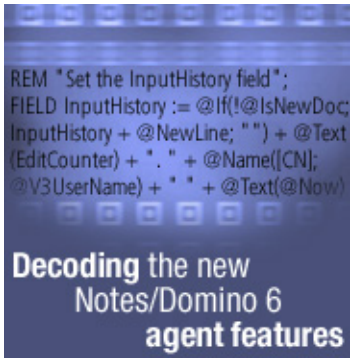
Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today



Level: Intermediate
Works with: Notes/Domino 6
Updated: 01-Oct-2002

by
Julie
Kadashevich

In Notes 6, our goal for agents is to address the most frequently customer-requested features. These include (among others) allowing server-based agents to access databases on remote servers, and providing the ability to specify the person on whose behalf the agent is running. The common thread that runs through these requests is *security*—enhancements to the agent security model.

We've also added programmability features to LotusScript, Java, and the formula language, including a large number of new back-end methods. And we've made numerous other improvements to agent functionality on both the client and server.

This article presents an overview of all new agent-related features in Notes/Domino 6. I'll begin by talking about security and then look at programmability improvements and other agent functionality. This article assumes you're experienced with agents and are familiar with agent terminology.

Security features

In Notes/Domino 6, we've dedicated a great deal of effort to agent security issues. These include:

- Accessing remote servers
- Modifying and saving agents on the server
- Enabling scheduled agents using a Web client
- Allowing editor-level users to run LotusScript and Java agents
- Providing the ability to specify the person on whose behalf the agent is running

Plus, we've added a number of other security features, which I'll describe later in this section.

Accessing remote servers

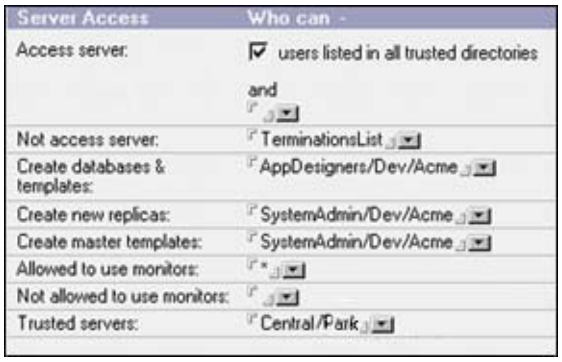
Prior to Notes/Domino 6, if a server-based scheduled or Web-based agent referenced a database on another server, the agent would fail. This happened because we did not have a security protocol that would allow us to authenticate the effective user of the agent (the person on whose behalf the agent is running) with server A while the agent was running on server B. For example, the following script would fail with an error on the bold line if a user tried to run it on the server. (In this example, the agent runs on server Central/Park and creates a replica of a database on server Gorky/Park.)

```
Sub Initialize
        Dim session As New NotesSession
        Dim db1 As NotesDatabase
        Dim db2 As NotesDatabase
        Dim dbname As String
```



```
        dbname = "newdb.nsf"
        Set db1 = session.CurrentDatabase
        Set db2 = db1.CreateReplica("Gorky/Park", dbname)
End Sub
```

In Notes/Domino 6, the preceding code runs successfully if server Gorky/Park (where the database is created) is configured to trust server Central/Park (where the agent is running). Note that both servers need to be running Domino 6. To ensure server Gorky/Park trusts server Central/Park, open Gorky's Server document in the Domino Directory. In the Security tab, go to the Server Access section and enter Central/Park in the Trusted servers field:

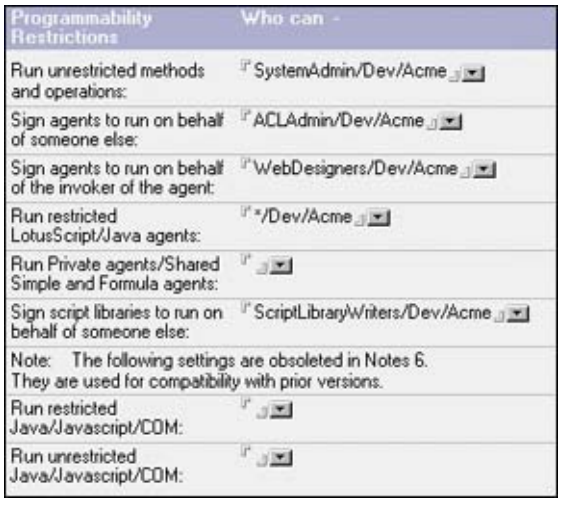


Saving agents on the server

In previous releases, agents could not manipulate and save other agents running on the server. For example, it was not possible to write an agent that could enable or disable another server-based agent. This was because we did not have a way to preserve the user identity associated with the agent being changed. In the following example, an error would be generated on the method highlighted in bold:

```
Sub Initialize
        Dim s As New notessession
        Dim db As notesdatabase
        Set db = s.currentdatabase
        Dim ag As NotesAgent
        Set ag = db.getAgent("Send reminder")
        ag.IsEnabled=False
        Call ag.save()
End Sub
```

This code will now run successfully in Notes/Domino 6. To maintain security, an agent can enable another agent if they have the same effective user (the identity under which the agent is running), or if the agents' signers are privileged and are listed in the “Sign agents to run on behalf of someone else” field. This is a new field that has been added to the Programmability Restrictions section of the Server document's Security tab:



This means users don't need any special rights to manipulate their own agents, but they need these rights to manipulate agents written by others. Note also that the agent cannot modify itself.

Enabling scheduled agents using a Web client
Agents running in the Web browser are run on the server. Thus prior to Notes/Domino 6, if browser agents attempted to programmatically modify other agents, they were under the same restrictions as server-based scheduled agents (as described in the previous section).

To enable scheduled agents from a Web client in Notes/Domino 6, you can create what I call a "helper agent." A helper agent is signed by a user who has the rights to sign agents that run on behalf of someone else. The purpose of the helper agent is to enable and save the agent to be scheduled (which I'll call the "worker" agent). Both the helper agent and the worker agent should run with the checkbox Run as web user selected in their Agent properties, to capture the Web user's identity. Also, the databases in which the agents reside must have the Maximum Internet name and password level set to at least Designer, because we are changing part of the agent design. (You can set this in the database by choosing File - Database - Access Control - Advanced, or through the Domino Administrator.)

The following is an example helper agent:

```
Set db = s.currentdatabase
        Set a = db.getAgent("Worker")
        a.isEnabled = True
        a.Servername = “Gorky/Park”
        Call a.Save
```

When the helper agent finishes running, the worker agent runs as a scheduled agent under the rights of the Web user.

Allowing editor-level users to run LotusScript and Java agents
To enable and run LotusScript and Java agents prior to Notes/Domino 6, you needed at least Designer access and the right to run LotusScript and Java agents. This was because enabling the agent consists of two parts: first, signing the agent, and second, making it eligible to run on a schedule. Because signing the agent requires changing its design, you needed Designer access to do this. So for example, users needed at least Designer access to their mail files (and the right to run LotusScript agents) to run the Out of Office agent, which is written in LotusScript.

Many of our customers requested the ability to give users Editor access to their mail databases, and yet still be able to run the Out of Office agent. To accommodate this, we separated the two parts of enabling the agent by allowing the agent to become eligible to run without resigning it. We also needed a way to run the agent on behalf of the editor-level user who is not privileged to change the agent (or possibly does not have rights to run LotusScript or Java agents). We now provide this functionality, through the Allow user activation checkbox in the Agents properties box:

When this option is selected, editors can activate (enable) the agent. When it is unselected (which is the default), the agent behaves the same as in previous releases. When users with Editor access activate an agent, all they can change is the status of the agent. Editor-level users do not need rights to run LotusScript or Java agents. (If they are not given the rights to run agents, these users will not be able to run any agents other than ones set up to run by an editor.)

You can configure agents for editor-level users manually or programmatically:

- Manually, you must have the right to sign agents to run on behalf of others. Then open Domino Designer, open the Agent properties box, and enter the names of the users into the “Run on Behalf of” field (shown in the preceding screen). Select the Allow user activation checkbox. The agent will be signed by you, and when an editor-level user you specified enables the agent, it will run as scheduled under the authority of that user.
- Programmatically, you can use the AdminP request “Set User name and Enable Agent.” This can be initiated by the ConfigureMailAgent method on the new AdminP class, as the following example illustrates:

```
Dim AdminRequest As Variant
Set AdminRequest = session.CreateAdministrationProcess(“GorkyPark/Moscow”)
UserName = "Ninochka"
AgentActivatable = True
Agent = “DemoAgent”
If Not(agent.IsEnabled) Then
        Call AdminRequest.ConfigureMailAgent(UserName, Agent, AgentActivatable, False)

Elseif
        Call AdminRequest.ConfigureMailAgent(UserName, Agent, AgentActivatable, True)
```

The primary reason we added this feature is to allow mail users who have Editor access to enable LotusScript agents. You can find several examples of agents using this feature in the Notes 6 Mail template.

Once you've set up the agent to allow user activation, any editor-level user named in the Run on behalf o field can enable/disable the agent. Please note that Domino 5 servers will not recognise that the agent was enabled by an editor using Notes/Domino 6. As a result, an agent can get into a state where Release 5 will treat it as enabled, and Notes/Domino 6 will treat it as disabled (and vice versa).

To keep track of the enabled status, we've added icons to the Agents view in Notes/Domino 6. A check with a number 5 next to it indicates the agent is enabled only for Release 5 servers (5.0.7 and earlier). A check with a number 6 represents agents that run on Notes/Domino 6 (also 5.0.8 and later). The plain check identifies agents enabled for both Release 5 and Notes/Domino 6. To summarize:

- A check with a 5 means the agent was disabled by an editor-level user in Notes/Domino 6. The agent will be treated as disabled by Domino 6 servers (also 5.0.8 and later), but as enabled on Domino 5 servers running 5.0.7 and earlier.
- A check with a 6 means the agent was enabled by an editor-level user in Notes/Domino 6. The agent will be

treated as enabled by Domino 6 servers (also 5.0.8 and later), but as disabled on Domino 5 servers running 5.0.7 and earlier.
- A check without a number means the agent was not enabled/disabled by an editor-level user and will behave the same in Release 5 and Notes/Domino 6.

Note that starting with Domino 5.0.8, the agent manager can recognize agents turned on by editor level users. These agents are recognized by the agent manager and executed on Domino 5.0.8 and later servers. The 5.0.8 user interface does not accommodate this new feature, however. Therefore Release 5 editor-level users cannot activate agents, nor can Release 5.0.x Designer graphically differentiate the status of agents with "check 5" or "check 6" icons, indicating whether they were enabled by Release 5 or Notes/Domino 6 users.

Other security features
In addition to the preceding functionality, we've added a number of other security-related features to agents.

New restriction lists
To support the new functionality described earlier (such as running agents on behalf of designated users) and to provide better control and granularity, we've modified the Agent Restrictions section of the Security tab in the Server document. For example, this section (shown in the second illustration in this article) has been renamed Programmability Restrictions. We've eliminated the separate sections for agents and Java/JavaScript/COM and combined the unrestricted rights fields into this single section. (The restricted list for Java/JavaScript/COM was equivalent to giving these users access to the server, so instead, we use the names specified in the Security tab's Who can Access server field to determine the restricted level of access for Java/JavaScript/COM users.)

Prior to Notes/Domino 6, Agent Restrictions consisted of three restriction lists: unrestricted, restricted, and personal. The new Programmability Restrictions consist of six lists:
- Run unrestricted methods and operations lists people who can perform all agent operations, including the ones that can compromise security. Only a small number of the most trusted users should have these rights on a production server.
- Sign agents to run on behalf of someone else gives the right to create agents that run as a scheduled agent on behalf of users other than the signer. This is a very powerful capability, as people with this right can gain access to the data of another person, as well as impersonate someone (as the sender of mail or creator of documents).
- Sign agents to run on behalf of the invoker of the agent allows members to create agents that can be invoked by someone else (for example, Web agents).
- Run restricted LotusScript/Java agents gives users the right to run LotusScript and Java agents that perform a large subset of operations that cannot violate security. The majority of users should be included in this list.
- Run Private agents/Shared Simple and Formula agents lets users run personal or shared formula and simple agents.
- Sign script libraries to run on behalf of someone else gives users the right to sign script libraries.

Note that two fields relating to Java/JavaScript/COM are included, with a note they are obsolete in Notes/Domino 6. These fields are included to maintain backward compatibility.

Updated agent security rules
The good news is that there are not a lot of new rules to learn for agent security! There's only one new option, and it applies to agents running on the server: Instead of always running with the effective user equal to the agent signer, now the agents can run with the authority of another person (but only if the agent signer has the rights to create an agent that runs on behalf of someone else).

Agent security still consists of two parts:
- Programmability restrictions (formerly called agent restrictions), which control who can run the agent with what level of rights. These are set in the Server document in the Domino Directory.
- Database ACL, which controls the level of access to the data the agent's effective user has.

Whether or not programmability restrictions apply depends on how the agent is invoked. If the agent is invoked on the client, these restrictions do not apply. If the agent is invoked on the server, restrictions do apply. Programmability restrictions are always determined based on the agent signer. The following table summarizes agent security rules:

Where the agent runs	Client		Server	
How the agent is	User initiated	Scheduled	HTTP run as	HTTP run

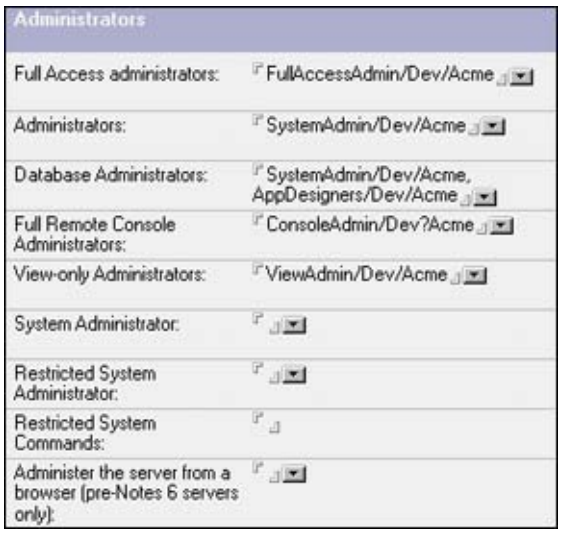
invoked			Web user	as signer
Restrictions	N/A	N/A	Signer	Signer

Which identity is used as the agent's effective user depends on how the agent is invoked. If invoked on the client, the identity of the person logged onto the workstation is used as the effective user of the agent. If the agent is invoked from the Web and is set to run as "Web user," the effective user is the Web user identity. For scheduled agents on the server (and agents invoked from the Web running in the "run as agent signer" mode), the effective user is either the person named in the On behalf of field (if specified), of the agent signer (if On behalf of is not specified).

Where the agent runs	Client		Server		
How the agent is invoked	User initiated	Scheduled	HTTP run as Web user	HTTP run as signer	Scheduled
ACL checks	Invoker	Invoker	Invoker	"On behalf of" or signer	"On behalf of" or signer

Full access administrator
Domino 6 supports several levels of administrators. A new "super" level of access is called full access administrator. If you are designated as a full access administrator in the Security tab of the Server document, you have access to all databases on the server, even if you are not explicitly listed in the ACL. You can, therefore, perform all administrative functions. In addition, you have unrestricted agent rights, even though you may not be explicitly listed in the who can sign unrestricted agents field.

To use this feature, you must be listed in the Full Access administrators field in the Server document:



Once you're included in the Server document, you can activate Full Access administrator either from the Domino Administrator client, or programmatically.

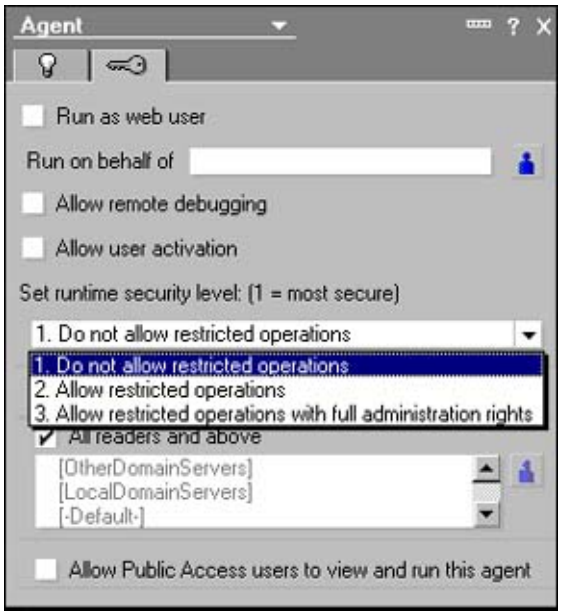
- In the Domino Administrator client, choose Administration - Full Access Administration.
- On the server, this functionality is available programmatically via agents. The agent signer has to be listed either in the Full Access administrator or Run unrestricted methods and operations fields. Also, the Set runtime security level option in the Agent properties box has to be set to Allow restricted operations with full administration rights. (More on this in the next section.)

Note that the agent settings have no effect when agents are running on the client; in this case, full access administrator has to be set from the Administrator client.

Selecting rights on a per agent basis

The right to sign unrestricted agents should be given out to a few of your most trusted users. Administrative IDs are often used for such purposes. But not all agents created by an administrator need unrestricted rights. Prior to Notes/Domino 6, administrators were forced to create multiple IDs so that they could sign agents that required unrestricted rights differently than agents that did not. Aside from licensing issues, this was a major inconvenience. To address this issue, we added an option called Set runtime security level in the Agent properties box to specify the maximum level of operation for a specific agent signed by unrestricted users.

In Notes/Domino 6, unrestricted signers can select one of three runtime security levels that agents signed by their ID can operate in. These levels are assigned on a per agent basis.



- Do not allow restricted operations limits the agent to running in restricted mode. The agent will not be allowed to perform file input/output or signing but will be allowed to do regular LotusScript or Java operations.
- Allow restricted operation makes the agent run in unrestricted mode. File input/output and signing are allowed.
- Allow restricted operations with full administration rights allows the agent to perform all operations, as well as open databases without having to be listed in the ACL explicitly. The agent can also perform functions that require administrative rights.

This functionality is available only to signers who already have unrestricted access. If a user with restricted access sets this value to anything other than “Do not allow restricted operations,” the agent will not run and generates a runtime error.

Note that for backwards compatibility, an existing agent that has not been edited with Designer 6 will run on a Domino 6 server with the rights of the signer. So for unrestricted signers, the agent will run with unrestricted rights. When the agent is subsequently edited in Designer 6, the default setting is “Do not allow restricted operations” (which is a safer setting). If you want to perform unrestricted operations in the agent after you have edited it with Designer 6, you need to explicitly set this security setting. (I wanted to point this out since I found this to be a common gotcha!)

Programmability enhancements

Up until now this article addressed new security features. Notes/Domino 6 also contains a large number of enhancements in programmability. These fall into several categories:

- Core languages such as LotusScript, Java, formula, or simple actions
- Methods and classes that enhance the core languages with capabilities specific to Notes and Domino
- Tools such as the debugger

Here are the highlights for these areas.

LotusScript
The LotusScript language in Notes/Domino 6 has two new data types, Byte and Boolean. Byte type allows a get statement to retrieve 1 byte from a file. Its range is from 0 to 256, and it is unsigned. Boolean type has a range of True or False (-1, 0). It is 1 byte in size.

A new LSX called LS2J provides interconnectivity between LotusScript and Java. LS2J allows LotusScript programs to instantiate Java objects, call Java objects methods, and read/write Java object properties.

The LotusScript USE statement now loads Java libraries. This requires the LS2J LSX.

Handling of UI methods in background agents
The handling of UI methods in background agents has been such a common source of errors, I would like to spend some extra time discussing this. Prior to Notes/Domino 6, when any UI classes were used in a background agent (even if just a Dim statement for NotesUIWorkspace), the agent would not run in the background. As you may know, back-end classes perform operations on Notes databases, while front-end classes manipulate the user interface (NotesUIWorkspace, NotesUIDatabase, NotesUIView, NotesUIDocument). Back-end classes can run anywhere (in the background or foreground, server or workstation), but front-end classes can only run in the foreground on the workstation. If you ran an agent containing a reference to one of the front-end classes in the background, the UI classes would not be found and the agent would not run. This would cause the agent to fail on loading. Load errors are hard to debug because they cannot be caught programmatically. The errors are displayed on the server console and in the server log, but some users either don't have access to these or don't think of looking.

Notes/Domino 6 handles UI classes in background agents in a way that should be easier to debug. Errors are now generated at the time the UI object is created rather than during the loading of the agent. This means it is acceptable to have a Dim statement referencing a UI class. The runtime error (error 217 "ErrAdtCreateError") will be generated when the UI object is being created. Because this is a runtime error, a simple error handling routine will allow you to catch this condition. This means you can write one agent that will run both on the client and the server, as long as you handle this error.

Here is an example of such an error handler:

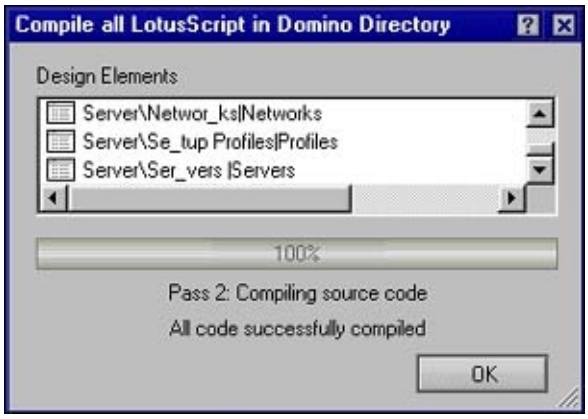
```
Function MayIUseFEClass() As Boolean
' error defined in lserr.lss
' Public Const ErrAdtCreateError = 217
    On Error 217 Goto NoYouMayNot
    Dim uiws As NotesUIWorkspace ' declare front-end class

    Set uiws = New notesuiworkspace

    MayIUseFEClass = True
    Exit Function
NOYOUMAYNOT:
    MayIUSEFEClass = False
    Exit Function
End Function

Recompile All LotusScript
```

Recompiling
When you modify a LotusScript script library and change class and type definitions, add or remove a parameter from a function, change a constant, or change the size of properties, a recompilation of LotusScript code is required. Designer 6 provides an option on the Tools menu called Recompile all LotusScript that lets you recompile all design documents within the database with one click. If there are errors in the code, these files will not be modified. Instead, a list of files with errors is generated. Remember that recompiling and re-saving the code will change the signature on that design document.



Java
In Notes/Domino 6, you can create script libraries in Java. These are created as part of the database and replicate with it. Java library routines can be called from LotusScript. Java objects can be accessed and manipulated from LotusScript.

Formula language
Formula language (the oldest of all programming languages in Notes) has been completely rewritten internally. This will improve efficiency and offer new functionality, without breaking existing applications. (Backwards compatibility has been our foremost priority in this re-architecture.) Not only is the formula engine faster, but the 64K limit has been removed. Here are some highlights of the new formula functionality:

- Instead of @Subset you can use subscripts. For example, before you would use:

  Filename:=@Subset(@Dbname;-1)

  Now instead, you can use:

  Filename:=@Dbname[2]

- Instead of @Set and @SetField you can use assignments, which can be nested:

  @if(FOO=”bar”;FIELD x:=”FOO”;y:=”bar”)

- You can use curly brackets to delimit strings as well as quotes, for example:

  REM {Code removed: days:=”Mon”:”Tue”:”Wed”:”Thu”:”Fri”;};

- Several looping constructs have been added, including @For, @While, @DoWhile.

- You can delete field x_1 through x_100 with an @For formula:

  @For(I=1;I<=100;I:=I+1; @SetField(“x_”+@Text(x);@DeleteField));

- You can compile and run a formula on the fly with @Eval:

  x={FIELD y:=@Prompt([OKCancelEdit];”Input”;”Input a value”;”Default”);};
  @Eval(x);

- @Sort function provides sort capabilities. To sort a text list by length, for example, you can do this:

  @Sort(x;[CustomSort];@Length($A) > @Length($B))

- In Notes/Domino 5, when you changed windows, the formula context did not change. Now it can be updated using @UpdateFormulaContext.

- @ThisValue and @ThisName allow you to write portable field formulas. For example you can write a generic field validation formula like this:


```
@If(@IsNull(@ThisValue); @Failure("Please enter value "+@ThisName+".");@Success);
```

- @ReplicaID returns the replica id of the database.
- @StatusBar prints a text message to the Notes status bar.
- @WebDBName returns the current database filename in URL format.

These are just a few samples. For more information on what's new in the formula language, see the *LDD Today* article, "**Enhancements to the formula language in Domino 6**."

Simpler security rules for formula and simple agents
One of our goals for this release was to make security rules for all agent types the same. This will help make these rules easier to understand and remember. Previously, both simple agents and formula agents had some quirks in their behavior to facilitate backward compatibility with Release 3. Notes/Domino 6 is still aware of the old rules and will behave appropriately, but we recommend you take advantage of the new features.

For example, as we mentioned earlier, formulas and simple agents that run on the server can access remote databases. This limitation has been removed for all agent types.

Also, prior to this release, private formula agents ran with the authority of the signer (similar to LotusScript agents), but shared formula agents ran under the authority of the replica ID. Now shared formula agents also run under authority of the signer. The replica ID is still respected for backward compatibility and is treated similarly to an alias name. So if you have existing databases that contain replica IDs in the ACL to allow @DBLookup formulas to access them, there is no need to change; these agents will continue to run without a problem.

Another change in simple and formula agents is agent-generated mail, specifically in how the name of the sender is determined. In earlier releases, rules for generating the sender field (the From field) for formula and simple agents were as follows:

Agent type	Shared
Formula agent "From"	Server
Simple agent "From"	Server

In Notes/Domino 6, all From fields are based on the effective user of the agent. To enact pre-6 behavior, you must sign the agent with the server ID; the mail will then come from the server.

Highlights of new classes and methods
Notes/Domino 6 contains a large set of new classes and methods. Here are some highlights of the areas covered by the new functionality:

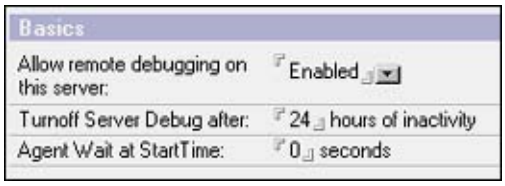
- Class NotesUIScheduler is a new front-end class built for Calendar and Scheduling. It allows you to programmatically control rooms, resources, and participants.
- Method NotesUIDocument.Close (bool Immediate) provides a solution to the "too many windows" problem.
- Methods Lock, Unlock, LockProvisional support document locking in Notes, including the ability to lock documents in databases with replicas on several servers, and work with locks in disconnected mode. Locking works on regular documents as well as design elements such as agents, views, and forms.
- An enhanced NotesReplication class and a new NotesReplicationEntry class provides full replication control, which allows you to programmatically control your replication formulas.
- Class NotesView is enhanced with many new methods. You can now create views with the CreateView method and add and remove columns with CreateColumn, CopyColumn, and RemoveColumn. Many properties are now read/write instead of read only. And a number of new properties have been added, for instance, SelectionFormula.
- The RichText class has undergone a major face-lift, with new methods and classes. A new Editing/Navigation framework provides navigation and manipulation of elements and ranges of elements. Table manipulation has been added as well.
- MIME support has been expanded. New features include the ability to fully navigate among entries. The NotesMIMEEntity class is now read/write. The NotesMIMEHeader class has been added.
- We've enhanced XML support in Notes. New classes include NotesDXLImporter, NotesDXLExporter, and NotesXSLTTransformer.
- The NotesStream class has been added as a helper for MIME and XML. It provides an efficient transport of

large data sets. It supports both text and binary data.
- NotesDatabase, NotesRegistration, NotesACL, and NotesAdministrationProcess have been enhanced with many new methods and properties to allow automation of administrative tasks. The Domino Web Administrator client is written using back-end classes exclusively to do everything it needs, so the coverage is very thorough.

Remote debugger
Notes/Domino 6 includes a remote debugger for server-based agents. The remote debugger can be used not only to debug agents, but also to see all agents executing on the server (including ones run by processes such as http and router). You can also use the remote debugger to cancel any agent run by any server process.

You can initiate a debug session from either the client or the server console. If you do it from the client, the interface looks very similar to the familiar client LotusScript debugger. You will be able to single step, set breakpoint, examine and change variables, list all agents, and kill any running agent.

Remote debugger is implemented as a server add-in task. To enable remote debugging, you must enable it on the server. You do this through the Server Tasks/Remote Debug Manager tab in the Server document. Plus, the agent itself has to be marked as debuggable.

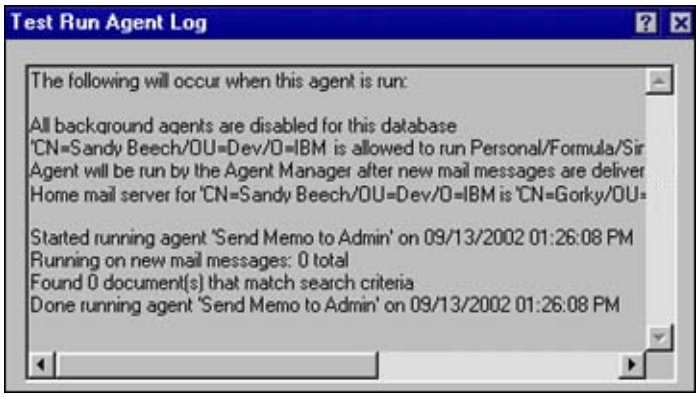


To invoke the remote debugger, choose File - Tools - Debug LotusScript Server Agent. Then select the server and database where the agent you want to debug resides. Agents marked as debuggable start running, and their names appear in the Debug Target window. At this point, you can select the agent you want to debug and then click Open. This begins the debug session. Note that you can debug and/or cancel agents executed by any process, not only the agent manager.

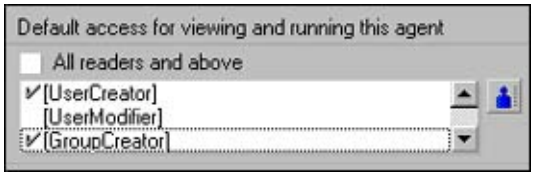
Other new agent features
Security and programmability aren't the only areas of agent functionality we've enhanced. We've also added and/or improved:
- Agent Test
- Reader lists
- Console commands
- Multithreaded client
- New agent builder UI
- Processing agents in templates
- Better performance

The following sections describe these and other new features.

Agent Test
In Designer 5, we added the Agent Test feature, an expert system that allows you to diagnose common mistakes in agent security and scheduling. Agent Test has been updated to work with the Notes/Domino 6 security model and provides diagnostic messages that will help adjust security settings if necessary. Agent Test is also smart enough to switch to the Release 5 security model if you are using a Notes 6 client to look at the agent on a Domino 5 server. To start Agent Test, click the agent in Designer, and then choose Agent - Test. The following is an example of Agent Test output:



Reader Lists in agents
In Notes/Domino 6, reader lists have been added to agents. They allow you to specify that an agent will be hidden from all users except those you name. If someone who is not included in the reader's field of the agent attempts to execute the agent, even if they know the agent name, they would get an error that this agent does not exist. To specify a reader list, open the Agent properties box and select or enter names in the reader list box near the bottom of the Security tab.



Console Commands
We have added several new server console commands for agents. These let you display online documentation for all agent manager commands, run and cancel agents, and display the names of agents in any given database. The new commands are:

Load amgr -?
This command returns a help index for the agent manager.

Tell amgr run "database name" 'agent name'
This command runs an agent in a separate thread. There is no limit to how many agents can be executed at the same time, other than the actual limits imposed by the size of your computer. The agent will be executed following all the rules of the server-based agents, as if invoked on a schedule. This means it will run with the rights of the signer, unless the On behalf of field is present, which would take precedence. This command is a handy way to do administrative tasks as well as test server-based agents.

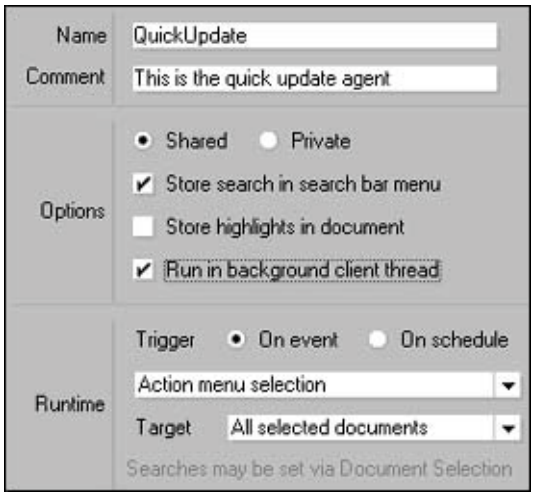
Tell amgr cancel "database name" 'agent name'
This command aborts a scheduled agent that is currently running by the Agent Manager. Note this does not cancel agents executed by other processes, such as http or router. However, you can cancel agents executed by any process if you use the remote debugger. Also, if a third-party application hangs and does not return control back to Notes, the agent cannot be canceled.

Tell amgr show [-v] "database name"
The abbreviated form of this command (without the *v*) displays the names of the agents in the database. It shows a list of all shared agents with a total number of agents, followed by a list of all private agents with a total number of agents. The verbose format of the command (with the *v*) shows all agents and script libraries with additional information for each.

Multithreading of the client
The Notes 6 client allows you to multitask. You can run several agents invoked from the menu at the same time in background threads. There is no limit to how many agents can be executed at the same time, other than the performance limits of your PC. The only limitation on the background thread agents is that as with background

agents, these agents cannot have UI functions.

New agent builder interface
Domino Designer 6 contains a new interface for the agent builder. To give you more room for coding, we removed agent configuration information from the programming pane and placed it in the Agent properties box. This has two tabs, one for scheduling and other information and the other for security-related settings. A radio button on the first tab lets you quickly toggle between making the agent shared or private. Another new feature on the first tab is the Run in background client thread checkbox. (See the previous section for a description of client multithreading.) The first tab of the Agent properties box appears below:



The second tab, which we've already shown you **earlier** in this article, contains all security-related settings. This tab contains information needed for new features, as well as previously available security settings such as Run as Web user and Allow public access. In addition to designating the agent to run as a Web user, you can specify the name of the user on whose behalf the agent is running, whether the agent is remotely debuggable, and whether this agent can be activated by an editor-level user. The drop-down list allows unrestricted users to set the mode in which the agent is running. You can also set readers fields for the agent.

Processing agents in templates
Based on feedback from customers, agents will no longer run in templates with the NTF extension. (They will, however, continue to run in templates with NSF extensions.) Because it will now be possible to have enabled agents in the templates, let me highlight some of the potential issues with deploying enabled agents. When something is inconvenient, typically there is a security-related issue to blame. This case is no exception. When trying to evaluate if you should deploy your agents enabled, consider the following two issues:

- Most frequently the person who signed the agent in the template is not the person under whose rights the agent should be running when the agent is deployed. If the agent is deployed disabled, with an option Choose when enabled selected, the user will have to enable the agent. This will sign the agent with an appropriate ID and prompts the user for the name of the server where the agent should run when enabled.
- In many cases, agents are deployed on the server whose name is not known at development time, and thus cannot be set in the template. Using the Choose when enabled option allows the user to specify the server.

Performance enhancements
Two performance enhancements are worth mentioning. There has been performance work on agent manager startup, which should result in quicker boot up. Databases with no enabled agents are marked in a special way and never need to be examined for agents. This information is now cached and shared by various processes, which should speed server start time.

The second performance enhancement should improve loading time for the agents loaded more than once without being modified. Some results of the "expensive" cryptographic operations are now cached. One example in which this change can benefit is Web agents, where the same agent might run many times without being changed.

Notes/Domino 6 agents: more features, easier to use

As you can see from this overview, Notes/Domino 6 contains many new features for agents, especially in the areas of security and programmability. I hope you found this article helpful. In future issues of *LDD Today*, I'll provide more in-depth details about Notes/Domino 6 agent security, as well as update my previous agent-related articles to ensure they're current with our latest release. These articles include:

- **Minimizing delays in the Agent Manager** (scheduling agents)
- **Demystifying the Out of Office agent** (the Out of Office agent, common mistakes, and customization)
- **Run and RunOnServer: Adding Parameters** (passing parameters between agents)
- **Controlling the agents in your system** (agent security)
- **Troubleshooting agents** (common mistakes, how to fix them, and general troubleshooting techniques)

ABOUT THE AUTHOR

Julie Kadashevich came to Iris in March of 1997 after working at FTP Software on Java and C++ mobile agent technology. This is her sixth article for *LDD Today.* She focuses on the Agent Manager as well as Java. Previously, she worked in the area of applied Artificial Intelligence at Wang Labs and received five patents in the field. She has Bachelor's and Master's degrees from Boston University, both in computer science.

