Syntax
@Command( [FolderCollapse] )

Usage
A database must be open to a view or folder. Focus must be in the navigation pane.
This command does not work on the Web.

FolderCustomize @Command
Displays the design pane for the currently selected view or folder.

Syntax
@Command( [FolderCustomize] )

Usage
A database must be open to a view or folder.
This command does not work on the Web.

FolderDocuments @Command
Moves or copies the selected document to a folder.

Note: This command is new with Release 6.

Syntax
@Command( [FolderDocuments] ; folderName ; moveOrCopy )

Parameters
folderName
Text. Optional. The name of the folder to which you want to move or copy the selected document. If you omit this parameter, Notes/Domino displays the Move to Folder dialog box, where you can choose a folder.

moveOrCopy
Number ("1" or "0"). Optional. A value of "1" moves the document to the folder. A value of "0" copies the document to the folder. If you omit this parameter, Folder assumes a value of "0" (copy).
If you include moveOrCopy with a value of "0" but omit folderName, Notes/Domino displays the Move to Folder dialog box with the Move button dimmed. If you include moveOrCopy with a value of "1" but omit folderName, Notes/Domino displays the Move to Folder dialog box with all of its options available.

**Usage**

This command executes immediately. Use the **Folder @command** to execute after all @functions. See the **Order of evaluation for formula statements** topic for more details.

If you don't want to specify a folderName, but you want to specify a moveOrCopy, use a NULL string as shown below.

@Command([FolderDocuments];"";"1")

A saved document must be open or selected in a view.

Folder does not work for new documents. If multiple documents are selected in a view, they are all moved or copied to the folder.

@AddToFolder works just like @Command([FolderDocuments]; Foldername; MoveOrCopy) except it can move a document from another folder.

**Note:** The following feature is new with Release 5.

This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder.

**Language cross-reference**

Folder method of LotusScript NotesUIWorkspace class

PutInFolder method of LotusScript NotesDocument class

putInFolder method of Java Document class

---

**FolderExpand @Command**

Expands the currently selected view or folder one level in the navigation pane.

**Syntax**

@Command( [FolderExpand] )

**Usage**

A database must be open to a view or folder. Focus must be in the navigation pane.

This command does not work on the Web.

---

**FolderExpandAll @Command**

Fully expands all views and folders in the navigation pane.

**Syntax**

@Command( [FolderExpandAll] )
**Usage**

A database must be open to a view or folder. Focus must be in the navigation pane.

This command does not work on the Web.

---

**FolderExpandWithChildren ** **@Command**

Fully expands the selected view or folder in the navigation pane.

**Syntax**

@Command( [FolderExpandWithChildren] )

**Usage**

A database must be open to a view or folder. Focus must be in the navigation pane.

This command does not work on the Web.

---

**FolderMove ** **@Command**

Displays the Move dialog box for a view or folder, which allows you to move the selected view or folder.

**Syntax**

@Command( [FolderMove] )

**Usage**

A database must be open to a view or folder.

To move documents to a folder, see the `FolderDocuments`, `Folder`, `ChooseFolders` @commands or the `@AddToFolder` function.

This command does not work on the Web.

---

**FolderProperties ** **@Command**

Displays the Properties box for a view or folder.

**Syntax**

@Command( [FolderProperties] )

**Usage**

A view or folder must be open in Design mode. Focus must be in the navigation pane.

This command does not work on the Web.

---

**Language cross-reference**

LotusScript `NotesView` class

Java `View` class

---

**FolderRename ** **@Command**

Displays the Rename dialog box, which allows you to rename the selected folder or view.
**Syntax**

@Command( [FolderRename] )

**Usage**

A database must be open to a view or folder.

This command does not work on the Web.

**Language cross-reference**

Name property of LotusScript NotesView class

---

**FormActions @Command**

Displays the actions pane for the current form, subform, or Page. The command is a toggle; selecting it again hides the actions pane.

**Syntax**

@Command( [FormActions] )

**Usage**

A form, subform, or Page must be open in Design mode.

This command does not work on the Web.

---

**FormTestDocument @Command**

Creates a document using the current form or Page, so you can test its fields, formulas, and scripts.

**Syntax**

@Command( [FormTestDocument] )

**Usage**

A form or Page must be open in Design mode. This command does not work for subforms. This command does not work on the Web.

---

**GoUpLevel @Command**

Displays the view containing the current document when the user closes that document.

**Syntax**

@Command( [GoUpLevel] )

**Usage**

If you open a document by activating a link, use GoUpLevel to display the view in which that document appears rather than close the database when you close the document.

This command does not work on the Web.

**Language cross-reference**

CurrentView property of LotusScript NotesUIWorkspace class
**HelpAboutDatabase @Command**

Displays the About This Database or database policy document for the current database (which typically explains the purpose of the application, as well as its intended audience).

**Syntax**

@Command( [HelpAboutDatabase] )

**Usage**

This can be used anywhere in Notes/Domino, provided a database is open or selected on the workspace.

For more on creating About This Database documents, see Creating About and Using documents for a database in the *Application Development with Domino Designer* guide.

This command does not work on the Web.

---

**HelpAboutNotes @Command**

Displays the Notes splash screen that appears when you launch Notes/Domino. The screen displays the current release number and date.

**Syntax**

@Command( [HelpAboutNotes] )

**Usage**

This command does not work on the Web.

---

**HelpUsingDatabase @Command**

Displays the Using This Database document for the current database. The Using This Database document typically provides pointers on how to use an application's forms and views.

For more information on database help documents, see "Creating About and Using documents for a database" in the *Application Development with Domino Designer* book.

**Syntax**

@Command( [HelpUsingDatabase] )

**Usage**

This can be used anywhere in Notes/Domino, provided a database is open or selected on the workspace. This command does not work on the Web.

---

**HotspotClear @Command**

Removes a hotspot, without deleting the underlying text or graphic.

**Syntax**

@Command( [HotspotClear] )
**Usage**

- A document must be open in Edit mode
  
or
- A form or subform must be open in Design mode.
- This `@command` does not work on hotspots that are on a picture. It will only work on the hotspots that are created via the Create-Hotspot menu item.

This command does not work on the Web.

**Language cross-reference**

Remove method of LotusScript NotesEmbeddedObject class

remove method of Java EmbeddedObject class

---

**HotspotProperties @Command**

Displays the Properties box for the current hotspot.

**Syntax**

```java
@Command( [HotspotProperties] )
```

**Usage**

- A document must be open in Edit mode
  
or
- A form or subform must be open in Design mode.
- The cursor must be within a hotspot.

This command does not work on the Web.

**Language cross-reference**

LotusScript NotesEmbeddedObject class

Java EmbeddedObject class

---

**InsertSubform @Command**

Displays the Insert Subform dialog box, where you can select a subform to be inserted on a form.

**Syntax**

```java
@Command( [InsertSubform] )
```

**Usage**

A form must be open in Design mode. There must be at least one subform in the database.

This command does not work on the Web.

---

**LayoutAddGraphic @Command**

Adds a graphic from the Clipboard to a layout region.
**Syntax**

@Command( [LayoutAddGraphic] )

**Usage**

A form or subform must be open in Design mode, a layout region must be selected, and the graphic you want to add must be on the Clipboard.

This command does not work on the Web.

---

**LayoutAddText @Command**

Creates a static text box in a layout region.

**Syntax**

@Command( [LayoutAddText] )

**Usage**

A form or subform must be open in Design mode and a layout region must be selected.

This command does not work on the Web.

---

**LayoutElementBringToFront @Command**

Brings the selected layout element to the front, which means it displays on top of any other layout elements that overlap it.

**Syntax**

@Command( [LayoutElementBringToFront] )

**Usage**

A form or subform must be open in Design mode, and an element within a layout region must be selected.

This command does not work on the Web.

---

**LayoutElementProperties @Command**

Displays the Properties box for the currently selected layout element.

**Syntax**

@Command( [LayoutElementProperties] )

**Usage**

A form or subform must be open in Design mode, and an element within a layout region must be selected.

This command does not work on the Web.

---

**LayoutElementSendToBack @Command**

Sends the selected layout element to the back, which means it displays underneath any other layout elements that overlap it.
Syntax
@Command( [LayoutElementSendToBack] )

Usage
A form or subform must be open in Design mode and an element within a layout region must be selected.

This command does not work on the Web.

---

**LayoutProperties @Command**

Displays the Properties box for the currently selected layout region.

Syntax
@Command( [LayoutProperties] )

Usage
A form or subform must be open in Design mode and a layout region (or an element within it) must be selected.

This command does not work on the Web.

---

**MailAddress @Command**

Displays the Mail Address window, where the user can select people and groups to include in an address field of a mail document.

Syntax
@Command( [MailAddress] )

Usage
For MailAddress to execute successfully, a mail document must be open in Edit mode, and the insertion point must be in an editable field.

The dialog box that displays is a dialog resource built into the core Notes program components, but it is based on the ($PeopleGroupsFlat) view in the local Address book or the server directory, depending on which is selected for use.

This command does not work on the Web.

**Examples: @Command([MailAddress])**

This formula, when added to the "Address" action button on the "Memo" form in a mail database, displays the Select Addresses dialog box when an editable field is selected and the "Address" action button is clicked. A user can select addresses from his or her address books and Notes populates the editable field with the addresses selected.

@Command([MailAddress])

---

**MailComposeMemo @Command**

Creates and displays a blank mail document. Notes/Domino uses the default form for the user’s mail database. This is a Memo document unless the user has changed it.
Syntax
@Command( [MailComposeMemo] )

Usage
This command can be used almost anywhere in Notes/Domino except from within a dialog box or on the Web.

Examples: @Command([MailComposeMemo])
1. This formula, when used in a hotspot button named Create Memo, opens a new mail memo in edit mode.
   @Command([MailComposeMemo])

MailForward @Command
Forwards the current document by placing its contents into a mail memo, which the user then addresses and sends like any other mail memo.

Syntax
@Command( [MailForward] )

Usage
A document must be open in Read or Edit mode, or selected in a view or folder. If multiple documents are selected, the contents of each selected document are placed into the mail memo.

This command does not work on the Web.

Language cross-reference
Forward method in LotusScript NotesUIDocument class

MailForwardAsAttachment @Command
In cc:Mail, forwards a Notes/Domino document as a cc:Mail attachment.

Syntax
@Command( [MailForwardAsAttachment] )

Usage
This command does not work on the Web.

MailOpen @Command
Opens the user’s mail database to the view or navigator to which it was most recently open.

Syntax
@Command( [MailOpen] )

Usage
This command does not work on the Web.
**Language cross-reference**
OpenMail method in LotusScript NotesDatabase class

OpenMailDatabase method in LotusScript NotesDbDirectory class

**MailRequestCrossCert @Command**
Displays the Choose ID to be Cross-Certified dialog box, which allows an administrator to send a safe copy of a Certifier ID to another administrator, who can certify it with a different Certifier ID to create a cross certificate.

**Syntax**
@Command( [MailRequestCrossCert] )

**Usage**
This can be used almost anywhere in Notes/Domino except from within a dialog box or on the Web.

**Language cross-reference**
CrossCertify method of LotusScript NotesRegistration class
crossCertify method of Java Registration class

**MailRequestNewName @Command**
Displays the Change User Name dialog box, which allows the user to send a portion of the Notes user ID to a Notes/Domino administrator, who can change the user name associated with the ID and return the ID to the user. The user then merges the updated portion back into the existing user ID.

**Syntax**
@Command( [MailRequestNewName] )

**Usage**
This can be used anywhere in Notes/Domino except from within a dialog box.

This command does not work on the Web.

**MailRequestNewPublicKey @Command**
Displays the Mail New Public Key Request dialog box, which allows the user to send a portion of his or her Notes/Domino ID to an administrator, who can create a new public key for the ID and return the ID to the user. The user then merges the updated portion back into the existing user ID.

**Syntax**
@Command( [MailRequestNewPublicKey] )

**Usage**
This can be used anywhere in Notes/Domino except from within a dialog box. The user’s ID file must be certified by a hierarchical certifier.

This command does not work on the Web.
MailScanUnread @Command
Opens the user’s mail database to the first unread document in the view to which the database was most recently open.

Syntax
@Command( [MailScanUnread] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box.
This command does not work on the Web.

MailSend @Command
Displays the Mail Send dialog box, which lets the user choose whether or not to encrypt, sign, or send the selected memo.

Syntax
@Command( [MailSend] )

Usage
A document must be open in read or Edit mode, or selected in a view. The document must contain a SendTo field, indicating the document’s recipients.
You cannot use this function in Web applications.

Language cross-reference
LotusScript NotesDocument class
Java Document class

MailSendCertificateRequest @Command
Displays the Mail Certificate Request dialog box. This lets the user send a safe copy of the Notes/Domino user ID to an administrator, who certifies and then returns it; the user then merges the updated safe copy back into the user ID.

Syntax
@Command( [MailSendCertificateRequest] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box.
This command does not work on the Web.

MailSendEncryptionKey @Command
Displays the User ID dialog box, where the user can define and send encryption keys. If the user’s ID is password–protected, the user will be required to enter the password before being allowed access to the dialog box.
Syntax
@Command( [MailSendEncryptionKey] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box.

This command does not work on the Web.

MailSendPublicKey @Command
Displays the Mail Public Key dialog box. This lets the user send the public key to another user (typically to an administrator who can then paste the public key into that user’s Person record in the Domino Directory).

Syntax
@Command( [MailSendPublicKeyp] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box.

This command does not work on the Web.

MoveToTrash @Command
Marks the currently selected document for deletion.

Note: This @command is new with Release 5.

Syntax
@Command( [MoveToTrash] )

Usage
The View applet is programmable via this @Command.

@Command([MoveToTrash]) is identical to @Command([EditClear]) except, on the Web
@Command([EditClear]) causes the current document to be deleted. @Command([MoveToTrash])
provides consistent behavior for the Notes Client and Web users.

This command does not permanently delete documents; it marks them for deletion. To permanently
delete the documents marked for deletion, the user must refresh the view manually, or programmatically
using any function that refreshes the view, such as the [ViewRefreshFields] @command. A user can also
permanently delete documents marked for deletion by triggering the [EmptyTrash] @command.

MoveToTrash has the same functionality as selecting a document in a view and pressing the Delete key.
You can toggle both. Just as you can remove the deletion mark from a selected document in a view by
pressing the Delete key a second time, if you trigger the MoveToTrash command a second time, the mark
for deletion is removed from the document.

Language cross-reference
DeleteDocument method of LotusScript NotesDocumentCollection class
deleteDocument method of Java DocumentCollection class
Examples: @Command([MoveToTrash])
This formula, when added to the Throw Away action button, marks the currently selected document for deletion.

@Command([MoveToTrash])

Tip: To permanently delete the documents now marked for deletion, call @Command([EmptyTrash]) after this command.

NavigateNext @Command
Navigates to the next document in the current view or folder.

Syntax
@Command([NavigateNext])

Usage
• A database must be open at the view or folder level
  or
• A document can be open in Read or Edit mode

This command executes after all @functions. Use @Command([NavNext]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

For Web applications, use this command only for forms. It does not work for view actions. For instance, if you have a view template form associated with a specific view, and want a Web user to be able to jump to the next document in a document list without having to return to the view to select it, add this command as an action or hotspot to the original form associated with the view, not to the view or view template form. See Designing a form as a view or navigator template in the Application Development with Domino Designer guide for details on view template forms.

Language cross-reference
GetNextDocument method of LotusScript NotesView class
GetNextDocument method of LotusScript NotesViewNavigator class
getNextDocument method of Java View class
getNextDocument method of Java ViewNavigator class

Examples: NavigateNext
1. This example, when added as an action to a form, enables a Web user to jump to the next document in a view without having to return to the view to select the next document from the view list:
   @Command([NavigateNext])

NavigateNextHighlight @Command
Navigates to the next full-text search highlight within a document, working from left to right and top to bottom.

Syntax
@Command([NavigateNextHighlight])
Usage

A document must be open in Read or Edit mode.

This command does not work on the Web.

NavigateNextMain @Command

Navigates to the next main document in the current view.

Syntax

@Command( [NavigateNextMain] )

Usage

• A database must be open at the view or folder level
  or
• A document can be open in Read or Edit mode

This command executes after all @functions. Use @Command([NavNextMain]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

For Web applications, use this command only for forms. It does not work for view actions.

Language cross-reference

GetNextCategory method of LotusScript NotesViewNavigator class

getNextCategory method of Java ViewNavigator class

NavigateNextSelected @Command

Navigates to the next selected document in the current view or folder.

Syntax

@Command( [NavigateNextSelected] )

Usage

• A database must be open at the view or folder level
  or
• A document can be open in Read or Edit mode

This command does not work on the Web.

This command executes after all @functions. Use @Command([NavNextSelected]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

Language cross-reference

GetNextDocument method of LotusScript NotesDocumentCollection class

getNextDocument method of Java DocumentCollection class
**NavigateNextUnread** @Command

Navigates to the next unread document in the current view or folder.

**Syntax**

@Command( [NavigateNextUnread] )

**Usage**

- A database must be open at the view or folder level or
- A document can be open in Read or Edit mode

This command does not work on the Web.

This command executes after all @functions. Use `NavNextUnread` to execute immediately. See the Order of evaluation for formula statements topic for more details.

---

**NavigatePrev** @Command

Navigates to the previous document in the current view or folder.

**Syntax**

@Command( [NavigatePrev] )

**Usage**

- A database must be open at the view level or
- A document can be open in Read or Edit mode

This command executes after all @functions. Use `@Command([NavPrev])` to execute immediately. See the Order of evaluation for formula statements topic for more details.

For Web applications, use this command only for forms. It does not work for view actions. For instance, if you have a view template form associated with a specific view, and want a Web user to be able to jump to the previous document in a document list without having to return to the view to select it, add this command as an action or hotspot to the original form associated with the view, not to the view or view template form. See Designing a form as a view or navigator template in the Application Development with Domino Designer guide for details on view template forms.

**Language cross-reference**

- GetPrevDocument method of LotusScript NotesViewNavigator class
- GetPrevDocument method of LotusScript NotesView class
- getPrevDocument method of Java ViewNavigator class
- getPrevDocument method of Java View class

**Examples: NavigatePrev**

1. This example, when added as an action to a form, enables a Web user to jump to the previous document in a view without having to return to the view to select the previous document from the view list:
@Command([NavigatePrev])

**NavigatePrevHighlight @Command**

Navigates to the previous full-text search highlight within a document, working from right to left and bottom to top.

**Syntax**

@Command( [NavigatePrevHighlight] )

**Usage**

The document must be open in Read or Edit mode and the highlight must already be moved past the first highlighted occurrence.

This command does not work on the Web.

**NavigatePrevMain @Command**

Navigates to the previous main document in the current view or folder.

**Syntax**

@Command( [NavigatePrevMain] )

**Usage**

- A database must be open at the view or folder level
  
or
  
- A document can be open in Read or Edit mode

This command executes after all @functions. Use `@Command([NavPrevMain])` to execute immediately. See the [Order of evaluation for formula statements](#) topic for more details.

For Web applications, use this command only for forms. It does not work for view actions.

**Language cross-reference**

GetPrevCategory method of LotusScript NotesViewNavigator class

getPrevCategory method of Java ViewNavigator class

**NavigatePrevSelected @Command**

Navigates to the previous selected document in the current view or folder.

**Syntax**

@Command( [NavigatePrevSelected] )

**Usage**

- A database must be open at the view or folder level
  
or
  
- A document can be open in Read or Edit mode

This command does not work on the Web.
This command executes after all @functions. Use @Command([NavPrevSelected]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

**Language cross-reference**

getPrevDocument method of Java DocumentCollection class

---

**NavigatePrevUnread @Command**

Navigates to the previous unread document in the current view or folder.

**Syntax**
@Command( [NavigatePrevUnread] )

**Usage**
This command executes after all @functions. Use @Command([NavPrevUnread]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

A database must be open at the view or folder level, or a document can be open in Read or Edit mode.

This command does not work on the Web.

---

**NavigateToBacklink @Command**

Returns to the document from which you launched the current document.

**Syntax**
@Command( [NavigateToBacklink] )

**Usage**
When you open a document, launch another document from it by activating a link, and then close the second document, NavigateToBacklink closes the database containing the second document and returns to the first document.

This command does not work on the Web.

**Language cross-reference**
GetPrevDocument method of LotusScript NotesView class

getPrevDocument method of Java View class

---

**NavigatorProperties @Command**

Displays the Properties box for a navigator.

**Syntax**
@Command( [NavigatorProperties] )
Usage
A navigator must be open in Design mode.

This command does not work on the Web.

**NavigatorTest @Command**

Opens a navigator in Testing mode, so you can test its hotspots, formulas, and scripts. Selecting this command again puts the navigator back into Design mode.

**Syntax**

```
@Command( [NavigatorTest] )
```

**Usage**

A navigator must be open in Design mode.

This command does not work on the Web.

**NavNext @Command**

Navigates to the next document in the current view or folder.

**Note:** This command is new with Lotus Domino Release 6.

**Syntax**

```
@Command( [NavNext] )
```

**Usage**

- A database must be open at the view or folder level
  - or
- A document can be open in Read or Edit mode

This command executes immediately. Use the @NavigateNext command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

For Web applications, use this command only for forms. It does not work for view actions. For instance, if you have a view template form associated with a specific view, and want a Web user to be able to jump to the next document in a document list without having to return to the view to select it, add this command as an action or hotspot to the original form associated with the view, not to the view or view template form. See Designing a form as a view or navigator template in the Application Development with Domino Designer guide for details on view template forms.

**Language cross-reference**

- GetNextDocument method of LotusScript NotesView class
- GetNextDocument method of LotusScript NotesViewNavigator class
- getNextDocument method of Java View class
- getNextDocument method of Java ViewNavigator class
**NavNextMain @Command**

Navigates to the next main document in the current view.

**Note:** This command is new with Lotus Domino Release 6.

**Syntax**

@Command( [NavNextMain] )

**Usage**

- A database must be open at the view or folder level
  or
- A document can be open in Read or Edit mode

This command executes immediately. Use the `NavigateNextMain @command` to execute after all `@functions`. See the `Order of evaluation for formula statements` topic for more details.

For Web applications, use this command only for forms. It does not work for view actions.

**Language cross-reference**

- GetNextCategory method of LotusScript NotesViewNavigator class
- getNextCategory method of Java ViewNavigator class

---

**NavNextSelected @Command**

Navigates to the next selected document in the current view or folder.

**Note:** This command is new with Lotus Domino Release 6.

**Syntax**

@Command( [NavNextSelected] )

**Usage**

- A database must be open at the view or folder level
  or
- A document can be open in Read or Edit mode

This command does not work on the Web.

This command executes immediately. Use the `NavigateNextSelected @command` to execute after all `@functions`. See the `Order of evaluation for formula statements` topic for more details.

**Language cross-reference**

- GetNextDocument method of LotusScript NotesDocumentCollection class
- getNextDocument method of Java DocumentCollection class
NavNextUnread @Command
Navigates to the next unread document in the current view or folder.

Note: This command is new with Lotus Domino Release 6.

Syntax
@Command( [NavNextUnread] )

Usage
- A database must be open at the view or folder level
  or
- A document can be open in Read or Edit mode

This command does not work on the Web.

This command executes immediately. Use the NavigateNextUnread @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

NavPrev @Command
Navigates to the previous document in the current view or folder.

Note: This command is new with Lotus Domino Release 6.

Syntax
@Command( [NavPrev] )

Usage
- A database must be open at the view level
  or
- A document can be open in Read or Edit mode

This command executes immediately. Use the NavigatePrev @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

For Web applications, use this command only for forms. It does not work for view actions. For instance, if you have a view template form associated with a specific view, and want a Web user to be able to jump to the previous document in a document list without having to return to the view to select it, add this command as an action or hotspot to the original form associated with the view, not to the view or view template form. See Designing a form as a view or navigator template in the Application Development with Domino Designer guide for details on view template forms.

Language cross-reference
GetPrevDocument method of LotusScript NotesViewNavigator class
GetPrevDocument method of LotusScript NotesView class
getPrevDocument method of Java ViewNavigator class
getPrevDocument method of Java View class
NavPrevMain @Command

Navigates to the previous main document in the current view or folder.

Note: This command is new with Lotus Domino Release 6.

Syntax

@Command( [NavPrevMain] )

Usage

- A database must be open at the view or folder level
  or
- A document can be open in Read or Edit mode

This command executes immediately. Use the NavigatePrevMain @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

For Web applications, use this command only for forms. It does not work for view actions.

Language cross-reference

GetPrevCategory method of LotusScript NotesViewNavigator class

getPrevCategory method of Java ViewNavigator class

NavPrevSelected @Command

Navigates to the previous selected document in the current view or folder.

Note: This command is new with Lotus Domino Release 6.

Syntax

@Command( [NavPrevSelected] )

Usage

- A database must be open at the view or folder level
  or
- A document can be open in Read or Edit mode

This command does not work on the Web.

This command executes immediately. Use the NavigatePrevSelected @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

Language cross-reference


getPrevDocument method of Java DocumentCollection class
**NavPrevUnread @Command**

Navigates to the previous unread document in the current view or folder.

**Note:** This command is new with Lotus Domino Release 6.

**Syntax**

@Command( [NavPrevUnread] )

**Usage**

This command executes immediately. Use the `NavigatePrevUnread @command` to execute after all @functions. See the [Order of evaluation for formula statements](#) topic for more details.

A database must be open at the view or folder level, or a document can be open in Read or Edit mode.

This command does not work on the Web.

---

**ObjectDisplayAs @Command**

Displays the Display As dialog box, which lets you change the display settings of the selected OLE object.

**Note:** `ObjectDisplayAs` is not supported under OS/2 or UNIX, or on the Macintosh.

**Syntax**

@Command( [ObjectDisplayAs] )

**Usage**

An OLE object must be selected.

This command does not work on the Web.

---

**ObjectOpen @Command**

Opens an OLE object for editing.

**Note:** `ObjectOpen` is not supported under OS/2 or UNIX, or on the Macintosh.

**Syntax**

@Command( [ObjectOpen] )

**Usage**

An OLE object must be selected.

This command does not work on the Web.

---

**Language cross-reference**

Activate method of LotusScript NotesEmbeddedObject class

activate method of Java EmbeddedObject class
**ObjectProperties @Command**
Displays the Properties box for the selected OLE object.

**Syntax**
@Command( [ObjectProperties] )

**Usage**
This command does not work on the Web.

**Language cross-reference**
LotusScript NotesEmbeddedObject class
Java EmbeddedObject class

---

**OpenCalendar @Command**
Opens a mail file to the Calendar view.

**Syntax**
@Command( [OpenCalendar]; username )

**Parameter**
*username*
Text. Optional. User name of the owner of the mail file.

**Usage**
If you specify *username*, this command opens the mail file of that user. If you omit *username*, this command displays the name picker dialog, then opens the mail file of the selected user. This command temporarily adds the icon for the mail database to the workspace.

This command does not work on the Web.

**Language cross-reference**
OpenView method of LotusScript NotesUIDatabase class

---

**OpenDocument @Command**
Opens a document.

**Syntax**
@Command( [OpenDocument]; writeOrReadOnly ; UNID ; width : height)

**Parameters**
*writeOrReadOnly*
Text ("1" or "0"). Optional. A value of "1" opens the document in Edit mode. A value of "0" (the default) opens the document in Read-only mode.

*UNID*
Text. Optional. The UNID (unique document ID) of the document you want to open. If you omit this parameter, Notes/Domino opens the currently selected document. You can find a document’s unique id on the Document IDs tab of the Document properties box. The UNID begins after "OF" on the first line and continues after "ON" on the second line. Omit the colons.

width : height

Number list. Optional. The width and height, in inches, of the window for the document you open. If you omit this parameter or use zero for either value, you create the window at the default size (usually the size that the last user set).

Note: The width and height parameters have no effect in Release 5 and later.

Usage
A database must be open to a database view and the view must contain the document you want to open.

The width-height parameter does not apply in MDI mode when the window is maximized. When restored, the window returns to the size you specify. The measurement in inches matches the ruler bar in the editor, so you can use the ruler bar to guide you in sizing the window. When you specify the width and height, you center the window in the enclosing Notes/Domino window (for MDI mode) or in the operating desktop (for Mac and SDI mode).

You can use this command in Web applications, but it must be used in conjunction with @Command([FileOpenDatabase]) or @Command ([OpenView]). You can use the URL command to open a document by its UNID; see the URL commands for opening documents by key section of the Application Development with Domino Designer guide.

Language cross-reference
EditDocument method of LotusScript NotesUIWorkspace class
GetDocumentByUNID method of LotusScript NotesDatabase class
getDocumentByUNID method of Java Database class

Examples: OpenDocument
1. This formula opens the selected document in Read mode.
   @Command([OpenDocument])
2. This formula also opens the selected document in Read mode.
   @Command([OpenDocument];"0")
3. This formula opens the selected document in Edit mode and moves the cursor to the Subject field.
   @Command([OpenDocument];"1");
   @Command([EditGotoField];"Subject")
4. This formula opens a document by UNID.
   @Command([OpenDocument];"";"F56ACC1F6F27155D8525686500603D43")
5. This formula opens the document, "Savage Mountain," in the By Title view of a Web application.
   @Command([OpenView];"By Title";"Savage Mountain");
   @Command([OpenDocument])
6. This formula opens the document, "Savage Mountain," in Edit mode when it is accessed by a browser.
   @Command([OpenView];"By Title";"Savage Mountain");
   @Command([OpenDocument]; "1")
OpenFrameset @Command

Opens a frameset defined for the current database. Framesets provide a way for designers to display several pages at the same time. A frame is actually one page; a frameset is a collection of pages. Page designers can create links between frames. A major advantage of framesets is the ability to leave one page constant as users scroll or link to other pages.

Note: This @command is new with Release 5.

Syntax
@Command( [OpenFrameset]; frameset )

Parameters
frameset
Text. The name of a frameset defined for the current database.

Usage
@Command([OpenFrameset]) is used in action formulas. You can use this command in Web applications.

Language cross-reference
OpenFrameSet method in LotusScript NotesUIWorkspace class

Examples: @Command([OpenFrameset])
This code, when added to an action button on a form, saves and closes the current form and opens the usersView frameset.
@Command([FileSave]);
@Command([CloseWindow]);
@Command([OpenFrameset];"usersView")

OpenHelpDocument @Command

Allows you to create your own context-sensitive help documents. To use this command, you must first create a view that has a sorted first column containing key values that uniquely identify each help document. @Command([OpenHelpDocument]) searches this sorted view for the specified key value and, if found, displays the associated help document in a separate Help window.

Tip: If you use this @command as the formula for the onHelp event for a form, for example, when a user presses <F1> in the context of this form, the custom help document associated with it displays instead of the standard Notes Help database that is usually triggered when a user presses <F1>.

Note: This @command is new with Release 5.

Syntax
@Command( [OpenHelpDocument]; server :database;viewname; key )

or

@Command( [OpenHelpDocument]; [ HelpDatabase ];viewname; key )
Parameters

server
Text. The name of the server where the database resides.

database
Text. Required. The path and file name (in the appropriate format for the operating system) of the database you want to open.

viewName
Text. The name of the view you want to open in the database.

key
Text. Indicates which document you want Notes to scroll to when it opens viewName. The key is a value that appears in the first sorted column of viewName. It can either be a formula or a hard-coded value such as the title of a document.

[HelpDatabase]
Keyword. If you supply one of the following keywords instead of server:database values, this @command opens the corresponding Lotus Notes Help files:


Usage
In the Help database, the column that contains the key must be sorted.

The Help database must be on either a server or in the local data directory. If you do not specify the server name, it defaults to your local data directory.

If you do not specify the database name, it searches within the Help database. For example, in the Notes 6 Client, the search occurs in Lotus Domino 6 Help (HELP6_CLIENT.nsf).

For Web access, this command is useful to create a button to get Help documents.

Tip: You can also create pages for Help and open them with @Command([OpenPage]).

Examples: @Command([OpenHelpDocument])

1. This formula, when added to a view action labeled Help, opens a specific help document when a user triggers it. It displays the document that contains the value "customerView" in its about field in a Help window. The myHelpDocs view is a hidden view containing all the custom help documents for the database. Its first column is a sorted column containing the value of the "about" field. Since this formula specifies no server, database, or keywords, OpenHelpDocument searches the current database for the specified Help document.
   @Command([OpenHelpDocument];"";"myHelpDocs";"customerView")

2. This formula, when added to a hotspot button labeled Help on a form, opens a Help window that displays a custom Help document. It displays the document that contains the value "orderForm" in its
topic field. The Keys view is a hidden view containing all the custom help documents for the database. Its first column is a sorted column containing the value of the "topic" field. Since this formula specifies no server, database, or keywords, OpenHelpDocument searches the current database for the specified Help document.

@Command([OpenHelpDocument];"*";"Keys";"orderForm")

3. This formula, when added to a Help hotspot button, displays the Formula Language Help topic from the Lotus Domino Designer Help database in a separate Help window.

@Command([OpenHelpDocument];[DesignerHelp];"(Help)";"FormulaLanguage")

---

**OpenNavigator @Command**

Opens a navigator defined for the selected database.

**Syntax**

@Command( [OpenNavigator] ; navigator ; solo )

**Parameters**

*navigator*

Text. The name of a navigator defined for the selected database. OpenNavigator opens this navigator in an existing navigation pane or window.

*solo*

Number ("1"). Optional. If you include this value, Notes opens the specified navigator in its own window.

**Usage**

OpenNavigator opens this specified navigator in an existing navigation pane or window.

You can use this command in Web applications but you must omit the solo argument.

**Language cross-reference**

OpenNavigator method in LotusScript NotesUIDatabase class

---

**OpenPage @Command**

Opens a page defined for the current database. A page is a design element that structures and displays information, including text, graphics, applets, and links. Unlike a form, a page cannot contain fields, subforms, layout regions, and some embedded controls.

**Note:** This @command is new with Release 5.

**Syntax**

@Command( [OpenPage] ; page )

**Parameters**

*page*

Text. The name of a page defined for the current database.
Usage
@Command([OpenPage]) is used in action formulas. You can use this command in Web applications.

Language cross-reference
OpenPage method in LotusScript NotesUIWorkspace class

OpenView @Command
Opens the specified view in the current database.

Syntax
@Command([OpenView]; viewName; key; newInstance)

Parameters

viewName
Text. Optional. The name of the view you want to open. If you omit the view name, the database opens to its default view; or, if the user has opened the database before, to the last view used by that person. If the database is already open to the specified view, Notes/Domino makes that the topmost window.

key
Text. Optional. Indicates which document you want Notes/Domino to scroll to when it opens viewName. The key is a value that appears in the first sorted column of viewName. If you omit the key, no document is selected.

newInstance
Number. Optional. Specify “1” if you want the view to open in a new window, even if there’s already a window open for the database. If you omit this parameter, the new window is opened only when it’s actually needed.

Note: The view must be sorted in order for the key to work; otherwise, no document is selected when the view opens. The key column must be the first sorted column when multiple sorted columns are present.

Usage
You can use this command in Web applications.

If you specify the newInstance parameter for @Command([OpenView]), the @SetTargetFrame function is ignored.

If you do not specify a viewName then the last view is the one that will open in the specified targetframe of @SetTargetFrame.

To open a view that is embedded on a page or form, use @Command([OpenPage]) or @Command([OpenDocument]) respectively.

Language cross-reference
OpenView method in LotusScript NotesUIDatabase class
Examples: OpenView

1. This formula opens the Reverse Chronology view for the current database.
   
   @Command([OpenView]; "Reverse Chronology")

2. This code, when added to as the formula for an action button on a form, opens the Managers Only view of the current database if the current user has Manager level access in the ACL. Otherwise, it opens the Employees view.
   
   @If(@UserAccess(@DbName) = "6"; @Command([OpenView]; "Managers Only"); @Command([OpenView]; "Employees"))

---

PasteBitmapAsBackground @Command

Paste a bitmap into the background of a navigator. All other objects in the navigator are displayed on top of the background bitmap.

Syntax

@Command( [PasteBitmapAsBackground] )

Usage

A navigator must be open in Design mode, and the bitmap you want to paste must be on the Clipboard.

This command does not work on the Web.

---

PasteBitmapAsObject @Command

Paste a bitmap into a navigator. The bitmap becomes a hotspot which can be edited like any other hotspot.

Syntax

@Command( [PasteBitmapAsObject] )

Usage

A navigator must be open in Design mode, and the bitmap you want to paste must be on the Clipboard.

This command does not work on the Web.

---

PictureProperties @Command

Displays the Properties box for a bitmap in a rich text field.

Syntax

@Command( [PictureProperties] )

Usage

A bitmap must be selected.

This command does not work on the Web.

---

PublishDatabase @Command

Displays the Publish Database dialog box, where the user selects a library in which to publish the selected database.
Syntax
@Command( [PublishDatabase] )

Usage
A database must be open or selected on the workspace. Only one database can be selected. There must be at least one library on the user’s workspace.

This command does not work on the Web.

---

**RefreshFrame @Command**

Refreshes the specified frame in a frameset.

**Note:** This command is new with Release 6.

Syntax
@Command( [RefreshFrame]; targetFrame )

**Parameters**

**targetFrame**

Text. Required. The name of the frame you want to refresh.

If you omit the frame name, no frames are refreshed. You can omit the targetFrame parameter if you specify a target frame using the `@SetTargetFrame` function before executing this command.

**Usage**

If the command is executed in nested framesets that have two frames which share the same name, Notes refreshes the frame in the outermost frameset only.

In Web applications, @Command([RefreshFrame]) ignores the targetFrame parameter and refreshes only the frame from which the command is executed.

**Examples: RefreshFrame command**

1. This code, when used in a hotspot button in a frame, refreshes the frame named “right” in that frameset.
   
   ```plaintext
   @Command([RefreshFrame];"right")
   ```

2. This code does not refresh any frames.
   
   ```plaintext
   @Command([RefreshFrame])
   ```

3. This code refreshes the “left” frame of the frameset.
   
   ```plaintext
   @SetTargetFrame("left");
   @Command([RefreshFrame])
   ```

4. This example is used in a nested frameset named “outerFrameset” that contains two frames, named “left” and “right.” Its “right” frame contains another frameset, named “innerFrameset”, which also contains two frames named “left” and “right.” This code, when used in a hotspot button in any of the frames, either in the “outerFrameset” or “innerFrameset” framesets, refreshes both frames in the “innerFrameset” frameset. It executes on only the “outerFrameset,” by updating the contents of its “right” frame.
   
   ```plaintext
   @Command([RefreshFrame];"right")
   ```
**RefreshHideFormulas @Command**

Refreshes only the hidden formulas in a document or view.

**Syntax**

@Command( [RefreshHideFormulas] )

**Usage**

A document or view must be open.

This command does not work on the Web.

**Language cross-reference**

RefreshHideFormulas method of LotusScript NotesUIDocument class

---

**RefreshParentNote @Command**

This formula sends the values entered in the dialog box to the parent document. A designer can update a parent note and close the dialog box without having to use the OK button on the dialog box.

**Note:** This @command is new with Release 5.

**Syntax**

@Command( [RefreshParentNote] )

**Usage**

Used only in dialog boxes.

This command does not work on the Web.

**Language cross-reference**

RefreshParentNote method of LotusScript NotesUIWorkspace class

**Examples:**

@Command([RefreshParentNote])

1. This code, when added to the Refresh hotspot button on the userInput form, updates fields in the parent document that share names with fields in the userInput form if the userInput form is called from the parent document using the @DialogBox function.

   @Command([RefreshParentNote])

   The @DialogBox function could be used in a hotspot button on the parent document with the following code:

   @DialogBox("userInput";[NOOKCANCEL];[NOCANCEL];"User input")

---

**RefreshWindow @Command**

Reloads or refreshes the contents of the current window.

**Note:** This command is new with Release 6.

**Syntax**

@Command( [RefreshWindow] )
Usage
This command executes immediately. Use the [ReloadWindow @command] to execute after all @functions. See the [Order of evaluation for formula statements] topic for more details.

If the window is a frameset, then the entire frameset and it's contents are reloaded. Also, a Web page in a frame in the current window will be reloaded from the Web. All other windows are refreshed.

Language cross-reference
ReloadWindow method of LotusScript NotesUIWorkspace class

ReloadWindow @Command
Reloads or refreshes the contents of the current window.

Syntax
@Command( [ReloadWindow] )

Usage
This command executes after all @functions. Use the [@Command([RefreshWindow])] to execute immediately. See the [Order of evaluation for formula statements] topic for more details.

If the window is a frameset, then the entire frameset and it's contents are reloaded. Also, a Web page in a frame in the current window will be reloaded from the Web. All other windows are refreshed.

Language cross-reference
ReloadWindow method of LotusScript NotesUIWorkspace class

RemoteDebugLotusScript @Command
From the Lotus Notes Remote Debugger client, opens the Select Debug Target dialog box.

Note: This @command is new with Release 6.

Syntax
@Command([RemoteDebugLotusScript])

Usage
This function works only from within the Remote Debugger client. To access the Lotus Notes Debugger client from Designer, select File - Tools - Remote Debugging from the Designer menu.

This function is used in the Attach to a debug target icon on the Debugger Welcome Page to mimic the File - Select Debug Target menu command.

This command does not work on the Web.

RemoveFromFolder @Command
Removes the selected document from the current folder.

Syntax
@Command( [RemoveFromFolder] )
**Usage**
A document must be selected in a folder or opened from a folder (not a view). If several documents are selected, they are all removed from the folder.

**Note:** The following feature is new with Release 5.

This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder.

**Language cross-reference**
RemoveFromFolder method of LotusScript NotesDocument class
removeFromFolder method of Java Document class

---

**RenameDatabase @Command**
For a particular database on the user’s workspace, RenameDatabase locates a replica of that database (based on its replica ID) on the specified server, and replaces the database icon with that of the replica. The only change the user sees is that the server name has changed on the database icon.

**Syntax**
@Command( [RenameDatabase] ; server : database ; newServer )

**Parameters**

*server*
Text. The name of the server where the database is.

*database*
Text. The path and file name of the database. Specify the database’s name and location using the appropriate format for the operating system.

*newServer*
Text. The name of the server where a replica of the database is.

**Usage**
You can only use this for replicas of a database, not copies. You cannot use this command to change the name of a server.

If the workspace has stacked replica icons for the database on both server and newServer, RenameDatabase brings the icon for newServer to the front.

This command does not work on the Web.

**Language cross-reference**
OpenByReplicaID method of LotusScript NotesDatabase class
Examples: RenameDatabase
1. This formula replaces the database icon pointing to the STATUSRP.NSF database on the CENTRAL server to that on the WEST server. To the user, the only change is that the database icon now displays WEST as the server name.
   @Command([RenameDatabase]; "CENTRAL" : "Statusrp.NSF"; "WEST")
2. This formula brings the database icon pointing to the WEST server to the front, if the workspace already has stacked icons for STATUSRP.NSF on both CENTRAL and WEST. The icon pointing to the CENTRAL server is stacked underneath.
   @Command([RenameDatabase]; "CENTRAL" : "Statusrp.NSF"; "WEST")

Replicator @Command
Displays the Replicator on the Notes workspace page.

Syntax
@Command( [Replicator] )

Usage
This command does not work on the Web.

ReplicatorReplicateHigh @Command
Initiates replication of databases that have been assigned a priority of High.

Syntax
@Command( [ReplicatorReplicateHigh] )

Usage
ReplicatorReplicateHigh is available only when Notes is open to the Replicator workspace page.

Note: This restriction is lifted starting with Release 6.

This command does not work on the Web.

ReplicatorReplicateNext @Command
Stops replication of the currently replicating database and initiates replication of the next database selected for replication.

Syntax
@Command( [ReplicatorReplicateNext] )

Usage
ReplicatorReplicateNext may result in partial replication of the currently replicating database.

ReplicatorReplicateNext is available only when Notes/Domino is open to the Replicator workspace page.

Note: This restriction is lifted starting with Release 6.

This command does not work on the Web.
**ReplicatorReplicateSelected @Command**

Initiates replication of the selected database.

**Syntax**

@Command( [ReplicatorReplicateSelected] )

**Usage**

ReplicatorReplicateSelected is available only when Notes/Domino is open to the Replicator workspace page.

**Note:** This restriction is lifted starting with Release 6.

This command does not work on the Web.

**ReplicatorReplicateWithServer @Command**

Displays the "Replicate With Which Server" dialog box and initiates replication with the server you select.

**Syntax**

@Command( [ReplicatorReplicateWithServer] )

**Usage**

ReplicatorReplicateWithServer is available only when Notes/Domino is open to the Replicator workspace page.

**Note:** This restriction is lifted starting with Release 6.

This command does not work on the Web.

**Language cross-reference**

Replicate method of LotusScript NotesDatabase class

replicate method of Java Database class

**ReplicatorSendMail @Command**

Sends local pending mail to the server.

**Syntax**

@Command( [ReplicatorSendMail] )

**Usage**

ReplicatorSendMail is available only when Notes/Domino is open to the Replicator workspace page.

**Note:** This restriction is lifted starting with Release 6.

This command does not work on the Web.

**ReplicatorSendReceiveMail @Command**

Initiates replication between the mail server and your local mail database.
Syntax
@Command( [ReplicatorSendReceiveMail] )

Usage
ReplicatorSendReceiveMail is available only when Notes/Domino is open to the Replicator workspace page.

Note: This restriction is lifted starting with Release 6.

This command does not work on the Web.

ReplicatorStart @Command
Initiates or resumes replication of the selected databases.

Syntax
@Command( [ReplicatorStart] )

Usage
ReplicatorStart is available only when Notes/Domino is open to the Replicator workspace page.

Note: This restriction is lifted starting with Release 6.

This command does not work on the Web.

Language cross-reference
Replicate method of LotusScript NotesDatabase class
replicate method of Java Database class

ReplicatorStop @Command
Stops the current replication process.

Syntax
@Command( [ReplicatorStop] )

Usage
ReplicatorStop may result in partial replication of the currently replicating database.

ReplicatorStop is available only when Notes/Domino is open to the Replicator workspace page.

Note: This restriction is lifted starting with Release 6.

This command does not work on the Web.

RunAgent @Command
Executes a specified agent.

Note: This command is new with Release 6.
Syntax
@Command( [RunAgent] ; agent )

Parameters
agent
Text. Optional. The name of the agent you want to run.

If you omit the agent name, Notes displays a list of Agents in the database, where the user can select which agent to run.

Usage
This command executes immediately. Use the ToolsRunMacro @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

When specifying a hidden agent, include the parentheses as shown below.
@Command([RunAgent];"(hiddenagentname)")

You can use this command in Web applications.

Language cross-reference
Run method of LotusScript NotesAgent class
run method of Java Agent class

RunScheduledAgents @Command
Runs all of the database’s scheduled agents, regardless of when they are scheduled to run. The agents will then run as usual at their regularly scheduled times.

Note: This command is new with Release 6.

Syntax
@Command( [RunScheduledAgents] )

Usage
This command executes immediately. Use the ToolsRunBackgroundMacros @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

A database must be open or selected on the workspace.

This command does not work on the Web.

SectionCollapse @Command
Collapses the current section in a document, form, or subform.

Syntax
@Command( [SectionCollapse] )

592 Programing Guide, Volume 1: Overview and Formula Language
**Usage**

- A document must be open in Read or Edit mode and a section must be selected.
- A form or subform must be open in Design mode and a section must be selected.

This command does not work on the Web.

---

**SectionCollapseAll @Command**

Collapses all the sections in a document, page, form, or subform.

**Syntax**

@Command( [SectionCollapseAll] )

**Usage**

- A document must be open in Read or Edit mode.
- A form or subform must be open in Design mode.

If sections exist in a table only, this command triggers the message, "Cannot execute the specified command." To prevent this behavior, add a section outside the table, to the page or form background.

This command does not work on the Web.

---

**Language cross-reference**

CollapseAllSections method of LotusScript NotesUIDocument class

---

**SectionDefineEditors @Command**

Displays the Edit Section dialog box for the current section on a form.

**Syntax**

@Command( [SectionDefineEditors] )

**Usage**

The current section must be a controlled access section and the form must be open in Edit mode.

This command does not work on the Web.

---

**SectionExpand @Command**

Expands the current section in a document, form, or subform.

**Syntax**

@Command( [SectionExpand] )

**Usage**

- A document must be open in Read or Edit mode and a section must be selected.
- A form or subform must be open in Design mode and a section must be selected.

This command does not work on the Web.
SectionExpandAll @Command
Expands all the sections in a document, page, form, or subform.

Syntax
@Command( [SectionExpandAll] )

Usage
- A document must be open in Read or Edit mode.
- A form or subform must be open in Design mode.

If sections exist in a table only, this command triggers the message, "Cannot execute the specified command." To prevent this behavior, add a section outside the table, to the page or form background.

This command does not work on the Web.

Language cross-reference
ExpandAllSections method of LotusScript NotesUIDocument class

SectionProperties @Command
On a form or subform, displays the Properties box for the selected section.

Syntax
@Command( [SectionProperties] )

Usage
- A form or subform must be open in Design mode and a section must be selected. SectionProperties does not work for sections on documents based on that form or subform.
- A document must be open in Edit mode and a section in a rich text field must be selected.

This command does not work on the Web.

SectionRemoveHeader @Command
Removes the contents of a section from the section. The contents are then displayed as they existed before the section was created.

Syntax
@Command( [SectionRemoveHeader] )

Usage
- A document must be open in Edit mode with a section selected
  or
- A form or subform must be open in Design mode with a section selected

This command does not work on the Web.
**SendInstantMessage @Command**

Starts a chat with one or more users.

*Note:* This @command is new with Release 6.5.

**Syntax**

@Command( [SendInstantMessage] ; names )

**Parameters**

*names*

Text list. Optional. The name or names to start chatting with. If this parameter is omitted, the user is prompted.

**SetCurrentLocation @Command**

Displays the Choose Location dialog box, where you can choose your current workstation location, such as Office, Island, or Travel.

**Syntax**

@Command( [SetCurrentLocation] )

**Usage**

This command works everywhere in Notes/Domino. It does not work in Web applications. It’s most convenient to use a toolbar button to invoke this command.

**Language cross-reference**

SetCurrentLocation method of LotusScript NotesUIWorkspace class

---

**ShowHideIMContactList @Command**

Toggles the display of the Instant Messaging Contact List.

*Note:* This @command is new with Release 6.5.

**Syntax**

@Command( [ShowHideIMContactList] )

**ShowHideLinkPreview @Command**

Toggles the display of the link preview pane.

**Syntax**

@Command( [ShowHideLinkPreview] ; showOrHide )

**Parameters**

*showOrHide*

Number. Optional. Specify "1" if you want to show the link preview pane, "0" if you want to hide it. If you omit this parameter, the @command toggles the current state of the link preview pane.
Usage
A document must be open in Read or Edit mode.

This command does not work on the Web.

Language cross-reference
PreviewDocLink property of LotusScript NotesUIDocument class

ShowHideParentPreview @Command
Toggles the display of the parent document preview pane.

Syntax
@Command( [ShowHideParentPreview] ; showOrHide )

Parameters
showOrHide
Number. Optional. Specify "1" if you want to show the parent preview pane, "0" if you want to hide it. If you omit this parameter, the @command toggles the current state of the parent preview pane.

Usage
A document must be open in Read or Edit mode.

This command does not work on the Web.

Language cross-reference
PreviewParentDoc property of LotusScript NotesUIDocument class

ShowHidePreviewPane @Command
Toggles the display of the document preview pane.

Syntax
@Command( [ShowHidePreviewPane] ; showOrHide )

Parameters
showOrHide
Number. Optional. Specify "1" if you want to show the document preview pane, "0" if you want to hide it. If you omit this parameter, the @command toggles the current state of the document preview pane.

Usage
A document must be selected in a view or folder.

This command does not work on the Web.

Language cross-reference
InPreviewPane property of LotusScript NotesUIDocument class
ShowProperties @Command
Displays the Properties box for the currently selected Notes object. For example, if a document is selected in a view, displays the document Properties box; if a field is selected on a form, displays the field Properties box.

Syntax
@Command( [ShowProperties] )

Usage
This command does not work on the Web.

SmartIconsFloating @Command
Makes the SmartIcons palette “float” so that users can move it around on the screen.

Note: This @command does not function in Release 6.

Syntax
@Command( [SmartIconsFloating] )

Usage
This command does not work on the Web.

SmartIconsNextSet @Command
Switches to display the next set of SmartIcons in the Icon bar.

Note: This @command does not function in Release 6.

Syntax
@Command( [SmartIconsNextSet] )

Usage
This command does not work on the Web.

StyleCycleKey @Command
Cycles through the list of styles that have been defined for the current document, form, or page. This is the same as pressing F11. Each time you invoke StyleCycleKey, a different style goes into effect.

Syntax
@Command( [StyleCycleKey] )

Usage
• A document must be open in Edit mode
  or
• A form, subform, or page must be open in Design mode

In both cases, there must be at least two styles assigned to the style cycle list.
This command does not work on the Web.

**SwitchForm @Command**

Changes the form used to display the current document.

**Note:** This command is new with Release 6.

**Syntax**

@Command( [SwitchForm] ; formName )

**Parameters**

*formName*

Text. Optional. The name of the form you want to switch to.

With no parameter, ViewSwitchForm displays a dialog box with a list of forms available in the current database.

**Usage**

This command executes immediately. Use the ViewSwitchForm @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

This command does not work on the Web.

---

**SwitchView @Command**

Switches to the specified view or folder within the current database or, if a view or folder is not specified, displays the View menu so the user can select a view.

**Note:** This command is new with Release 6.

**Syntax**

@Command( [SwitchView] )

or

@Command( [SwitchView] ; viewName )

**Parameters**

*viewName*

Text. Optional. The name of the view or folder you want to switch to.

**Usage**

This command executes immediately. Use the ViewChange @command to execute after all @functions. See the Order of evaluation for formula statements topic for more details.

This command doesn’t work from within a document or form, so you cannot use it to view a document through another form. Use @Command([SwitchForm]) instead.

You can use this command in Web applications, but you must use the viewName parameter.
**Language cross-reference**
OpenView method of LotusScript NotesUIDatabase class

---

**TextColor @Command**
Centers the current text.

**Syntax**
```
@Command( [TextColor] )
```

**Usage**
- A document must be open in Edit mode with the insertion point in a rich text field
- A form, subform, or page must be open in Design mode.

This command does not work on the Web.

---

**Language cross-reference**
Alignment property of LotusScript NotesRichTextParagraphStyle class
Alignment property of Java RichTextParagraphStyle class

---

**TextColor @Command**
Aligns the text at both the right and left edges of the field so that the text forms a block.

**Syntax**
```
@Command( [TextColor] )
```

**Usage**
- A document must be open in Edit mode with the insertion point in a rich text field
- A form, subform, or page must be open in Design mode.

This command does not work on the Web.

---

**Language cross-reference**
Alignment property of LotusScript NotesRichTextParagraphStyle class
Alignment property of Java RichTextParagraphStyle class

---

**TextColor @Command**
Aligns text along the left margin.

**Syntax**
```
@Command( [TextColor] )
```
Usage

- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode.

This command does not work on the Web.

Language cross-reference
Alignment property of LotusScript NotesRichTextParagraphStyle class

Alignment property of Java RichTextParagraphStyle class

**textAlignNone @Command**

Reverses the previously specified alignment settings.

Syntax

@Command( TextAlignNone )

Usage

- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode.

This command does not work on the Web.

Language cross-reference
Alignment property of LotusScript NotesRichTextParagraphStyle class

Alignment property of Java RichTextParagraphStyle class

**textAlignRight @Command**

Aligns text along the right margin.

Syntax

@Command( TextAlignRight )

Usage

- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode.

This command does not work on the Web.

Language cross-reference
Alignment property of LotusScript NotesRichTextParagraphStyle class

Alignment property of Java RichTextParagraphStyle class
TextBold @Command

Makes the selected and subsequently entered text bold. This command is a toggle; selecting it again removes the bold.

Syntax
@Command( [TextBold] )

Usage

• A document must be open in Edit mode with the insertion point in a rich text field or
• A form, subform, or page must be open in Design mode or
• A textbox must be selected in a navigator, with all the text in the textbox becoming bold or
• A textbox must be selected in a layout region on a form, with all of the text in the textbox becoming bold
• It’s most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference
Bold property of LotusScript NotesRichTextStyle class
Bold property of Java RichTextStyle class

TextBullet @Command

Applies the bullet attribute to selected and subsequently entered text.

Syntax
@Command( [TextBullet] ; onOff )

Parameters

onOff

Number. Optional. Specify "1" to turn the bullet attribute on; specify "0" to turn it off. If you omit this parameter, the command toggles from the current state.

Usage

• A document must be open in Edit mode with the insertion point in a rich text field or
• A form, subform, or page must be open in Design mode or
• It’s most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

TextCycleSpacing @Command

Resets the interline spacing below the selected text to single, one and a half, or double.
Syntax
@Command( [TextCycleSpacing] )

Usage
Text must be selected in a rich text field in a document open in Edit mode or on a form or Page open in Design mode. TextCycleSpacing acts as a three-way toggle that sets the spacing below each line of selected text from single to one and a half, or from one and a half to double, or from double to single. If interline spacing in the selected text is not uniform, TextCycleSpacing sets it to single.

Some text must be selected in a rich text field in a document open in Edit mode or a form open in Design mode.

This command does not work on the Web.

TextEnlargeFont @Command
Increases selected and subsequently entered text to the next larger point size.

Syntax
@Command( [TextEnlargeFont] )

Usage
• A document must be open in Edit mode with the insertion point in a rich text field
  or
• A form, subform, or page must be open in Design mode
• It’s most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference
FontSize property of LotusScript NotesRichTextStyle class
FontSize property of Java RichTextStyle class

TextFont @Command
Displays the Properties box for the current text, where the user can select the typeface, size, color, and style attributes.

Syntax
@Command( [TextFont] )

Usage
• A document must be open in Edit mode with the insertion point in a rich text field
  or
• A form, subform, or page must be open in Design mode
• It’s most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.
Language cross-reference
CreateRichTextStyle method of LotusScript NotesSession class
createRichTextStyle method of Java Session class

TextItalic @Command
Italicizes the selected and subsequently entered text. This command is a toggle; selecting it a second time removes the italics.

Syntax
@Command( [TextItalic] )

Usage
- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode
  or
- A textbox must be selected in a navigator, with all the text in the textbox becoming italics
  or
- A textbox must be selected in a layout region on a form, with all the text in the textbox becoming italics
- It’s most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference
Italic property of LotusScript NotesRichTextStyle class
Italic property of Java RichTextStyle class

TextNormal @Command
Removes all style attributes from selected and subsequently entered text.

Syntax
@Command( [TextNormal] )

Usage
- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode
  or
- A textbox must be selected in a navigator, with all the text in the textbox going back to plain text
  or
- A textbox must be selected in a layout region on a form, with all the text in the textbox going back to plain text
- It’s most convenient to use a toolbar button to invoke this command if the text is already selected
This command does not work on the Web.

**TextNumbers @Command**

Applies the numbers attribute to selected and subsequently entered text.

**Syntax**

@Command( [TextNumbers] ; onOff )

**Parameters**

onOff

Number. Optional. Specify "1" to turn the numbers attribute on; specify "0" to turn it off. If you omit this parameter, the command toggles from the current state.

**Usage**

- A document must be open in Edit mode with the insertion point in a rich text field
  - or
- A form, subform, or page must be open in Design mode
- It's most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

**TextOutdent @Command**

Outdents selected and subsequently entered text by narrowing its left margin.

**Syntax**

@Command( [TextOutdent] )

**Usage**

- A document must be open in Edit mode with the insertion point in a rich text field
  - or
- A form, subform, or page must be open in Design mode
- It's most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

**Language cross-reference**

LeftMargin property of LotusScript NotesRichTextParagraphStyle class

LeftMargin property of Java RichTextParagraphStyle class

**TextParagraph @Command**

Displays the Paragraph Alignment panel of the Text Properties box.

**Syntax**

@Command( [TextParagraph] )
Usage

- A document must be open in Edit mode with the insertion point in a rich text field or
- A form, subform, or page must be open in Design mode
- You can use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference

Alignment property of LotusScript NotesRichTextParagraphStyle class

Alignment property of Java RichTextParagraphStyle class

TextParagraphStyles @Command

Displays the Paragraph Styles panel of the Text Properties box.

Syntax

@Command( [TextParagraphStyles] )

Usage

- A document must be open in Edit mode with the insertion point in a rich text field or
- A form, subform, or page must be open in Design mode
- You can use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference

Alignment property of LotusScript NotesRichTextParagraphStyle class

Alignment property of Java RichTextParagraphStyle class

TextPermanentPen @Command

Toggles the permanent pen.

Syntax

@Command( [TextPermanentPen] ; onOff )

Parameters

onOff

Number. Specify "1" to turn permanent pen on; specify "0" to turn it off. Omit this parameter to toggle the permanent pen.

Usage

- A document must be open in Edit mode with the insertion point in a rich text field or
- A form, subform, or page must be open in Design mode
• You can use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference
IsDefault property of LotusScript NotesRichTextStyle class

TextReduceFont @Command
Decreases the selected text to the next smaller point size.

Syntax
@Command( [TextReduceFont] )

Usage
• A document must be open in Edit mode with the insertion point in a rich text field
  or
• A form, subform, or page must be open in Design mode
• It’s most convenient to use a toolbar button to invoke this command if the text is already selected

This command does not work on the Web.

Language cross-reference
FontSize property of LotusScript NotesRichTextStyle class
FontSize property of Java RichTextStyle class

TextSetFontColor @Command
Displays selected or subsequently entered text using the specified color.

Syntax
@Command( [TextSetFontColor] ; [ color ] )

Parameters
[ color ]

The name of the color you want your text to be. The available colors are:
• Black
• Gray
• Red
• DarkRed
• Green
• DarkGreen
• Blue
• DarkBlue
• Magenta
• DarkMagenta
• Yellow
• Brown
• Cyan
• DarkCyan
• White

Usage
• A document must be open in Edit mode with the insertion point in a rich text field
  or
• A form, subform, or page must be open in Design mode

It is most convenient to use a toolbar button to invoke this command if the text is already selected.

This command does not work on the Web.

Language cross-reference
NotesColor property of LotusScript NotesRichTextStyle class
Color property of Java RichTextStyle class

Examples: TextSetFontColor
1. This formula sets the text color to dark cyan.
   @Command([TextSetFontColor]; [DarkCyan])

TextSetFontFace @Command
Displays selected or subsequently entered text using the specified typeface.

Syntax
@Command( [TextSetFontFace] ; typeface )

Parameters
  typeface
  Text. The name of the font you want. The list of available typefaces depends upon the platform you are using, and whether you are using any add-in typeface software. Click the left-most tab on the status bar for a list of available typefaces.

Usage
• A document must be open in Edit mode with the insertion point in a rich text field
  or
• A form, subform, or page must be open in Design mode

It is most convenient to use a toolbar button to invoke this command if the text is already selected.

This command does not work on the Web.

Language cross-reference
NotesFont property of LotusScript NotesRichTextStyle class
Font property of Java RichTextStyle class
Examples: **TextSetFontFace**

1. This formula sets the font to Courier.
   
   ```
   @Command([TextSetFontFace]; "Courier")
   ```

2. This formula, when added to the "Apply font" hotspot button, applies the font a user selects from the "fonts" Dialog list field (which derives its list of fonts by choosing to Use formula for choices and entering @FontList as the formula) to the text the user enters or highlights in the "Body" Rich Text field. If no font was selected from the dialog list, an error message displays which tells the user to select one.
   
   ```
   @Command([EditGoToField];"Body");
   @Command([EditSelectAll]);
   result := @Command([TextSetFontFace];size);
   @If(@IsError(result);Prompt([Ok];"Error encountered";"You must enter a font size first");result)
   ```

---

**TextSetFontSize @Command**

Displays text using the specified point size.

**Syntax**

```
@Command( [TextSetFontSize]; size )
```

**Parameters**

`size`

Text. The font size you want. If you specify a nonexistent size, Notes uses the closest match.

**Usage**

- A document must be open in Edit mode with the insertion point in a rich text field
- or
- A form, subform, or page must be open in Design mode

It is most convenient to use a toolbar button to invoke this command if the text is already selected.

This command does not work on the Web.

**Language cross-reference**

FontSize property of LotusScript NotesRichTextStyle class

FontSize property of Java RichTextStyle class

---

**Examples: **TextSetFontSize**

This formula sets the font size to 10 points.

```
@Command([TextSetFontSize]; "10")
```
Usage
Text must be selected in a rich text field in a document open in Edit mode or on a form or page open in Design mode.

This command does not work on the Web.

Language cross-reference
InterLineSpacing property of LotusScript NotesRichTextParagraphStyle class
InterLineSpacing property of Java RichTextParagraphStyle class

TextSpacingOneAndAHalf @Command
Sets interline spacing below the selected text to one and a half.

Syntax
@Command( [TextSpacingOneAndAHalf] )

Usage
Text must be selected in a rich text field in a document open in Edit mode or on a form or page open in Design mode.

This command does not work on the Web.

Language cross-reference
InterLineSpacing property of LotusScript NotesRichTextParagraphStyle class
InterLineSpacing property of Java RichTextParagraphStyle class

TextSpacingSingle @Command
Sets interline spacing below the selected text to single.

Syntax
@Command( [TextSpacingSingle] )

Usage
Text must be selected in a rich text field in a document open in Edit mode or on a form or page open in Design mode.

This command does not work on the Web.

Language cross-reference
InterLineSpacing property of LotusScript NotesRichTextParagraphStyle class
InterLineSpacing property of Java RichTextParagraphStyle class

TextUnderline @Command
Underlines selected and subsequently entered text. This command is a toggle; selecting it a second time removes the underlining.
Syntax
@Command([TextUnderline])

Usage
- A document must be open in Edit mode with the insertion point in a rich text field
  or
- A form, subform, or page must be open in Design mode
  or
- A textbox must be selected in a navigator, with all of the text in the textbox being underlined
  or
- A textbox must be selected in a layout region on a form, with all of the text in the textbox being underlined

It’s most convenient to use a toolbar button to invoke this command if the text is already selected.

This command does not work on the Web.

Language cross-reference
Underline property of LotusScript NotesRichTextStyle class

Underline property of Java RichTextStyle class

ToolsCall @Command
Displays the Call Server dialog box, where the user can select a server to dial in to.

Syntax
@Command([ToolsCall])

Usage
This command does not work on the Web.

Language cross-reference
ServerName property of LotusScript NotesAgent class

ServerName property of Java Agent class

ToolsCategorize @Command
Categorizes the current document.

Syntax
@Command([ToolsCategorize])

or

@Command([ToolsCategorize]; category)

Parameters
category
Text. Optional. The name of the category you want to put the document in. You can only list a single category name.

**Usage**

- If you include `category`, the selected documents are moved to that category. If you don’t include `category`, Notes/Domino displays the Categorize dialog box so the user can select a category.
  - In a view, all selected documents are categorized.
  - In a document in Read or Edit mode, only that document is categorized.
- The current view must be a categorized view where the first categorized column sorts on a field named "categories"

This command does not work on the Web.

**Language cross-reference**

Categorize method in LotusScript NotesUIDocument class

**Examples: ToolsCategorize**

1. This formula displays the Categorize dialog box.
   ```
   @Command([ToolsCategorize])
   ```
2. This formula moves the selected documents to the "Weekly Status Report" category.
   ```
   @Command([ToolsCategorize]; "Weekly Status Reports")
   ```

---

**ToolsHangUp @Command**

Displays the Hang Up dialog box.

**Syntax**

```
@Command( [ToolsHangUp] )
```

**Usage**

This command does not work on the Web.

---

**ToolsMarkAllRead @Command**

Marks all of the documents in a database as read.

**Syntax**

```
@Command( [ToolsMarkAllRead] )
```

**Usage**

- A database must be open at the view or folder level
  - or
- A document must be open in Read or Edit mode

**Note:** In some databases, there is no difference in the display of read and unread documents in views and folders.

This command does not work on the Web.
**ToolsMarkAllUnread @Command**
Marks all of the documents in a database as unread.

**Syntax**
@Command( [ToolsMarkAllUnread] )

**Usage**
- A database must be open at the view level
  or
- A document must be open in Read or Edit mode

*Note:* In some databases, there is no difference in the display of read and unread documents in views and folders.

This command does not work on the Web.

**ToolsMarkSelectedRead @Command**
In a view or folder, marks all of the selected documents as read.

**Syntax**
@Command( [ToolsMarkSelectedRead] )

**Usage**
- A database must be open at the view level
  or
- A document must be open in Read or Edit mode

*Note:* In some databases, there is no difference in the display of read and unread documents in views and folders.

This command does not work on the Web.

**ToolsMarkSelectedUnread @Command**
In a view or folder, marks all of the selected documents as unread.

**Syntax**
@Command( [ToolsMarkSelectedUnread] )

**Usage**
- A database must be open at the view level
  or
- A document must be open in Read or Edit mode

*Note:* In some databases, there is no difference in the display of read and unread documents in views and folders.

This command does not work on the Web.
ToolsRefreshAllDocs @Command

Refreshes the fields of all the documents in a view or folder.

Syntax
@Command( [ToolsRefreshAllDocs] )

Usage
A database must be open at the view or folder level.

This command does not work on the Web.

Language cross-reference
Refresh method of LotusScript NotesView class
refresh method of Java View class

ToolsRefreshSelectedDocs @Command

Refreshes the fields of all the selected documents in a view or folder.

Syntax
@Command( [ToolsRefreshSelectedDocs] )

Usage
A database must be open at the view or folder level and at least one document must be selected.

This command does not work on the Web.

ToolsReplicate @Command

Displays the Replicate dialog box, where the user can choose to replicate the current database using the options defined in Replicator or using special one-time options.

Syntax
@Command( [ToolsReplicate] ; repMethod )

Parameters
repMethod

Number. Optional. A value of "1" specifies replication using the options that the user can set. A value of "0" specifies replication using the options defined in Replicator. If this parameter is omitted, Notes/Domino prompts the user to choose between these two replication methods.

Usage
A database must be open or selected on the workspace.

This command does not work on the Web.
Language cross-reference
Replicate method of LotusScript NotesDatabase class

replicate method of Java Database class

ToolsRunBackgroundMacros @Command

Runs all of the database’s scheduled agents, regardless of when they are scheduled to run. The agents will then run as usual at their regularly scheduled times.

Syntax
@Command( [ToolsRunBackgroundMacros] )

Usage
This command executes after all @functions. Use @Command([RunScheduledAgents]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

A database must be open or selected on the workspace.

This command does not work on the Web.

ToolsRunMacro @Command

Executes a specified agent.

Syntax
@Command( [ToolsRunMacro] ; agent )

Parameters
agent

Text. Optional. The name of the agent you want to run.

If you omit the agent name, Notes displays a list of Agents in the database, where the user can select which agent to run.

Usage
This command executes after all @functions. Use @Command([RunAgent]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

When specifying a hidden agent, include the parentheses as shown below.
@Command([ToolsRunMacro];"(hiddenagentname)")

You can use this command in Web applications.

Language cross-reference
Run method of LotusScript NotesAgent class

run method of Java Agent class
Example: ToolsRunMacro
This formula runs the agent named "PurgeObsoleteRecords."
@Command([ToolsRunMacro];"PurgeObsoleteRecords")

ToolsScanUnreadChoose @Command
Displays the Scan Unread Preferred Setup dialog box, where the user can select the preferred databases to be scanned for unread documents.

Syntax
@Command( [ToolsScanUnreadChoose] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsScanUnreadPreferred @Command
Displays the Scan Unread dialog box, where the user can see unread counts for each of the user preferred databases.

Syntax
@Command( [ToolsScanUnreadPreferred] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsScanUnreadSelected @Command
Opens the selected database to the first unread document. The user can then navigate to subsequent unread documents.

Syntax
@Command( [ToolsScanUnreadSelected] )

Usage
- If one database is selected on the workspace, Notes/Domino opens the first unread document in the database
- If multiple databases are selected, Notes/Domino displays the Scan Unread dialog box, where the user can see the unread count for each database
- If no database is selected on the workspace, Notes/Domino displays the Scan Unread dialog box, which allows you to select a database with unread documents to open

This command does not work on the Web.

ToolsSetupLocation @Command
Opens the current location document in your Personal Address Book in Edit mode, allowing you to change your home server, mail database location, and time zone.
Syntax
@Command( [ToolsSetupLocation] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsSetupMail @Command
Displays the Mail section of the User Preferences dialog box, where the user can indicate how often Notes should check for new mail, whether to sign and encrypt outgoing mail, and so on.

Syntax
@Command( [ToolsSetupMail] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsSetupPorts @Command
Displays the Ports section of the User Preferences dialog box, where the user can enable and disable network ports.

Syntax
@Command( [ToolsSetupPorts] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsSetupUserSetup @Command
Displays the Basics section of the User Preferences dialog box, where the user defines Notes startup options.

Syntax
@Command( [ToolsSetupUserSetup] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsSmartIcons @Command
Displays the Toolbar Preferences dialog box where you can create new toolbars and customize the content and display of existing toolbars.

Note: In pre-Release 6 clients, displays the SmartIcons dialog box where the user can create SmartIcons and define SmartIcon sets.
Syntax
@Command( [ToolsSmartIcons] )

Usage
You can also access this dialog box manually, by clicking File - Preferences - Toolbar Preferences.

This can be used anywhere within Notes/Domino except from within a dialog box. This command does not work on the Web.

ToolsSpellCheck @Command

Syntax the Notes spell checker.

Syntax
@Command( [ToolsSpellCheck] )

Usage
A document must be open in Edit mode.

This command does not work on the Web.

Language cross-reference
SpellCheck method in LotusScript NotesUIDocument class

ToolsUserLogoff @Command

Logs the user off of all Domino servers. Reconnecting to a Domino server requires the user Notes ID and, if one is set, password.

Syntax
@Command( [ToolsUserLogoff] )

Usage
This can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

UserIDCertificates @Command

Displays the Certificates section of the User ID dialog box. If the user ID is password-protected, the user must enter the password before displaying the dialog box.

Syntax
@Command( [UserIDCertificates] )

Usage
This command can be used almost anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

UserIDClearPassword @Command

Displays the Enter Password dialog box where the password associated with the user ID can be removed.
Syntax
@Command( [UserIDClearPassword] )

Usage
This can be used almost anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

UserIDCreateSafeCopy @Command
Displays the Enter Safe Copy ID File Name dialog box, where the user defines a file name for storing a safe copy of the Notes/Domino user ID. The safe copy contains only the user name and public key. The private key and all certificates are omitted from the safe copy.

Syntax
@Command( [UserIDCreateSafeCopy] )

Usage
This command can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

UserIDEncryptionKeys @Command
Displays the Encryption section of the User ID dialog box. If the user ID is password–protected, the user must enter the password before displaying the dialog box.

Syntax
@Command( [UserIDEncryptionKeys] )

Usage
This command can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

UserIDInfo @Command
Displays the Basics section of the User ID Information dialog box.

Syntax
@Command( [UserIDInfo] )

Usage
This command can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

Language cross-reference
GetUserInfo method of LotusScript NotesRegistration class
getUserInfo method of Java Registration class
UserIDMergeCopy  @Command
Displays the Choose User ID to Merge into Current ID dialog box. This lets the user select an updated user ID (for example, one that has been certified by an administrator) and merge it into the existing user ID (thus retaining the information already stored in the existing ID). If the user ID is password-protected, Notes/Domino requires the user to enter the password before displaying the dialog box.

Syntax
@Command( UserIDMergeCopy )

Usage
This command can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

UserIDSetPassword  @Command
Displays the Set Password dialog box where the user can enter a new password for the Notes/Domino ID. If a password already exists, the user must enter the existing password before specifying a new one.

Syntax
@Command( UserIDSetPassword )

Usage
This command can be used anywhere in Notes/Domino except from within a dialog box. This command does not work on the Web.

Language cross-reference
RegisterNewCertifier method of LotusScript NotesRegistration class
registerNewCertifier method of Java Registration class

UserIDSwitch  @Command
Displays the Choose User ID to Switch To dialog box. If the selected ID is password-protected, Notes/Domino next displays the Enter Password dialog box, where the user must enter the correct password.

Syntax
@Command( UserIDSwitch )

Usage
This command can be used in hotspot actions or buttons and view or form action formulas.

This command does not work on the Web.

Language cross-reference
SwitchToID method of LotusScript NotesRegistration class
switchToID method of Java Registration class
**V3EditNextField @Command**

In a document in Edit mode, moves the insertion point to the next editable field.

**Syntax**

@Command( [V3EditNextField] )

**Usage**

- Use V3EditNextField to enable your application to run under Release 3 of Lotus Notes
- When used in a SmartIcons formula (pre-Release 6), V3EditNextField moves the insertion point to the next editable field in the document, working left to right and top to bottom
- When used in a button that’s built into the form, the first occurrence of V3EditNextField always moves the insertion point to the first field in the document. You must add an additional @Command([V3EditNextField]) to move to each subsequent field (or you could use a related command such as EditUp or EditDown)
- A document must be open in Edit mode

This command does not work on the Web.

**Language cross-reference**

GoToNextField method of LotusScript NotesUIDocument class

---

**V3EditPrevField @Command**

In a document in Edit mode, moves the insertion point to the first editable field.

**Syntax**

@Command( [V3EditPrevField] )

- Use V3EditPrevField to enable your application to run under Release 3 of Notes
- When used in a SmartIcons formula (pre-Release 6), V3EditPrevField moves the insertion point to the first editable field in the document, working right to left and bottom to top
- When used in a button that’s built into the form, the first occurrence of V3EditPrevField always moves the insertion point to the first field in the document. You must add an additional @Command([V3EditPrevField]) to move to the last field on the form, and again for each additional field (or you could use a related command such as EditUp or EditDown)
- A document must be open in Edit mode

This command does not work on the Web.

**Language cross-reference**

GoToPrevField method of LotusScript NotesUIDocument class

---

**ViewArrangeIcons @Command**

Aligns database icons on the current workspace page from left to right and top to bottom.

**Syntax**

@Command( [ViewArrangeIcons] )
Usage
This command can only be used on the workspace.

This command does not work on the Web.

ViewBelowFolders  @Command
Places the view pane at the bottom of the screen and the folder pane at the top.

Syntax
@Command( [ViewBelowFolders] )

Usage
A database must be open at the view or folder level.

This command does not work on the Web.

ViewBesideFolders  @Command
Places the view pane at the right of the screen and the folder pane at the left.

Syntax
@Command( [ViewBesideFolders] )

Usage
A database must be open at the view or folder level.

This command does not work on the Web.

ViewCertify  @Command
Displays the Choose Certifier ID dialog box, in which you specify the certification to be renewed for the selected user or users.

Syntax
@Command( [ViewCertify] )

Usage
ViewCertify is available once you have opened a server Domino Directory to the People view and selected one or more Person documents. View Certify enables you to initiate recertification of the selected user or users by the Administration Process.

This command does not work on the Web.

Language cross-reference
Recertify method of LotusScript NotesRegistration class
recertify method of Java Registration class
**ViewChange @Command**

Switches to the specified view or folder within the current database or, if a view or folder is not specified, displays the View menu so the user can select a view.

**Syntax**

@Command( [ViewChange] )

or

@Command( [ViewChange]; viewName )

**Parameters**

*viewName*

Text. Optional. The name of the view or folder you want to switch to.

**Usage**

This command executes after all @functions. Use @Command([SwitchView]) to execute immediately. See the Order of evaluation for formula statements topic for more details.

This command doesn’t work from within a document or form, so you cannot use it to view a document through another form. Use SwitchForm instead.

You can use this command in Web applications, but you must use the viewName parameter.

**Language cross-reference**

OpenView method of LotusScript NotesUIDatabase class

**Examples: ViewChange**

This formula switches to the By Author view.

@Command([ViewChange]; "By Author")

---

**ViewCollapse @Command**

Collapses all levels of subcategories, documents, and responses within the current category, so only the topmost category name shows. If the view or folder has a response hierarchy but is not categorized, ViewCollapse collapses all levels of response documents under the current main document.

**Syntax**

@Command( [ViewCollapse] )

**Usage**

- A database must be open at a view or folder that uses categories and/or a response hierarchy

  **Note:** The following feature is new with Release 5.

- This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder
ViewCollapseAll @Command

Collapses all levels of categories, subcategories, documents, and responses within a view or folder so that only the topmost level of category names appears. If the view or folder has a response hierarchy but is not categorized, ViewCollapseAll collapses all levels of response documents under the main documents.

Syntax
@Command( [ViewCollapseAll] )

Usage
• A database must be open at the view or folder level
• You can use this command in Web applications

Note: The following feature is new with Release 5.
• This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder

Examples: @Command([ViewCollapseAll])
This code, when added as the formula for an Action to a view, collapses all main and response documents per category in views that have categories and collapses all response documents under main documents in views that do not have categories.
@Command([ViewCollapseAll])

ViewExpand @Command

Expands one level of subcategories, documents, and responses within the current category. If the view or folder has a response hierarchy but is not categorized, ViewExpand expands all levels of response documents under the current main document.

Syntax
@Command( [ViewExpand] )

Usage
• A database must be open at a view or folder that uses categories and/or a response hierarchy

Note: The following feature is new with Release 5.
• This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder

ViewExpandAll @Command

Expands all levels of categories, subcategories, documents, and responses within the view or folder.

Syntax
@Command( [ViewExpandAll] )

Usage
• A database must be open at a view or folder that uses categories, a response hierarchy, or both. If the view or folder has a response hierarchy but is not categorized, ViewExpandAll expands all levels of response documents under the main documents
• You can use this command in Web applications

Chapter 7. Formula Language @Commands A-Z  623
**Note:** The following feature is new with Release 5.
- This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder

---

**ViewExpandWithChildren @Command**
Expands all levels of subcategories, documents, and responses within the selected category.

**Syntax**

```
@Command( [ViewExpandWithChildren] )
```

**Usage**
This command does not work on the Web.

---

**ViewHorizScrollBar @Command**
Toggles display of the horizontal scroll bar in a view or folder.

**Note:** ViewHorizScrollbar is not supported in Release 6; with Release 6, the view automatically supplies a scrollbar when necessary. In pre-release 6 versions, ViewHorizScrollbar is not supported under OS/2 and on the Macintosh.

**Syntax**

```
@Command( [ViewHorizScrollBar] )
```

**Usage**
A database must be open to a view or folder.

This command does not work on the Web.

---

**ViewMoveName @Command**
Displays the Choose Certifier ID dialog box, which allows you to specify the organizational unit for which you want to certify the selected user.

**Syntax**

```
@Command( [ViewMoveName] )
```

**Usage**

ViewMoveName is available once you have opened a server Domino Directory to the People view and selected a Person document. ViewMoveName initiates the process of changing the user’s hierarchical name through the Administration Process.

This command does not work on the Web.

---

**Language cross-reference**

OrgUnit property of LotusScript NotesRegistration class

OrgUnit property of Java Registration class
ViewNavigatorsFolders @Command
Displays the Folders and Views navigators in the navigator pane and the view or folder that the user most recently selected.

**Syntax**
@Command( [ViewNavigatorsFolders] )

**Usage**
A database must be open to a view or folder.
This command does not work on the Web.

ViewNavigatorsNone @Command
Hides the navigator pane.

**Syntax**
@Command( [ViewNavigatorsNone] )

**Usage**
A database must be open to a view or folder.
This command does not work on the Web.

ViewRefreshFields @Command
Recalculates the fields in the current document or updates the current view or folder.

**Syntax**
@Command( [ViewRefreshFields] )

**Usage**
- A document must be open in Edit mode
  or
- A database must be open at the view or folder level
- When you execute this command on the Web in the current document, it recalculates all fields formulas without closing the document. On the Web, you can only use this command for databases where you have enabled JavaScript to generate the pages except as noted below

  **Note:** The following feature is new with Release 5.
- This @command works on the Web if "Use applet in the browser" is in effect for the implementing view or folder

**Language cross-reference**
ViewRefresh method of LotusScript NotesUIWorkspace class
Refresh method of LotusScript NotesUIDocument class
**ViewRefreshUnread @Command**

Updates the unread counts on all database icons displayed on the current workspace page.

**Syntax**

@Command( [ViewRefreshUnread] )

**Usage**

A database must be open at the workspace.

This command does not work on the Web.

---

**ViewRenamePerson @Command**

Displays the Certify Selected Entries dialog box, which enables you to upgrade the selected user name to a hierarchical name, change the user’s common name, or change the user’s hierarchical name.

**Syntax**

@Command( [ViewRenamePerson] )

**Usage**

ViewRenamePerson is available once you have opened a server Domino Directory to the People view and selected a Person document. ViewRenamePerson initiates the change you specify through the Administration Process.

This command does not work on the Web.

---

**ViewShowFieldHelp @Command**

Shows field Help, so that the Help description for the current field (if available) appears on the status bar while the user composes or edits a document. (Field Help is a design option specified in the Field Properties box.) This command is a toggle; selecting it a second time hides the field Help.

**Syntax**

@Command( [ViewShowFieldHelp] )

**Usage**

A document must be open in Edit mode.

This command does not work on the Web.

---

**Language cross-reference**

FieldHelp property of LotusScript NotesUIDocument class

---

**ViewShowObject @Command**

Displays the "object delimiter" -- a dotted frame surrounding an OLE/LEL object within a field. This command is a toggle; selecting it a second time hides the object delimiter so the OLE/LEL object looks like regular, editable data.

**Syntax**

@Command( [ViewShowObject] )
Usage

This command is necessary only when a document is open in Read mode; the object delimiter always displays in Edit mode.

This command does not work on the Web.

**ViewShowOnlyCategories @Command**

Collapses the view or folder so that only category and subcategory names show up. This command is a toggle; selecting it a second time expands all category levels.

*Note:* This view setting will remain active for all views until it is toggled off. If a view is not categorized, and ViewShowOnlyCategories is toggled on, no documents will be displayed.

**Syntax**

```plaintext
@Command( [ViewShowOnlyCategories] )
```

**Usage**

A database must be open at the view level.

This command does not work on the Web.

**Language cross-reference**

CreateViewNavMaxLevel method of LotusScript NotesView class

createViewNavMaxLevel method of Java View class

**ViewShowOnlySearchResults @Command**

Shows the results of a full-text search as selected documents in a view or folder. This command is a toggle. Instead of listing only the documents that satisfy the search query (the default for a full-text search), the view or folder lists all the documents it normally lists, with a check mark in front of those documents that satisfy the search query.

**Syntax**

```plaintext
@Command( [ViewShowOnlySearchResults] )
```

**Usage**

A database must be open at the view or folder level, and a full-text search must have been run.

This command does not work on the Web.

**Language cross-reference**

FTSearch method of LotusScript NotesView class

FTSearch method of Java View class

**ViewShowOnlySelected @Command**

Displays only the selected documents or categories in the view or folder (those documents with a check mark). This command is a toggle; selecting it a second time displays all documents or categories.
Syntax
@Command( [ViewShowOnlySelected] )

Usage
A database must be open at the view or folder level.

This command does not work on the Web.

ViewShowOnlyUnread  @Command
Displays only the unread documents in the view or folder. This command is a toggle; selecting it again displays all documents in the view or folder.

Syntax
@Command( [ViewShowOnlyUnread] )

Usage
A database must be open at the view or folder level.

This command does not work on the Web.

ViewShowPageBreaks  @Command
Displays a line representing each page break in the document. The page breaks indicate where Notes will end each page when the document is printed on the currently selected printer. This command is a toggle; selecting it a second time suppresses the display of automatic page breaks (manual page breaks will still be displayed).

Syntax
@Command( [ViewShowPageBreaks] )

Usage
A document must be open in Edit mode.

This command does not work on the Web.

ViewShowRuler  @Command
Toggles the display of the ruler while a document is open in Edit mode.

Syntax
@Command( [ViewShowRuler] )

Usage
This command does not work on the Web.

ViewShowSearchBar  @Command
Toggles the display of the full-text search bar at the top of the view. If the database does not have a full-text index, the search bar displays but is not usable.
Syntax
@Command( [ViewShowSearchBar] ; onOff )

Parameters

onOff

Number. Specify "1" to show the search bar. Specify "0" to hide it. Omit this parameter to toggle the display of the search bar.

Usage

You can use this command in Web applications. In the Notes client, this command can only be used with views and folders.

ViewShowServerNames @Command

Toggles the display of server names on the database icons in the user’s workspace.

Syntax
@Command( [ViewShowServerNames] )

Usage

This command does not work on the Web.

ViewShowUnread @Command

Toggles the display of the unread document count on each database icon on the user’s workspace.

Syntax
@Command( [ViewShowUnread] )

Usage

This command does not work on the Web.

ViewSwitchForm @Command

Changes the form used to display the current document.

Syntax
@Command( [ViewSwitchForm] ; formName )

Parameters

formName

Text. Optional. The name of the form you want to switch to.

With no parameter, ViewSwitchForm displays a dialog box with a list of forms available in the current database.

Usage

This command executes after all @functions. Use @Command([SwitchForm]) to execute immediately. See the Order of evaluation for formula statements topic for more details.
This command does not work on the Web.

**WindowCascade @Command**

Resizes all open Notes/Domino windows to less than 50% of their maximum window size and layers them in a cascading stack. Because the layered stack is cascaded, the title bar of each window is visible behind the stack’s top-most window. To display a window that is lower in the stack, you can bring it to the top by clicking its title bar.

*Note:* This @command is new with Release 6.

**Syntax**

@Command( [WindowCascade] )

**Usage**

This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

**WindowMaximize @Command**

Maximizes the active Notes/Domino window (the window whose title bar is highlighted).

**Syntax**

@Command( [WindowMaximize] )

**Usage**

This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

**WindowMaximizeAll @Command**

Maximizes all open Notes/Domino windows. Notes/Domino windows include the Designer and Administration client windows as well as any of the Notes Help windows.

**Syntax**

@Command( [WindowMaximizeAll] )

**Usage**

This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

**WindowMinimize @Command**

Minimizes the active Notes/Domino window (the window whose title bar is highlighted).

**Syntax**

@Command( [WindowMinimize] )

**Usage**

This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.
WindowMinimizeAll @Command

Minimizes all open Notes/Domino windows. Notes/Domino windows include the Designer and Administration client windows as well as any of the Notes Help windows.

Syntax
@Command( [WindowMinimizeAll] )

Usage
This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

WindowNext @Command

Maximizes the Notes window whose taskbar button is to the right of the current window’s taskbar button or, if the windows are cascaded, moves the next window in the stack to the top of the stack.

Note: This @command is new with Release 6.

Syntax
@Command( [WindowNext] )

Usage
This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

WindowRestore @Command

Restores the active window to its former size (before it was maximized or minimized).

Syntax
@Command( [WindowRestore] )

Usage
This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

If you use this command to restore a window that was maximized or minimized using @Command([WindowMaximizeAll]) or @Command([WindowMinimizeAll]), only the current window is restored, the other windows that were changed remain changed. You cannot use this command to restore a cascaded or tiled window to its former size.

WindowTile @Command

Resizes all open Notes/Domino windows to display them all at once. The open windows are tiled across the screen until they fill the background. If two windows are currently open, Notes displays them side-by-side. If four windows are open, Notes displays one in each of the four quadrants of the window.

Note: This @command is new with Release 6.

Syntax
@Command( [WindowTile] )
Usage
This command can be used anywhere in Notes/Domino, except for an open dialog box. It does not work on the Web.

If you tile two or more windows, then close all but one of them, the remaining window does not readjust its size. Use @Command([WindowMaximize]) to resize it to fit the screen.

---

WindowWorkspace @Command
Displays the Notes/Domino workspace as the active (topmost) window.

Syntax
@Command( [WindowWorkspace] )

Usage
This command can be used anywhere in Notes/Domino. It does not function in Web applications.

---

WorkspaceProperties @Command
Displays the Properties box for the Notes/Domino workspace.

Syntax
@Command( [WorkspaceProperties] )

Usage
The Notes/Domino workspace must be open in the current window. This command does not work on the Web.

---

WorkspaceStackReplicaIcons @Command
For databases on the workspace that are replicas of one another, stacks them into a single icon. The command is a toggle; selecting it a second time unstacks the replica icons and displays each icon individually.

Syntax
@Command( [WorkspaceStackReplicaIcons] )

Usage
The Notes/Domino workspace must be open in the current window. This command does not work on the Web.

---

ZoomPreview @Command
Toggles the ZoomPreview setting in a view.

Syntax
@Command( [ZoomPreview]; size )

Parameters
size
Text ("0" or "1"). Optional. Specify "1" to zoom the preview pane to the maximum size. Specify "0" to return the preview pane to its previous size.

**Usage**

ZoomPreview enlarges the preview pane when enabled, and shrinks it when disabled. The preview pane must be open. Using the "0" parameter is not the same as shrinking the window to its minimum size. The window shrinks to its previous size; if the current size and previous size are the same, it does not shrink.

This command does not work on the Web.
Index

Special characters
[AddBookmark] command formula language 477
[AddDatabase] command formula language 478
[AddDatabaseRepID] command formula language 479
[AddDocMCreatContactList] command formula language 479
[AdminCertify] command formula language 480
[AdminCreateGroup] command formula language 480
[AdminCrossCertifyIDFile] command formula language 480
[AdminCrossCertifyKey] command formula language 481
[AdminDatabaseAnalysis] command formula language 481
[AdminDatabaseQuotas] command formula language 481
[AdminIDFileClearPassword] command formula language 482
[AdminIDFileExamine] command formula language 482
[AdminIDFileSetPassword] command formula language 482
[Administration] command formula language 482
[AdminNewOrganization] command formula language 483
[AdminNewOrgUnit] command formula language 483
[AdminOpenAddressBook] command formula language 483
[AdminOpenCatalog] command formula language 484
[AdminOpenCertLog] command formula language 484
[AdminOpenGroupsView] command formula language 484
[AdminOpenServerLog] command formula language 485
[AdminOpenServersView] command formula language 485
[AdminOpenStatistics] command formula language 485
[AdminOutgoingMail] command formula language 486
[AdminRegisterFromIDFile] command formula language 487
[AdminRegisterServer] command formula language 487
[AdminRegisterUser] command formula language 487
[AdminRemoteUser] command formula language 487
[AdminSendMailTrace] command formula language 488
[AdminStatisticsConfig] command formula language 488
[AdminTraceConnection] command formula language 488
[AgentEdit] command formula language 489
[AgentEnableDisable] command formula language 489
[AgentLog] command formula language 489
[AgentRun] command formula language 490
[AgentSetServerName] command formula language 490
[AgentTestRun] command formula language 491
[AttachmentAttachAll] command formula language 491
[AttachmentLaunch] command formula language 491
[AttachmentProperties] command formula language 492
[AttachmentView] command formula language 492
[CalendarFormat] command formula language 493
[CalendarGoTo] command formula language 493
[CalendarGoTo] command formula language 493
[CheckCalendar] command formula language 494
[ChooseFolders] command formula language 494
[Clear] command formula language 495
[CloseWindow] command formula language 496
[Compose] command formula language 496
[ComposeWithReference] command formula language 498
[CreateAction] command formula language 500
[CreateAgent] command formula language 500
[CreateControlledAccessSection] command formula language 500
[CreateEllipse] command formula language 500
[CreateFolder] command formula language 501
[CreateForm] command formula language 501
[CreateLayoutRegion] command formula language 501
[CreateNavigator] command formula language 502
[CreatePolygon] command formula language 502
[CreatePolyline] command formula language 502
[CreateRectangle] command formula language 502
[CreateRectangularHotspot] command formula language 503
[CreateSection] command formula language 503
[CreateSubform] command formula language 503
[CreateTextbox] command formula language 504
[CreateView] command formula language 504
[DatabaseDelete] command formula language 504
[DatabaseRepSettings] command formula language 505
[DebugLotusScript] command formula language 505
[DesignDocumentInfo] command formula language 505
[DesignFormAttributes] command formula language 506
[DesignFormFieldDef] command formula language 506
[DesignFormNewField] command formula language 506
[DesignForms] command formula language 506
[DesignFormShareField] command formula language 507
[DesignFormUseField] command formula language 507
[DesignFormWindowTitle] command formula language 507
[DesignHelpAboutDocument] command formula language 507
[DesignHelpUsingDocument] command formula language 508
[DesignIcon] command formula language 508
[DesignMacro] command formula language 508
[DesignRefresh] command formula language 508
[DesignReplace] command formula language 509
[DesignSharedFields] command formula language 509
[DesignSynopsis] command formula language 509
[DesignViewAppendColumn] command formula language 509
[DesignViewAttributes] command formula language 510
Index 637
[SectionExpandAll] command
formula language 594
[SectionProperties] command
formula language 594
[SectionRemoveHeader] command
formula language 594
[SendInstantMessage] command
formula language 595
[SetCurrentLocation] command
formula language 595
[ShowHideMContactList] command
formula language 595
[ShowHideLinkPreview] command
formula language 595
[ShowHideParentPreview] command
formula language 596
[ShowHidePreviewPane] command
formula language 596
[ShowProperties] command
formula language 597
[SmartIconsFloating] command
formula language 597
[SmartIconsNone] command
formula language 597
[StyleCycleKey] command
formula language 597
[SwitchForm] command
formula language 598
[SwitchView] command
formula language 598
Web programming 71
[TextAlignCenter] command
formula language 599
[TextAlignFull] command
formula language 599
[TextAlignLeft] command
formula language 599
[TextAlignNone] command
formula language 600
[TextAlignRight] command
formula language 600
[TextBold] command
formula language 601
[TextBullet] command
formula language 601
[TextCycleSpacing] command
formula language 601
[TextEnlargeFont] command
formula language 602
[TextFont] command
formula language 602
[TextItalic] command
formula language 603
[TextNormal] command
formula language 603
[TextNumbers] command
formula language 604
[TextOutdent] command
formula language 604
[TextParagraph] command
formula language 604
[TextParagraphStyles] command
formula language 604
[TextPermanentPen] command
formula language 605
[TextReduceFont] command
formula language 606
[TextSetFontColor] command
formula language 606
[TextSetFontFace] command
formula language 606
[TextSetFontSize] command
formula language 608
[TextSpacingDouble] command
formula language 608
[TextSpacingOneAndAHalf] command
formula language 609
[TextSpacingSingle] command
formula language 609
[TextUnderline] command
formula language 609
[ToolsCall] command
formula language 610
[ToolsCategorize] command
formula language 610
[ToolsHangUp] command
formula language 611
[ToolsMarkAllRead] command
formula language 611
[ToolsMarkAllUnread] command
formula language 612
[ToolsMarkSelectedRead] command
formula language 612
[ToolsMarkSelectedUnread] command
formula language 612
[ToolsRefreshAllDocs] command
formula language 613
[ToolsRefreshSelectedDocs] command
formula language 613
[ToolsReplicate] command
formula language 613
[ToolsRunBackgroundMacros] command
formula language 614
[ToolsRunMacro] command
formula language 614
Web programming 71
[ToolsScanUnreadChoose] command
formula language 615
[ToolsScanUnreadPreferred] command
formula language 615
[ToolsScanUnreadSelected] command
formula language 615
[ToolsSetupLocation] command
formula language 615
[ToolsSetupMail] command
formula language 616
[ToolsSetupPorts] command
formula language 616
[ToolsSmartIcons] command
formula language 616
[ToolsSpellCheck] command
formula language 617
[ToolsUserLogoff] command
formula language 617
[UserIDCertificates] command
formula language 617
[UserIDClearPassword] command
formula language 617
[UserIDCreateSafeCopy] command
formula language 618
[UserIDEncryptionKeys] command
formula language 618
[UserIDInfo] command
formula language 618
[UserIDMergeCopy] command
formula language 619
[UserIDPassword] command
formula language 619
[UserIDSetPassword] command
formula language 619
[UserIDSwitch] command
formula language 619
[V3EditNextField] command
formula language 620
[V3EditPrevField] command
formula language 620
[ViewArrangeIcons] command
formula language 620
[ViewBelowFolders] command
formula language 621
[ViewBesideFolders] command
formula language 621
[ViewCertify] command
formula language 621
[ViewChange] command
formula language 622
Web programming 71
[ViewCollapse] command
formula language 622
Web programming 71
[ViewCollapseAll] command
formula language 623
Web programming 71
[ViewExpand] command
formula language 623
Web programming 71
[ViewExpandAll] command
formula language 623
Web programming 71
[ViewExpandWithChildren] command
formula language 624
[ViewHorizScrollBar] command
formula language 624
[ViewMoveName] command
formula language 624
[ViewNavigatorsFolders] command
formula language 625
[ViewNavigatorsNone] command
formula language 625
[ViewRefreshFields] command
formula language 625
Web programming 71
[ViewRefreshUnread] command
formula language 626
[ViewRenamePerson] command
formula language 626
[ViewShowFieldHelp] command
formula language 626
[ViewShowObject] command
formula language 626
[ViewShowOnlyCategories] command
formula language 627
[ViewShowOnlySearchResults] command
formula language 627
[ViewShowOnlySelected] command
formula language 627
[ViewShowOnlyUnread] command
formula language 628
[ViewShowPageBreaks] command
formula language 628
ECL defined function side-effects 127

Commands defined 127, 227
ECL security and 475
@Power function
  formula language 154, 389
@Prompt function
debbuging with 130
  formula language 390
  getting user input with 141
  side-effects 127
@SetViewInfo function
  formula language 153, 393
@SetTargetFrame function
  formula language 154, 394
@RefreshECL function
  formula language 395
@RegQueryValue function
  formula language 395
@Repeat function
  formula language 153, 397
@Replace function
  formula language 135, 398
@ReplaceSubString function
  formula language 399
@RepeatAsID function
  formula language 160, 400
@Responses function
  formula language 160, 161, 400
@Return function
  formula language 128, 146, 401
@Right function
  formula language 151, 402
@RightBack function
  formula language 151, 403
@Round function
  formula language 154, 404
@Second function
  formula language 156, 405
@Select function
  formula language 407
@ServerAccess function
  formula language 407
@ServerName function
  formula language 407
@Set function
  formula language 410
@SetDocField function
  formula language 164, 411
@SetEnvironment function
  formula language 146, 412
@SetField function
  formula language 161, 413
@SetHTTPHeader function
  formula language 414
  Web programming 68
@SetProfileField function
  formula language 415
@SetTargetFrame function
  formula language 416
@SetViewInfo function
  formula language 417
@Sign function
  formula language 154, 418
@Sin function
  formula language 154, 419
@Sort function
  formula language 135, 419
@Soundex function
  formula language 422
@Sqrt function
  formula language 154, 422
@StatusBar function
  formula language 158, 423
@Subset function
  formula language 135, 423
@Success function
  formula language 146, 424
  Web programming 68
@Sum function
  formula language 154, 424
@Tan function
  formula language 154, 425
@TemplateVersion function
  formula language 426
@Text function
  formula language 148, 426
@TextToNumber function
  formula language 428
@TextToTime function
  formula language 429
@ThisName function
  formula language 430
@ThisValue function
  formula language 431
@Time function
  formula language 156, 431
@TimeMerge function
  formula language 433
@TimeToTextInZone function
  formula language 433
@TimeZoneToText function
  formula language 434
@Today function
  formula language 156, 435
@Tomorrow function
  formula language 156, 436
@ToNumber function
  formula language 437
@ToTime function
  formula language 438
@Transform function
  formula language 135, 438
@Trim function
  formula language 153, 440
@True function
  formula language 440
@Unavailable function
  formula language 441
@UndeleteDocument function
  formula language 442
@Unique function
  formula language 135, 442
@UpdateFormulaContext function
  formula language 443
@UpperCase function
  formula language 153, 444
@URLDecode function
  formula language 444
@URLEncoder function
  formula language 445
@URLGetHeader function
  formula language 446
@URLHistory function
  formula language 447
@URLOpen function
  formula language 448
@URLOpen function (continued)
  Web programming 68
@UrlQueryString function
  formula language 451
@UserAccess function
  formula language 158, 452
@UserName function
  formula language 454
@UserNameLanguage function
  formula language 456
@UserNamesList function
  formula language 158, 457
@UserPrivileges function
  formula language 158, 457
@UserRoles function
  formula language 158, 458
@V2If function
  formula language 459
@V3UserName function
  formula language 459
@V4UserAccess function
  formula language 461
@ValidateInternetAddress function
  formula language 462
@VerifyPassword function
  formula language 464
@Version function
  formula language 158, 464
@ViewShowThisUnread function
  formula language 465
@ViewTitle function
  formula language 160, 466
@WebDbName function
  formula language 467
  Web programming 68
@Weekday function
  formula language 156, 467
@While function
  formula language 156, 467
@Wide function
  formula language 139, 468
@Word function
  formula language 151, 470
@Year function
  formula language 156, 471
@Yes function
  formula language 471
@Yesterday function
  formula language 156, 472
@Zone function
  formula language 156, 472

A
About Database document editing 507
opening 560
Abs function
  formula language 154, 196
Absolute values
  formula language 196
Abstract function
  formula language 197
Accent sensitivity
  sorting 419
Access levels
    formula language 261, 452, 459, 461
    reading 452
Accessed function
    formula language 156, 203
Accessibility
    LotusScript debugger 55
    programmer’s pane 37, 40, 42, 43, 44
ACL
    database 539
    editing 539
    populating a column with 12
Acos function
    formula language 154, 204
Action scripts and formulas
    formula language 134
Actions
    coding 7
    creating 500
AddBookmark command
    formula language 477
AddDatabase command
    formula language 478
AddDatabaseReplID command
    formula language 479
Addition
    formula language 154
Addition operator (+)
    formula language 120
AddToFolder function
    formula language 205
AddToMContactList command
    formula language 479
Adjust function
    formula language 156, 206
AdminCertify command
    formula language 480
AdminCreateGroup command
    formula language 480
AdminCrossCertifyIDFile command
    formula language 480
AdminCrossCertifyKey command
    formula language 481
AdminDatabaseAnalysis command
    formula language 481
AdminDatabaseQuotas command
    formula language 481
AdminIDFileClearPassword command
    formula language 482
AdminIDFileExamine command
    formula language 482
AdminIDFileSetPassword command
    formula language 482
Administration command
    formula language 482
AdminNewOrganization command
    formula language 483
AdminNewOrgUnit command
    formula language 483
AdminOpenAddressBook command
    formula language 483
AdminOpenCatalog command
    formula language 484
AdminOpenCertLog command
    formula language 484
AdminOpenGroupsView command
    formula language 484
AdminOpenServerLog command
    formula language 485
AdminOpenServersView command
    formula language 485
AdminOpenStatistics command
    formula language 485
AdminOpenUsersView command
    formula language 486
AdminOutgoingMail command
    formula language 486
AdminRegisterFromFile command
    formula language 487
AdminRegisterServer command
    formula language 487
AdminRegisterUser command
    formula language 487
AdminRemoteConsole command
    formula language 487
AdminSendMailTrace command
    formula language 488
AdminStatisticsConfig command
    formula language 488
AdminTraceConnection command
    formula language 488
AgentEdit command
    formula language 489
AgentEnableDisable command
    formula language 489
AgentLog command
    formula language 489
AgentRun command
    formula language 490
Agents
    Web programming 79
    Agents, types of
      running on the Web 71
      scheduled 490
    Agents, working with
      checking status of in formula language
      coding 5, 134
      creating 500
      debugging 57, 489
      editing a Java agent 47
      editing in formula language 489
      enabling/disabling 489
      running 490, 591, 592, 614
      running on the Web 490, 491
      viewing the log 489, 490, 491
    AgentSetServerName command
    formula language 490
    AgentTestRun command
    formula language 491
Alarms
    checking 225
    enabling 289
    All function 5, 12
    All function
      formula language 133, 208
    AllChildren function
    formula language 146, 161, 209
    AllDescendants function
    formula language 161, 209
    Alternate names
    getting in formula language 454
    AND operator
    formula language 120
Angles
    calculating in formula language 204, 212, 213, 231, 419, 425
Applets array
    Java 86
Applications
    launching in formula language 342, 491, 537
    testing existence of in formula language 320
Arguments
    Web programming 79
Arithmetic operators
    formula language 123, 154
Ascii function
    formula language 211
Asin function
    formula language 154, 212
ASP environment
    accessing organization directories in 381
Assignment operators
    formula language 120, 121
Atan function
    formula language 154, 212
Atan2 function
    formula language 154, 213
AttachmentDetachAll command
    formula language 491
AttachmentLaunch command
    formula language 491
AttachmentLengths function
    formula language 161, 214
AttachmentModifiedTimes function
    formula language 161, 215
AttachmentNames function
    formula language 161, 216
AttachmentProperties command
    formula language 492
Attachments
    creating 522
    deleting highlighted 495, 513, 515
    detaching. See saving 516
    editing in Designer 492, 597
    getting details of in formula language 214, 215, 216, 217
    launching 491
    saving 491, 516
    viewing 492
Attachments function
    formula language 161, 217
AttachmentView command
    formula language 492
Author function
    formula language 161, 217
Authors
    getting list of in formula language 217
Auto complete
    in script area of programmer’s pane 45
    properties 45

B
Backslash ( )
    formula language 117
Base class box
Imported Java 39

Begin function
formula language 151, 218

Binding
Web services 91
Block statements
in script area of programmer’s
pane 52

Body field
formula language 114

Bold
formula language 601

Bookmarks
creating and filing URLs 477
filing databases with 478, 479

Braces
formula language 117

Breakpoints
LotusScript debugger 55

BrowserInfo function
formula language 219
Web programming 68

Browsers
getting details of in formula
language 219
getting information 68

BusinessDays function
formula language 156, 221

Button object
JavaScript 86
Buttons
coding 9, 134
creating 521
editing 513

C

Calendar views
checking schedules in 494
controlling 493
formatting 493
opening 578
setting focus in 493
Web use of 493

CalendarFormat command
formula language 493
Web programming 71

CalendarGoto command
Web programming 71

CalendarGoTo command
formula language 493
Web programming 71

Carriage returns
inserting with formula language 223, 376
showing/hiding in formula language 533

Cascading
form names 130
view names 130

Catalogs
searching for databases in 484

Categories
displaying only documents in 417
getting details of in formula language 273, 275, 281

Categories (continued)
searching views for in formula language 321
Certificate function
formula language 222
Certificates
accessing user 617
extracting data from in formula language 222
renewing 621
Certification log
opening 484
Certifiers
registering 480, 483
Char function
formula language 148, 223
CheckAlarms function
formula language 225
Checkbox object
JavaScript 86
CheckCalendar command
formula language 494
CheckFormulaSyntax function
formula language 225
ChooseFolders command
formula language 494

Class statement
in script area of programmer’s pane 51

Classes
Classes tab 39
Classes tab
Java 39

Clear command
formula language 495
Web programming 71

Client information
Web programming 68

ClientType function
formula language 226
Web programming 68

CloseWindow command
formula language 496
Web programming 71

Column formulas
coding 12, 133, 160

Columns
adding carriage returns in 376
appending 509
copying highlighted 514
creating 509, 511
deleting 513, 515
editing in Designer 510, 597
inserting 511
pasting in formula language 527, 528
properties of 510

Commands
@command reference 475
Comments
formula language 396
Compare function
formula language 149, 228

Compile button
Java 39

Compose command
formula language 496
Web programming 71

ComposeWithReference command
formula language 498
Computed field formulas
coding 20, 133
Concatenating
strings in a formula 149
Concatenation operator
formula language 120, 123
Concatenation operators ( & ) and ( + )
formula language 120

Conditional statements
formula language 138
ConfFile function
formula language 230
Connections
testing network 488

Constants
formula language 117
Contains function
formula language 151, 230
Continue LotusScript debugger command 55

Conversions
changing case in formula language 352, 444

Cos function
formula language 154, 231

Count function
formula language 232

CreateAction command
formula language 500

CreateAgent command
formula language 500

CreateControlledAccessSection command
formula language 500

Create function
formula language 156, 233

CreateEllipse command
formula language 500

CreateFolder command
formula language 501

CreateForm command
formula language 501

CreateLinearLayoutRegion command
formula language 501

CreateNavigator command
formula language 502

CreatePolyline command
formula language 502

CreatePolyline command
formula language 502

CreateRectangle command
formula language 502

CreateRectangularHotspot command
formula language 503

CreateSection command
formula language 503

CreateSubform command
formula language 503

CreateTextbox command
formula language 504

CreateView command
formula language 504

Cross-certification
formula language 480, 481

Curly brackets
formula language 117
Currency
converting to text in formula language 426
Current document
formula language 161
Custom controls
adding to forms 522

D

Data types
@functions 126, 437
determining in formula language 334, 335, 336
formula language 114
Database access
getting details of in formula language 261, 452
troubleshooting 539
Database design
creating synopsis 509
refreshing 508
replacing template 509
Database names
getting URL encoded in formula language 467
Web programming 68
Database performance
troubleshooting 481
DatabaseDelete command
formula language 504
DatabaseRepI6Settings command
formula language 505
Databases
access control lists 539
accessing 160, 400, 541
adding to a library 584
changing 539
compacting 539
copying 540
creating 546
creating bookmarks for 478
deleting 504, 504, 541
extracting data from in formula language 237, 240, 245, 246, 251, 256
getting details of in formula language 262, 263, 481, 540
icons 478, 479, 508, 620, 626, 629, 632
indexing 543, 544
opening 546, 548
print settings 550
replicating 505, 541, 546, 613
size limits 481
statistics 481, 485
switching replicas 588
templates 508, 509
testing existence of in formula language 250
titling 546
unread marks 615
Date and time handling (continued)
reading in formula language 235, 313, 366, 368, 379, 405, 431, 435, 436, 471, 472
setting in formula language 206
Date function
formula language 156, 234
Dates
formula language 118
performing calculations on in formula language 206
Day function
formula language 156, 235
DbColumn function
formula language 164
ODBC 166
side-effects 127
DbColumn(Notes data source) function
formula language 237
DbColumn(ODBC data source) function
formula language 240
DbCommand function
formula language 166
Web programming 68
DbCommand function(Domino data source)
formula language 245
DbCommand function(ODBC data source)
formula language 246
DBCS
formula language 376
DbExists function
formula language 250
DbLookup function
Notes database 164
ODBC 166
side-effects 127
DbLookup(Notes data source) function
formula language 251
DbLookup(ODBC data source) function
formula language 256
DbManager function
formula language 160, 261
DbName function
formula language 160, 262
DbTitle function
formula language 160, 263
DDE
formula language 264, 265, 267
DDEExecute function
formula language 264
side-effects 127
DDEInitiate function
formula language 265
side-effects 127
DDEPoke function
formula language 267
side-effects 127
DDETerminate function
formula language 267
side-effects 127
Debug console
Java 39
Debugging
a script 505
agents 57
Debugging (continued)
formula language 130
Java 59
LotusScript 53, 57, 505, 587
LotusScript debugger controls 54
DebugLotusScript command
formula language 505
Decimals
formula language 118
Default reserved word
formula language 161, 268
DEFAULT reserved word
formula language 129
Default validation formulas
coding 133
Default value formulas
coding 19, 133
Deftype statement
in script area of programmer’s pane 51
DeleteDocument function
formula language 269
DeleteField function
formula language 161, 269
Deletions
restoring with formula language 442, 531
Design elements
table of programmable objects 2
Design synopsis
creating 509
DesignDocumentInfo command
formula language 505
DesignFormAttributes command
formula language 506
DesignFormFieldDef command
formula language 506
DesignForms command
formula language 506
DesignFormShareField command
formula language 507
DesignFormUseField command
formula language 507
DesignFormWindowTitle command
formula language 507
DesignHelpAboutDocument command
formula language 507
DesignHelpUsingDocument command
formula language 508
DesignIcon command
formula language 508
DesignMacros command
formula language 508
DesignRefresh command
formula language 508
DesignReplace command
formula language 509
DesignSharedFields command
formula language 509
DesignSynopsis command
formula language 509
DesignViewAppendColumn command
formula language 509
DesignViewAttributes command
formula language 510
Documents (continued) 
checking for fields in with formula language 320, 337 
checking if new in formula language 331 
checking if response in formula language 335 
checking the status of in formula language 208, 322, 323, 324, 325, 326, 327, 331 
closing 496, 538 
closing on the Web 71 
copying 514 
creating 496 
creating on the Web 71 
decrypting 518 
deleting 269, 312, 441, 452, 461, 495, 513, 515, 537 
deleting on the Web 71 
displaying parent view for 559 
editing in Designer 597 
editing in formula language 208, 322, 323, 513, 517, 518, 519, 521, 523, 524, 525, 527, 528, 530, 532, 535, 536, 578 
editing on the Web 71 
encrypting 518 
exporting 542 
filing 494 
footers 520 
forms and 598, 629 
getting details of in formula language 203, 233, 367, 378, 400, 505 
headers 520 
importing files into 544 
inserting objects into 522 
linking 526 
locking/unlocking 279 
marking for deletion 567 
opening 578 
opening on the Web 71 
pasting in formula language 527, 528 
previewing 595, 596 
printing 520, 550, 552 
properties of 273, 275, 276, 277, 278, 281, 282, 505 
read/unread status 611, 612, 615 
refreshing 586, 587, 613, 625 
refreshing hide formulas 586 
replacing design 598, 629 
restoring 531 
retrieve truncated 536 
reversing deletion of in formula language 442 
saving 552, 553 
saving changes in formula language 280 
saving on the Web 71 
searching 518, 519 
searching for 518, 519 
selecting 208, 209, 515, 532, 533 
setting focus in 513, 517, 525, 535, 536 
toggling edit mode 517 
toggling scroll bar in 520 
tracking usage of in formula language 203 
Documents (continued) 
truncation information 282 
Web and 513, 515 
Documents, mailing 
formula language 564 
DocumentUniqueID function 
formula language 161, 284 
Domain function 
formula language 158, 285 
Domains 
getting name of user’s in formula language 285 
Domino Administrator 
opening 482 
Domino Directory 
accessing in formula language 303, 483, 484, 485, 486, 512, 526, 528 
Domino Objects 
JavaScript access 90 
DoWhile function 
formula language 139, 286 
Dynamic Data Exchange. See DDE 264 
E 
ECL 
editing in formula language 287, 288, 395 
ECL security 
@commands 475 
@functions 195 
Edit mode 
for forms and documents 208, 322, 323 
Edit Project button 
Java 39 
EditBottom command 
formula language 513 
EditButton command 
formula language 513 
EditClear command 
formula language 513 
Web programming 71 
EditCopy command 
formula language 514 
EditCut command 
formula language 515 
EditDeselectAll command 
formula language 515 
EditDetach command 
formula language 516 
EditDocument command 
formula language 517 
Web programming 71 
EditDown command 
formula language 517 
EditECL function 
formula language 287 
EditEncryptionKeys command 
formula language 518 
EditFind command 
formula language 518 
EditFindInPreview command 
formula language 519 
EditFindNext command 
formula language 519 
Directory 
checking virtualization of in formula language 338 
going names of in formula language 299, 512 
Directories command 
formula language 512 
DiscoverFolders command 
formula language 512 
Division operator (/) 
formula language 120 
Do function 
formula language 128, 273 
DocChildren function 
formula language 160, 273 
DocDescendants function 
formula language 160, 275 
DocFields function 
formula language 161, 276 
DocLength function 
formula language 161, 277 
DocLevel function 
formula language 160, 278 
DocLock function 
formula language 279 
DocMark function 
formula language 161, 280 
DocNumber function 
formula language 160, 281 
DocParentNumber function 
formula language 160, 282 
DocSiblings function 
formula language 160, 283 
Document object 
JavaScript 86 
Documents 
accessing 161 
assigning values to fields in with formula language 411, 413 
categorizing 610 
check spelling 617 
Index 649
Field values calculating in formula language 424
Fields
  formula language 506, 527, 528
  JavaScript access 86
Fields, types of
  editable 519
  shared 507, 509
Fields, working with
  adding to existing documents in formula language 298
  assigning default value of in formula language 268
  assigning values to in formula language 298, 411, 413
  copying values in with formula language 514
  creating 506
  deleting values in with formula language 269, 441, 513, 515
  editing 506
  editing in Designer 597
  getting value of in formula language 304, 305
Help
  626
  hiding 586
  modifying 507
  referencing in formula language 430, 431
  refreshing 625
  setting focus of 519, 527, 528, 620
  testing existence of in formula language 320
  validation on Web 68
FileCloseWindow command
  formula language 538
  Web programming 71
FileDatabaseACL command
  formula language 539
FileDatabaseCompact command
  formula language 539
FileDatabaseCopy command
  formula language 540
FileDatabaseDelete command
  formula language 540
FileDatabaseInfo command
  formula language 540
FileDatabaseRemove command
  formula language 541
FileDatabaseUseServer command
  formula language 541
FileDir function
  formula language 299
FileExit command
  formula language 541
FileExport command
  formula language 542
FileImport command
  formula language 544
FileShow function
  formula language 299
FileNewDatabase command
  formula language 546
FileNewReplica command
  formula language 546
FileOpenDatabase command
  formula language 546
  Web programming 71
FileOpenDBRepID command
  formula language 548
FilePageSetup command
  formula language 550
FilePrint command
  formula language 550
FilePrintSetup command
  formula language 552
Files
  detaching 491, 516
  Files, working with importing 544
FileSave command
  formula language 552
  Web programming 71
FileSaveNewVersion command
  formula language 553
FileUpload object
  JavaScript 86
  Find and Replace using in documents 518, 519
  FindFreeTimeDialog command
  formula language 553
FloatEq function
  formula language 154, 300
Folder command
  formula language 555
  Web programming 71
FolderCollapse command
  formula language 556
FolderCustomize command
  formula language 556
FolderDocuments command
  formula language 556
FolderExpand command
  formula language 556
  Web programming 71
FolderExpandAll command
  formula language 557
FolderExpandWithChildren command
  formula language 558
FolderMove command
  formula language 558
FolderProperties command
  formula language 558
FolderRename command
  formula language 558
Folders
  copying documents to 555
  creating 501
  discovering in formula language 512
  editing in Designer 510, 558, 597
  moving documents to in formula language 205, 494, 555, 556
  moving documents to on the Web 71
  navigating through documents
    in 568, 569, 570, 571, 572, 573, 574, 575, 576, 577
  removing documents from on the
    Web 71
  removing documents from without
    deleting 205, 587
    renaming 558
    selecting 494
FontList function
  formula language 300
Fonts
  displaying available in formula language 300
Footer
  editing 520
For function
  formula language 139, 301
Form fields
  formula language 114
Form formulas
  coding 11, 133, 511
FormActions command
  formula language 559
Formatting
  code 43
FormLanguage function
  formula language 303
Forms
  editing in Designer 506, 597
  properties of 506
  saving in Designer 552
  testing 559
  Web applications 515
Forms array
  JavaScript 86
Forms, working with
  creating 501
FormTextDocument command
  formula language 559
Formula language
  @functions 126
  coding guidelines 133
  constants 117
  debugging 130
  fields 114
  lexical elements 113
  numeric constants 118
  operator precedence 120
  operators 120
  overview 1
  reserved words 129
  rules 113
  syntax 113
  temporary variables 116
  text constants 117
  time/date constants 118
  variables 114
  Web programming 67
  where to use 1
Formulas
  breaking execution of 401
  changing context 443
  commenting in 396
  conditional statements in 138, 297, 314, 377, 440, 471
  looping 286, 301, 378, 438, 468
  specifying result to return in formula language 273
Frames
  refreshing 585
Frames (continued)
  refreshing on the Web  71
  specifying a target frame in formula language  416
Frames array
  JavaScript  86
Framesets
  editing in Designer  597
  opening  580
  opening on the Web  71
  refreshing  586, 587
Full-text indexes
  creating  543
  deleting  544
  information  544
  updating  544
Function block
  in script area of programmer’s pane  51
Functions
  @function syntax  126

G
GetAddressBooks function
  formula language  303
GetCurrentTimeZone function
  formula language  304
GetDocField function
  formula language  164, 304
GetField function
  formula language  305
GetFocusTable function
  formula language  161, 306
GetHTTPHeader function
  formula language  307
Web programming  68
GetIMContactListGroupNames function
  formula language  308
GetPortsList function
  formula language  309
GetProfileField function
  formula language  309
GetViewInfo function
  formula language  311
Global variables
  defining  51
GoUpLevel command
  formula language  559
Graphics
  adding to layout regions  561
  editing in Designer  584, 597
  resizing in formula language  531
Greater than operator (>)
  formula language  120
Greater than or equal to operator (>= or =>)
  formula language  120
Groups
  creating  480

H
HardDeleteDocument function
  formula language  312
HashPassword function
  formula language  312
Headers
  editing  520
Help
  for fields  626
  opening on the Web  71
  triggering for an application  580
HelpAboutDatabase command
  formula language  560
HelpAboutNotes command
  formula language  560
HelpRequest event
  table of events  21
HelpUsingDatabase command
  formula language  560
Hidden column formulas
  coding  15
Hidden object
  JavaScript  86
Hidden paragraph formulas
  coding  15, 133
Hide action formulas
  coding  7
Hierarchical names
  converting in formula language  369
History object
  JavaScript  86
HotSpotClear command
  formula language  560
HotSpotProperties command
  formula language  561
Hotspots
  coding  9, 133
  creating  503, 524
  editing in Designer  561, 597
  pop-ups  524
  removing  560
Hour function
  formula language  156, 313
HTML
  using JavaScript in  84
HTML attributes formulas
  coding  20
HTTP headers
  formula language  307, 414, 446
Imported Java
  interface elements  39
Importing
  Java into the programmer’s pane  49
Index operator
  formula language  120, 122
Info field
  formula language  114
Info List
  using  40
InheritedDocumentUniqueID function
  formula language  161, 317
Initialize event
  Sequencing  33
  table of events  21
Input enabled formulas
  coding  20
Input translation formulas
  coding  19, 133
Input validation formulas
  coding  19
  formula language  297, 424
Insert subformulas
  formulas  14, 133
Insertion point
  setting in documents  513, 517, 519, 525, 527, 528, 532, 535, 536, 620
InsertSubForm command
  formula language  561
Instant messaging
  formula language  308, 479, 595
Integer function
  formula language  154, 318
Integers
  formula language  118
Internet addresses
  validating in formula language  462
IsAgentEnabled function
  formula language  319
IsAppInstalled function
  formula language  320
IsAvailable function
  formula language  161, 320
IsCategory function
  formula language  160, 321
IsDocBeingEdited function
  formula language  161, 208, 322, 323
IsDocBeingLoaded function
  formula language  161, 324
IsDocBeingMail function
  formula language  161, 324
IsDocBeingRecalculated function
  formula language  161, 325
IsDocBeingSaved function
  formula language  161, 326
IsDocTruncated function
  formula language  326, 327
IsError function
  formula language  144, 328
IsExpandable function
  formula language  160, 329
IsMember function
  formula language  135, 330
IsModalHelp function
  formula language  331
IsNewDoc function
  formula language  161, 331
JUnit function 135, 332
IsNull function 333
IsNumber function 148, 334
IsResponseDoc function 161, 335
IsText function 148, 335
IsTime function 148, 336
IsUnavailable function 161, 337
IsValid function 337
IsVirtualizedDirectory function 338
Italic 603
Iterative statements 139

J
Java 48
compiling and running 39
debug console 39
editing a Java agent 47
importing into the programmer’s pane 49
programmer’s pane 39
Web programming 79
Web services 91
where to use 1
Java programs 49
importing into agents 49
JavaScript 90
accessing the Domino Objects 50
compiling 50
events 83
importing/exporting 50
in the programmer’s pane 49
object model 86
overview 1
page header 50
Web programming 82
JS Header 50
JS Header event 21
JSHeader event 83

K
Key combinations. See Shortcut keys 44
Keyword field formulas 20, 133
Keyfunctions 135, 338

L
Language codes 346
identifying in formula language 346
Language Preference function 340
Languages 303, 340, 346, 456
LaunchApp function 342
Layout elements 562, 597
Layout regions 562
adding text to 562
changing 562
creating 501
editing in Designer 563, 597
graphics in 561
LayoutAddGraphic command 561
LayoutAddText command 562
LayoutElementBringToFront command 562
LayoutElementProperties command 562
LayoutElementSendToFront command 562
LayoutProperties command 563
LDAP directories 369
LDAP listener 342
LDAPServer function 342
Left function 151, 342
LeftBack function 151, 343
Length function 149, 344
Libraries 584
adding databases to 584
Lightweight Directory Access Protocol. See LDAP directories 369
Like function 149, 345
Links 526
creating 526
deleting highlighted 495, 513, 515
designing 525
launching programmatically 527
opening 527
pasting in formula language 527, 528
previewing 595
retrieving parent view from document 559
returning from 572
Links array 86
Links to pages 68
Web programming 68
List concatenation operator 120, 123
List operator 124
List subscript operator 120, 122
Lists 288
counting elements in 114, 135, 232, 288, 407, 419
Ln function 154, 346
Locale function 346
Location object 86
Locations 595, 615
Log function 154, 351
Logical operators 120
Loops 139
LotusScript 561
compiling 52, 63
debugger 53
importing/exporting 52
overview 1
programming environment 37
Web programming 79
Web services 91
where to use 1
LotusScript debugger 54
debugger controls 54
LowerCase function 153, 352

M
Mail 355, 563
creating a memo in formula language 355
creating default form for 563
delivery 488
forwarding 564
opening a user’s mail database 564, 566
receiving 590
sending in formula language 355, 566, 590
user preferences 353, 354, 355, 358, 616, 618
Mail addresses 380, 563
Mail logs 486
opening MAIL.BOX 486
MailAddress command 563
MailComposeMemo command 563
MailDbName function 158, 352
MailEncryptSavedPreference function 353
MailEncryptSentPreference function 354
MailForward command 564
MailForwardAsAttachment command 564
MailOpen command 564
MailRequestCrossCert command
formula language 565
MailRequestNewName command
formula language 565
MailRequestNewPublicKey command
formula language 565
MailSavePreference function
formula language 355
MailScanUnread command
formula language 566
MailSend command
formula language 566
MailSend function
defined 161
formula language 355
side-effects 127
MailSendCertificateRequest command
formula language 566
MailSendEncryptionKey command
formula language 566
MailSendPublicKey command
formula language 567
MailSignPreference function
formula language 358
Matches function
formula language 149, 359
Max function
formula language 154, 361
Member function
formula language 135, 362
Memos
creating a default mail document 563
Message
Web services 91
Messages
writing in formula language 140, 423
Middle function
formula language 151, 363
MiddleBack function
formula language 151, 364
Min function
formula language 154, 365
Minus operator (-)
formula language 123
Minus sign
formula language 118
Minute function
formula language 156, 366
Modems
settings 512
Modified function
formula language 156, 367
Modulo function
formula language 154, 367
Month function
formula language 156, 368
MoveToTrash command
formula language 567
Web programming 71
Multiplication operator (*)
formula language 120
Multiplying
formula language 154
N
Name function
formula language 158, 369
Named elements
coding 15
NameLookup function
formula language 374
Names
going User in formula language 454
Narrow function
formula language 376
NavigateNext command
formula language 568
Web programming 71
NavigateNextHighlight command
formula language 568
NavigateNextMain command
formula language 569
Web programming 71
NavigateNextSelected command
formula language 569
NavigateNextUnread command
formula language 570
NavigatePrev command
formula language 570
Web programming 71
NavigatePrevHighlight command
formula language 571
NavigatePrevMain command
formula language 571
Web programming 71
NavigatePrevSelected command
formula language 571
NavigatePrevUnread command
formula language 572
NavigateToBacklink command
formula language 572
Navigation pane
editing 556, 557, 558
Navigator object
JavaScript 86
Navigator pane
displaying 621
hiding 625
opening views-folders in 625
NavigatorProperties command
formula language 572
Navigators
adding objects to 500, 502, 503, 504
creating 502
designing 500, 502, 504
editing in Designer 572, 597
graphics and 584
hotspots and 584
opening 582
opening on the Web 71
testing 573
NavigatorTest command
formula language 573
NavNext command
formula language 573
Web programming 71
NavNextMain command
formula language 574
Web programming 71
NavNextSelected command
formula language 574
NavNextUnread command
formula language 575
NavPrev command
formula language 575
Web programming 71
NavPrevMain command
formula language 576
Web programming 71
NavPrevSelected command
formula language 576
NavPrevUnread command
formula language 577
Negation operator (-)
formula language 120
Negative numbers
formula language 118
New Class button
Java 39
NewLine function
formula language 153, 376
No function
formula language 377
NoExternalApps
environment variable 127
Not equal to operator
formula language 120
Not operator
formula language 120
NoteID function
formula language 161, 378
Notes client
exiting 538, 541
getting details of in formula language 560
startup settings 616
notes.ini file
formula language 129, 143
retrieving path to 230
Nothing function
formula language 135, 378
Now function
formula language 156, 379
Null fields
formula language 114, 333
Number handling
formula language 295, 300, 318, 346, 351, 361, 365, 367, 389, 404, 418, 422, 424
Number lists
finding largest number in with
formula language 361
finding smallest number in with
formula language 365
Numbers
comparing in formula language 300
converting to time-date in formula language 234
generating random in formula language 394
rounding 404
verifying data type in formula language 334
Numeric constants
formula language 118
Object model
JavaScript 86
ObjectDisplayAs command
formula language 577
ObjectExecute event
table of events 21
ObjectOpen command
formula language 577
ObjectProperties command
formula language 578
Objects
creating 522
deleting highlighted 495, 513, 515
Objects tab
defined 37
defining subprograms and globals 51
Java 39
using 40
OLE
displaying 577
editing 577, 626
editing in Designer 578, 597
OLE objects
formula language 522
onBlur event
table of events 21
Web programming 83
onChange event
table of events 21
onClick event
table of events 21
Web programming 83
onDBClick event
table of events 21
onFocus event
table of events 21
Web programming 83
onHelp event
table of events 21
onKeyDown event
table of events 21
onKeyPress event
table of events 21
onKeyUp event
table of events 21
onLoad event
Web programming 83
onLoad/PostOpen event
table of events 21
onMouseDown event
table of events 21
onMouseMove event
table of events 21
onMouseOut event
table of events 21
onMouseOver event
table of events 21
onReset event
table of events 21
onSelect event
table of events 21
onSubmit event
Web programming 83
onSubmit/QuerySave event
table of events 21
onUnload event
Web programming 83
onUnload/QueryClose event
table of events 21
OpenCalendar command
formula language 578
OpenDocument command
formula language 578
Web programming 71
OpenFrameset command
formula language 580
Web programming 71
OpenHelpDocument command
formula language 580
Web programming 71
OpenNavigator command
formula language 582
Web programming 71
OpenPage command
formula language 582
Web programming 71
OpenView command
formula language 583
Web programming 71
Operation
Web services 91
Operators
comparison 123
formula language 114
OptimizeMailAddress function
formula language 158, 380
Option statement
in script area of programmer’s pane 51
OR operator
formula language 120
Order of evaluation
@functions and @commands 128
formula language 121
Organizational units
formula language 483, 624
Organizations
registering 483
retrieving directory names of 381
OrgDir function
formula language 381
Output
output panel in LotusScript
debugger 56

P
Page breaks
adding in formula language 523
in views 628
Pages
editing in Designer 506, 597
opening 582
opening on the Web 71
refreshing 586, 587
Pair-wise list operators
formula language 124
Paragraphs
aligning 604
indenting in formula language 521

Paragraphs (continued)
setting line length 530
styling 605
Parent documents
formula language 282, 284, 304, 317, 411, 586, 596
Parentheses
formula language 121
Password function
formula language 158, 381
Password object
JavaScript 86
PasswordQuality function
formula language 382
Passwords
formula language 312, 381, 382, 464, 482
PasteBitmapAsBackground command
formula language 584
PasteBitmapAsObject command
formula language 584
Pathnames
getting directory name in formula language 299
Permutated operators
formula language 120
Personal Address Book
formula language 526, 528
Pi function
formula language 154, 382
PickList function
formula language 383
getting user input with 141
side-effects 127
PictureProperties command
formula language 584
Platform function
formula language 158, 386
Plus operator (+)
formula language 123
Policies
formula language 388
PolicyListFieldLocked function
formula language 388
Pop-ups
creating 524
Port
Web services 91
Port type
Web services 91
Ports
displaying available in formula language 309
enabling/disabling 616
Positive operator
formula language 120
PostDocumentDelete event
table of events 21
PostDragDrop event
table of events 21
PostedCommand function
defined 147
formula language 388
side-effects 127
PostEntryResize event
table of events 21
PublishDatabase command  
formula language 584

PostModeChange event  
table of events 21
PostOpen event  
table of events 21
PostPaste event  
table of events 21
PostRecalc event  
table of events 21
PostSave event  
table of events 21
PostSend event  
table of events 21
Power function  
formula language 154, 389
Precedence of operators  
formula language 120
Preferences file  
formula language 129
Previews  
documents 632
Primary names  
extracting in formula language 454
Print command 550
Print setup  
formula language 550, 552
Profile documents  
creating 528, 529
editing 528, 529
retrieving fields from in formula language 309
setting field values in with formula language 415
Programmable design elements  
table 2
Programmer’s pane  
accessing 37
defined 37
JavaScript 49
properties 43
properties and events 41
Programming  
Auto complete 45
LotusScript programming environment 37
overview 1
Web 67
Prompt function  
formula language 390
getting user input with 141
side-effects 127
writing messages with 140
ProperCase function  
formula language 153, 393
Properties  
user interface 41
Property block  
in script area of programmer’s pane 51
Public keys  
extracting data from in formula language 222
PublishDatabase command  
formula language 584

Q
QueryAddToFolder event  
table of events 21
QueryClose event  
table of events 21
QueryDocumentDelete event  
table of events 21
QueryDocumentUndelete event  
table of events 21
QueryDragDrop event  
table of events 21
QueryEntryResize event  
table of events 21
QueryModeChange event  
table of events 21
QueryOpen event  
table of events 21
QueryOpenDocument event  
table of events 21
QueryPaste event  
table of events 21
QueryRecalc event  
table of events 21
QuerySend event  
table of events 21
Quotations marks  
formula language 117

R
Radio object  
JavaScript 86
Random function  
formula language 154, 394
Redundant spaces  
removing in formula language 440
Reference list  
defined 37
Reference tab  
defined 37
in programmer’s pane 42
RefreshECL function  
formula language 395
RefreshFrame command  
formula language 585
Web programming 71
RefreshHideFormulas command  
formula language 586
RefreshParentNote command  
formula language 586
RefreshWindow command  
formula language 586
RegionDoubleClick event  
table of events 21
RegKeys keys  
querying in formula language 395
RegQueryValue function  
formula language 395
ReloadWindow command  
formula language 587
Rem directive  
%Rem in script area of programmer’s pane 52
REM reserved word  
formula language 129, 396
Remote debugger  
Java 59
Remote Debugger  
LotusScript 57
RemoteDebugLotusScript @command  
formula language 587
RemoveFromFolder command  
formula language 587
Web programming 71
RenameDatabase command  
formula language 588
Renaming  
in script area of programmer’s pane 45
Repeat function  
formula language 153, 397
Replace function  
formula language 135, 398
ReplaceSubstring function  
formula language 399
ReplicaID function  
formula language 160, 400
Replicas  
choosing database 541
Replication  
initializing 589, 590, 591
settings 505
specifying server 590
stopping 589, 591
Replication formulas  
coding 5, 133
Replicator command  
formula language 589
ReplicatorReplicateHigh command  
formula language 589
ReplicatorReplicateNext command  
formula language 589
ReplicatorReplicateSelected command  
formula language 590
ReplicatorReplicateWithServer command  
formula language 590
ReplicatorSendMail command  
formula language 590
ReplicatorSendReceiveMail command  
formula language 590
ReplicatorStart command  
formula language 591
ReplicatorStop command  
formula language 591
Response documents  
creating 498
previewing parent 596
properties 273
Responses function  
formula language 160, 161, 400
Return function  
formula language 128, 146, 401
Return values  
@functions 126
Rich text fields  
formula language 114, 528, 332, 533
Rich text paragraph styles  
formula language 521
Right function  
formula language 151, 402
RightBack function  
formula language 151, 403
Roles
getting User’s in formula language 458
Round function
formula language 154, 404
Row custom color formulas
coding 15
Rows
checking if expandable in formula language 329
Ruler
displaying in documents 628
Run box
defined 37
Run-time errors
formula language 144, 315, 328
RunAgent command
formula language 591
Web programming 71
RunScheduledAgents command
formula language 592

S
SaveOptions
Web programming 79
SBCS
formula language 469
Schedules
reviewing 553
Scientific notation
formula language 118
Script area
programmer’s pane 37, 42
Script Debugger
running 505
Script libraries
creating 61
Java 47, 62
LotusScript 62
Scroll bars
for documents 520
for views 624, 628
Search bar
displaying 71
Second function
formula language 156, 405
Section access formulas
coding 14, 133, 500
Section title formulas
coding 14, 133
SectionCollapse command
formula language 592
SectionCollapseAll command
formula language 593
SectionDefineEditors command
formula language 593
SectionExpand command
formula language 593
SectionExpandAll command
formula language 594
SectionProperties command
formula language 594
SectionRemoveHeader command
formula language 594
Sections
access to 500
Sections (continued)
collapsing 592, 593
creating 500, 503
editing in Designer 594, 597
editors and 593
expanding 593, 594
removing content 594
Security
ECL in @commands 475
ECL in @functions 195
Select function
formula language 407
Select object
JavaScript 86
Select reserved word
formula language 133, 405
SELECT reserved word
formula language 129
Selection formulas
coding 12, 511
formula language 133, 208, 515, 532,
533
SendInstantMessage command
formula language 595
Server commands
entering remotely 487
ServerAccess function
formula language 407
ServerName function
formula language 160, 409
Servers
access levels for 407
accessing log 485
certifying IDs 480
debugging LotusScript on 57
dialing into 610, 611
logging users off 617
monitoring 488
registering 487
testing network connections 488
Service
Web services 91
Set function
formula language 410
SetCurrentLocation command
formula language 595
SetDocField function
formula language 164, 411
SetEnvironment function
formula language 146, 412
SetField function
formula language 161, 413
SetHTTPHeader function
formula language 414
Web programming 68
SetProfileField function
formula language 415
SetTargetFrame function
formula language 416
SetViewInfo function
formula language 417
Shared fields
converting 507
creating 509
inserting 507
Shortcut keys
formula language 130
Shortcut keys (continued)
Reference tab in programmer’s pane 42
script editor 43, 44
Show action formulas
coding 133
ShowHideIMContactList command
formula language 595
ShowHideLinkPreview command
formula language 595
ShowHideParentPreview command
formula language 596
ShowHidePreviewPane command
formula language 596
ShowProperties command
formula language 597
Side-effects
@functions 127
Sign function
formula language 154, 418
Sin function
formula language 154, 419
SmartIcons
accessing 597
coding 3, 616
SmartIconsFloating command
formula language 597
SmartIconsNextSet command
formula language 597
Sort function
formula language 135, 419
Sound function
formula language 422
Spaces
formula language 114, 533
Sqr function
formula language 154, 422
Square roots
formula language 422
Status bars
writing messages to in formula language 423
StatusBar function
formula language 158, 423
Step Exit
LotusScript debugger 55
Stepping through applications
LotusScript debugger 55
String conversions
formula language 211, 223, 376, 393,
426, 428, 429, 437, 438, 469
String handling
changing case in formula language 352, 393, 444
conversion in formula language 422
extracting in formula language 342,
343, 363, 364, 402, 403, 470
finding length in formula language 344
finding position in string with formula language 362
finding substrings in with formula language 218, 230, 289, 330, 332
itemizing in formula language 295
pattern matching in formula language 345, 359
repeating in formula language 397
String handling (continued)
  replacing in formula language  399
  trimming in formula language  440
Strings
  encoding in formula language  312, 381, 464
  formula language  148
StyleCycleKey command
  formula language  597
Styles
  cycling through assigned  597
Sub block
  in script area of programmer’s pane  51
Subforms
  creating  503
  editing in Designer  506
  inserting  14, 561
Subprograms
  defining  51
  selecting in LotusScript debugger  54
Superscript operator
  formula language  120, 122
Subset function
  formula language  135, 423
Substrings
  formula language  151
Subtraction
  formula language  154
Subtraction operator (-)
  formula language  120
Success function
  formula language  146, 424
  Web programming  68
Sum function
  formula language  154, 424
SwitchForm command
  formula language  598
SwitchView command
  formula language  598
  Web programming  71
Syntax errors
  formula language  144

T
Tables
  columns  534, 535
  creating  524
  deleting highlighted  495, 513, 515
  editing in formula language  534
  name and column information in formula language  306
  rows  534, 535
Tabs
  inserting with formula language  223
  showing/hiding in formula language  533
Tan function
  formula language  154, 425
TemplateVersion function
  formula language  426
Temporary variables
  formula language  116, 410
Terminate event
  table of events  21

Text
  abstracting from with formula language  197
  adding to layout regions  562
  aligning  599, 600, 604
  coloring  605, 606
  copying highlighted  514
  deleting highlighted  495, 513, 515
  editing in script area of programmer’s pane  44
  font  602, 607
  formatting for Internet  530
  inserting programmatically  524
  pasting in formula language  527, 528
  properties in programmer’s pane  43
  searching for  518, 519
  sizing  602, 606, 608
  spacing  601, 608, 609
  styling  601, 603, 609
  text constants in formula language  117
  verifying data type in formula language  335
Text function
  formula language  148, 426
Text lists
  comparing in formula language  228
  concatenating in formula language  294, 316
  extracting from  470
  finding and replacing in with formula language  398, 399
  finding keywords in with formula language  330, 332, 338
  removing duplicates in with formula language  442
  styling  601, 604
Text object
  JavaScript  86
Text operator
  formula language  123
TextAlignCenter command
  formula language  599
TextAlignFull command
  formula language  599
TextAlignLeft command
  formula language  599
TextAlignNone command
  formula language  600
TextAlignRight command
  formula language  600
TextArea object
  JavaScript  86
TextBold command
  formula language  601
TextBullet command
  formula language  601
TextCycleSpacing command
  formula language  601
TextEnlargeFont command
  formula language  602
TextColor command
  formula language  602
TextItalic command
  formula language  603
TextNormal command
  formula language  603
TextNumbers command
  formula language  604
TextOutdent command
  formula language  604
TextParagraph command
  formula language  604
TextParagraphStyles command
  formula language  605
TextPermanentPen command
  formula language  605
TextReduceFontSize command
  formula language  606
TextSetFontColor command
  formula language  606
TextSetFontFace command
  formula language  607
TextSetFontSize command
  formula language  608
TextSpacingDouble command
  formula language  608
TextSpacingOneAndAHalf command
  formula language  609
TextSpacingSingle command
  formula language  609
TextToNumber function
  formula language  428
TextToTime function
  formula language  429
TextUnderline command
  formula language  609
ThisName function
  formula language  430
ThisValue function
  formula language  431
Time
  formula language  118
  formula language  156, 431
  time zones
    specifying  304, 434
  time-dates
    verifying data type in formula language  336
TimeMerge function
  formula language  433
TimeToTextInZone function
  formula language  433
TimeZoneToText function
  formula language  434
Title bar
  defined  37
Title field
  formula language  114
Today function
  formula language  156, 435
Tomorrow function
  formula language  156, 436
ToNumber function
  formula language  437
Toolbars
  using and customizing  3, 616
ToolsCall command
  formula language  610
ToolsCategorize command
  formula language  610
ViewCollapse command (continued)
  Web programming 71
ViewCollapseAll command
  formula language 623
  Web programming 71
ViewExpand command
  formula language 623
  Web programming 71
ViewExpandAll command
  formula language 623
  Web programming 71
ViewExpandWithChildren command
  formula language 624
ViewHorizScrollBar command
  formula language 624
ViewMoveName command
  formula language 624
ViewNavigatorsFolders command
  formula language 625
ViewNavigatorsNone command
  formula language 625
ViewRefreshFields command
  formula language 625
  Web programming 71
ViewRefreshUnread command
  formula language 626
ViewRenamePerson command
  formula language 626
Views
  collapsing 622, 623
  columns in 509, 511
  creating 504
  designing 511
  documents and 527
  editing in Designer 510, 597
  expanding 623, 624
  exporting on the Web 71
  filtering in formula language 417, 627, 628
  getting details of in formula language 311, 466
  importing files into 544
  naming in a formula language 130
  navigating through documents
    in 568, 569, 570, 571, 572, 573, 574, 575, 576, 577
  opening 583
  opening on the Web 71
  previewing documents 596
  printing 590, 552
  properties of 510
  refreshing 586, 587, 625
  refreshing on the Web 71
  saving in Designer 552
  scroll bars 624
  searching 518, 519
  selecting documents in 532
  selecting documents to include in 405, 511
  setting design for documents in 11, 511
  switching 598, 622

ViewShowFieldHelp command
  formula language 626
ViewShowObject command
  formula language 626
ViewShowOnlyCategories command
  formula language 627
ViewShowOnlySearchResults command
  formula language 627
ViewShowOnlySelected command
  formula language 627
ViewShowOnlyUnread command
  formula language 628
ViewShowPageBreaks command
  formula language 628
ViewShowRuler command
  formula language 628
ViewShowSearchBar command
  formula language 628
  Web programming 71
ViewShowServerNames command
  formula language 629
ViewShowThisUnread function
  formula language 465
ViewShowUnread command
  formula language 629
ViewSwitchForm command
  formula language 629
ViewTitle function
  formula language 146, 160, 466
  Virtualized directories
  formula language 338

W
Web agents
  creating 79
Web programming environment 67
Web services
  design element 92
  invoking 95
  LotusScript and Java mappings 95
  overview 91
  terminology 91
WebDbName function
  Web programming 68
WebDBName function
  formula language 467
WebQueryOpen events
  table of events 21
  Web programming 79
WebQuerySave events
  table of events 21
  Web programming 79
Weekday function
  formula language 156, 467
While function
  formula language 139, 468
White space
  in formula language 533
Wide function
  formula language 469
Window
  refreshing 586, 587

Window (continued)
  resizing 630, 631
  switching 631
Window object
  JavaScript 86
Window title formulas
  coding 13, 133, 160, 507
WindowCascade command
  formula language 630
WindowMaximize command
  formula language 630
WindowMaximizeAll command
  formula language 630
WindowMinimize command
  formula language 630
WindowMinimizeAll command
  formula language 631
WindowNext command
  formula language 631
WindowRestore command
  formula language 631
Windows registry
  querying in formula language 395
WindowTile command
  formula language 631
WindowTitle field
  formula language 114
WindowWorkspace command
  formula language 632
Word function
  formula language 151, 470
Workspace
  displaying 632
  editing in Designer 597, 632
  stacking icons 632
WorkspaceProperties command
  formula language 632
WorkspaceStackReplicaIcons command
  formula language 632
WSDL
  Web Services Description Language 91

Y
Year function
  formula language 156, 471
Yes function
  formula language 471
Yesterday function
  formula language 156, 472

Z
Zone function
  formula language 156, 472
ZoomPreview command
  formula language 632
Notices

This information was developed for products and services offered in the USA. IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

Lotus Software
IBM Software Group
One Rogers Street
Cambridge, MA 02142
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or by any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.
Trademarks

IBM, the IBM logo, AIX, DB2, Domino, Freelance, Freelance Graphics, iSeries, i5/OS, Lotus, Lotus Notes, LotusScript, Notes, 1-2-3, OS/400, Quickplace, S/390, Sametime, SmartSuite, WebSphere, z/OS, and zSeries are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.

Additional IBM copyright information can be found at: http://www.ibm.com/legal/copytrade.shtml

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property of CompuServe Incorporated.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.