

Level: Intermediate

Works with: Notes/Domino 5 and 6

Updated: 02-Jan-2003

by
Julie
Kadashevich

In March of 1998, we published "[Troubleshooting agents](#)," one of the most popular articles that Notes.net/LDD has ever offered. To this day, it remains a favorite with high reader interest. Of course, a lot has changed since that article made its first appearance. For example, we shipped not one but two major releases: Notes/Domino 5 and 6. So although much of the article remains relevant, we felt it was time to update it with more current information.

However, we still want to retain the earlier article, which our numbers show is still frequently used by our readers. So we made the decision to keep the original and to offer a completely new article dedicated to troubleshooting agents in Notes/Domino 5 and 6. This way users running Domino 4.6 (and there's quite a number who do) still have the first article available to them, while the rest of us running later releases now have the most current, all-inclusive information at the ready.

As with the earlier article, this article examines steps you can take to troubleshoot agents and to find out why your agent isn't running as you expect. We start by showing you the tools available for troubleshooting agents in Notes/Domino 5 and 6, including Notes.ini settings, the LotusScript NotesLog class, and server console commands. Then we discuss some of the more common issues you may encounter when designing and running agents, offering possible causes and solutions.

This article assumes that you're an experienced Notes/Domino application developer, ideally with some system administration knowledge.

A quick review of agents

When troubleshooting agent problems, it's best to start with a good understanding of how agents and the Agent Manager work. So let's begin by reviewing how agents are invoked.

On the server, agents can be invoked by:

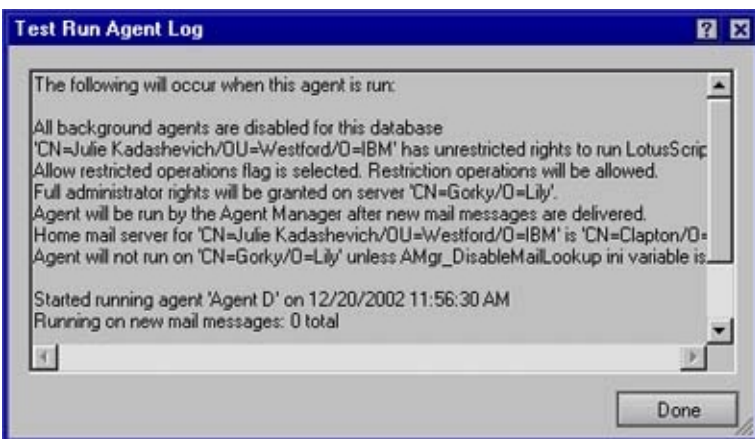
- The Agent Manager, a server task that controls scheduled agents
These include agents that run on a schedule specified by the user or that are scheduled when an event occurs (such as new mail or new document creation).
- The HTTP task, running agents invoked via a Web browser
- The router, running agents executed as part of mail delivery ("before new mail arrives" agent trigger)
- The RunOnServer method, included in the code of the agent running on the client
- The server console command "Tell amgr run" (This is a Notes/Domino 6 only command.)

On the client, agents can be invoked as:

- Interactive agents invoked by the Notes client, as a result of a user action (menu pick or button click)
- Locally scheduled agents invoked by the Client TaskLoader, running scheduled agents on the client using an embedded version of the Agent Manager (Notes/Domino 6) or the Agent Manager task running locally on the client (R5).

If your agent won't run, the first thing you need to do is learn more about the nature of the problem. Start by looking at the Agent Log. The Agent Log shows you the last time the agent executed and whether or not it completed its execution.

Starting with R5, Domino Designer includes Agent Test, an expert system that helps you diagnose common mistakes in agent security and scheduling. Agent Test is a handy tool, providing diagnostic messages that help you adjust agent settings as necessary. In Domino Designer 6, we updated Agent Test to work with the Notes/Domino 6 security model. If you are using the Notes 6 client to look at an agent on a Domino 5 server, Agent Test will automatically switch to the R5 security model. To run Agent Test, open Domino Designer and go to the Agents view in your database, highlight the agent to test, and then select Agent - Test.



For additional information about a server-based agent, examine the server console or Notes Log (under Miscellaneous events) for any messages from the malfunctioning agent. If you don't have physical access to the actual server console, you might have access to the live console by choosing File - Tools - Server Administration and clicking the Console button. If you don't have access to the server console, you cannot issue the commands that we cover later in this article, but you can see the output generated during an attempt to execute the agent in the Domino Server Log.

If you are using LotusScript, you can use the Debug LotusScript option to diagnose client-based agent problems. In Notes/Domino 6, you can use the Remote Debugger to diagnose server-based problems. To start the client-side debugger choose File - Tools - Debug LotusScript; to start the Remote Debugger choose File - Tools - Remote Debugger. These debuggers can diagnose problems that occur during code execution. Keep in mind that if the agent encounters a security problem with the rights to run the agent on the server, or if the agent is set to run on a wrong server, the remote debugger can't start. The good news is that agents never fail silently; later in this article, we discuss how to get the maximum information about the nature of the problem. In addition, not all agents are well-suited to being debugged interactively, so we'll also look at how you can use the NotesLog class to debug agents.

Useful Notes.ini settings

There are a number of Notes.ini settings you can use to analyze agent problems. You set these variables in your server's Notes.ini file when debugging scheduled server-based agents, or in your Notes client's Notes.ini file when debugging client-based scheduled agents. Note that these variables are only used for debugging scheduled agents run by the Agent Manager. As we mentioned earlier, HTTP, router, console, RunOnServer, and interactive client agents are not run by the Agent Manager. Therefore the Notes.ini variables discussed in this section have no effect on these agents. (We discuss how to debug agents run by tasks other than Agent Manager later in this article.)

The Notes.ini variable Debug_Amgr turns on Agent Manager debugging, which can be useful for diagnosing problems with scheduled agents (both on the Domino server as well as the Notes client). When you discover that your scheduled agent won't run, the first thing you can do is add the following line to your Notes.ini file:

Debug_AMgr = flag

where *flag* can be one or more of the following (listed in alphabetical order):

Flag	Description
c	Outputs agent control parameters.

e	Outputs information about Agent Manager events.
l	Outputs agent loading reports.
m	Outputs agent memory warnings.
p	Outputs agent performance statistics.
r	Outputs agent execution reports.
s	Outputs information about Agent Manager scheduling.
v	Indicates verbose mode, which outputs more messages about agent loading, scheduling, and queues.
*	Outputs all of the above information (same as setting all the flags).

Setting `Debug_Amgr=*` provides the most information, but it may generate more output than you want. Also, be aware that turning on all debugging flags typically results in a five percent performance hit on average user response time.

The output appears in the console log and the Notes Log. This variable can also be set interactively from the server console (we discuss the console commands later in this article). If you change the `Notes.ini` file, you must restart the Agent Manager (or the server). If you set the variable using the console commands the restart is not necessary.

You can also turn on agent execution logging by adding the following line to the `Notes.ini` file. (You can also do this in the server's Server Configuration document in the Domino Directory.)

`Log_AgentManager = value`

where *value* can be one of the following:

Value	Description
0	Does not show logging.
1	Shows partial and complete successes.
2	Shows complete successes.

The `Log_AgentManager` setting provides you with a subset of the debugging information that `Debug_AMgr` generates. This option provides less output, but also has less impact on performance. If you have problem agents, you may want `Log_AgentManager` turned on at all times to record additional information in the log. If you have both `Log_AgentManager` and `Debug_AMgr` specified, the `Debug_AMgr` settings take precedence.

Background agents (by definition) cannot generate output to the user interface (UI). All output generated by the background agent (for example, print statements) goes to the server console and to the Notes Log under Miscellaneous events. The same is true of the error and warning messages generated on behalf of the agent. So, it is important to always examine the server console or Notes Log for this information.

You also can record server console messages in a text file by adding the `Debug_Outfile` setting to your server's `Notes.ini` file. For example, `Debug_Outfile="C:\mydebug.txt"` instructs the server to output console messages to a file called `mydebug.txt`. You might find it easier to search this file rather than to examine entries in the Notes Log or the server console. Bear in mind, however, that the extra I/O can adversely affect performance.

If you are running locally scheduled agents, the debugging information does not appear on the server console because Agent Manager code runs locally on the client rather than on the server. In this case, you can output the information to a text file by adding the `Debug_Outfile` setting to your client's `Notes.ini` file. Then the file will be created locally on your workstation.

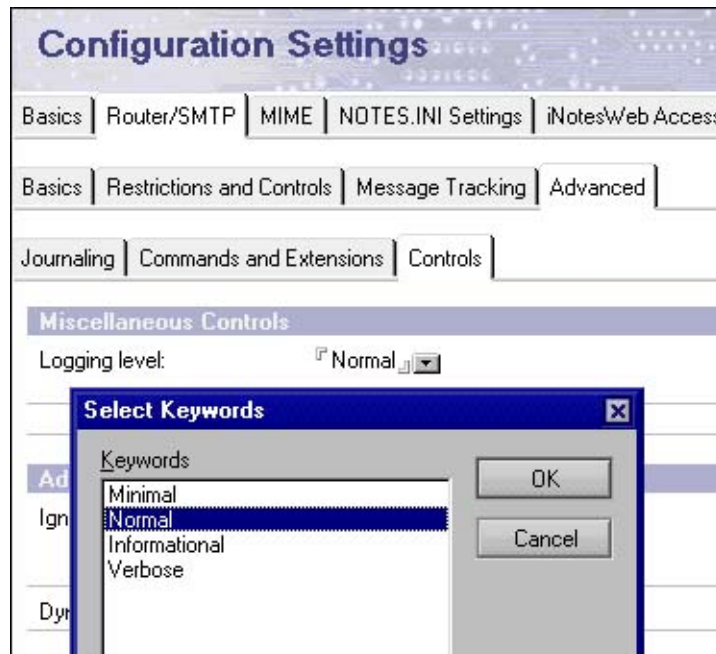
For more on `Notes.ini` variables that affect agents, see "[Agent variables](#)," the Ask Professor INI column in the [December issue of LDD Today](#). Also consult the article "[Minimizing delays in the Agent Manager](#)."

Error messages generated by router and Web agents

As mentioned in the introduction, the Agent Manager is not the only process that can run agents. In this section, we highlight how to view diagnostic information in router and Web agents.

Because the router executes pre-delivery mail agents, the Agent Manager `Notes.ini` settings for logging and debugging have no

effect. Instead, use the Logging level field in the Router/SMTP - Advanced - Controls tab of the server's Configuration Settings document to control the level of errors written to the server console for pre-delivery agents. You can set the Logging level to Minimal, Normal (the default), Informational, or Verbose.



If you select Verbose, your agent errors are logged to the console. Note that you also get a very verbose output of all router operations. By using this separate setting to control the output of pre-delivery mail agents, you can tune the performance of the router and still have your other agents generate as much output as needed for your other operations. The output from the pre-delivery agent on the server console is prefixed by Addin: as shown in the following example:

Addin: Agent message box: Message generated by mail pre-delivery agent.

This happens because the messages are generated by the router task (which is executing agent routines via API calls), and the router task is a server add-in.

The default setting for errors returned by agents in the browser suppresses Notes/Domino errors and displays a generic error message ("HTTP 500 - Internal server error"). Web developers can get a more meaningful error message by opening Microsoft Internet Explorer, selecting Tools - Internet Options - Advanced, and de-selecting the "Show friendly HTTP error messages" checkbox. This lets you see the actual error message generated by Domino.

Debugging with the NotesLog class

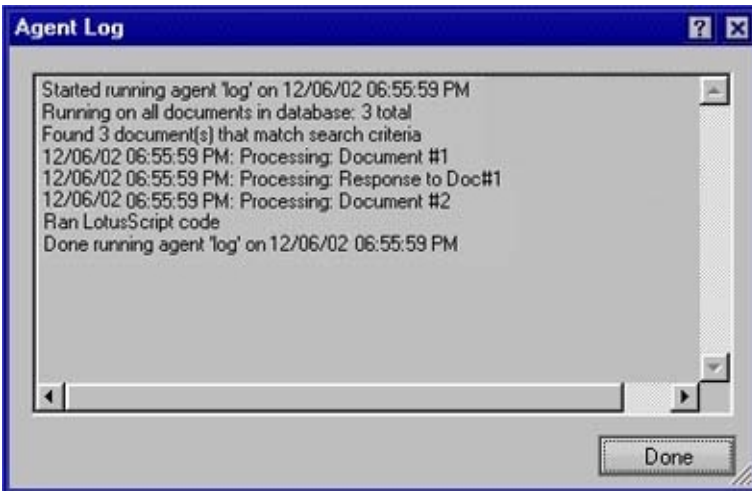
You can use the NotesLog class to output information to the Agent Log. This can help you debug LotusScript and Java agents. The Agent Log is a valuable tool for error handling and checking variable values, as well as checking the logic of your agent. For example, you can see whether or not the Search Builder is selecting documents you expected the agent to find.

The following is a sample LotusScript agent that processes all unprocessed documents, then puts the title of each document into the Agent Log. The four lines of code that generate an Agent Log appear in bold:

```
Dim agentLog As New NotesLog("Agent log")
Call agentLog.OpenAgentLog
Set s = New NotesSession
Set db = s.CurrentDatabase
Set collection = db.UnprocessedDocuments
Set note = collection.GetFirstDocument
count = collection.Count
Do While (count > 0)
    Subject = note.Subject
```

```
Call agentLog.LogAction( "Processing: "+Subject(0))
Set note = collection.GetNextDocument (note)
count = count - 1
Loop
Call agentLog.Close
```

To examine the generated log, select your agent, and then choose Agent - Log (or right-click and select Log). The following illustration shows the Agent Log generated by our sample agent:



As an alternative to logging errors in the Agent Log, you can write them to a text file. The following code is an example of an error handler that writes errors to a file:

```
On Error DivisionByZero Goto LogError
Dim errorLog As New NotesLog( "Errors" )
Dim x As Integer, y As Integer, z As Integer
Call errorLog.OpenFileLog( "c:\errlog.txt" )
y = 13
z = 0
rem The following line will generate an error for this example
x = y / z
Call errorLog.Close
Exit Sub
LogError:
Call errorLog.LogError( ErrDivisionByZero, Error$( ErrDivisionByZero ) )
Resume Next
```

Debugging at the server console

The following server commands are useful for troubleshooting agent problems in either Domino 5 or 6:

- Tell amgr schedule
- Tell amgr status
- Tell amgr debug

In addition, Domino 6 offers these commands to help troubleshoot agent problems:

- Tell amgr run
- Tell amgr cancel
- Tell amgr show
- Load amgr -?

Tell amgr schedule

The Tell amgr schedule command displays the Agent Manager schedule of all agents scheduled to run for the current schedule period. The start of the current schedule period is defined in the server record as "cache refresh time." The current schedule period is not constant (in other words, it does not look 24 hours ahead at any point of the day). Instead it starts with 24 hours and

counts down to zero, when the schedule for the next 24 hour schedule is built. Newly scheduled agents in the current schedule period are added to the schedule, but the Agent Manager does not need to keep track of agents that are not scheduled to run in the current schedule period. This approach allows the Agent Manager to handle its schedule queue in the most efficient way.

The Tell amgr schedule command shows the agent trigger type, the time the agent is scheduled to run, the name of the agent, and the name of the database on which the agent runs. Checking the Agent Manager schedule is useful for debugging because it lets you see if your agent is waiting in one of the three Agent Manager queues:

Queue	Name	Description
E	Eligible to run	Queue in which agents are eligible to run
S	Scheduled	Queue in which agents are scheduled to run
V	Event	Queue in which event-triggered agents are waiting for their event to occur

Scheduled agents are queued in the S (scheduled) queue. When the time that they are scheduled to run arrives, they are placed in the E (eligible to run) queue. Event-triggered agents (new mail and document creation/modification agents) appear in the V (event) queue until their triggering event occurs. When this happens, the agents move first into the S queue, then finally into the E queue.

In addition, there are three trigger types:

Trigger	Description
S	Agent is scheduled to run
M	Agent is a new mail-triggered agent
U	Agent is a new/updated document-triggered agent

Here is a sample output from the Tell amgr schedule command:

E	S	04:03 PM Today	agent1	SALES.NSF
S	S	05:04 PM Today	agent2	SALES.NSF
V	U		agent3	SALES.NSF

The first column of the output lists Agent Manager queue type. The second column contains the agent trigger type. The remaining columns show the time the agent is scheduled to run, the name of the agent, and the database name.

In the sample output, agent1 is a scheduled agent that has been moved to the E queue; it will run as soon as the Agent Manager is free to run it. Agent2 is a scheduled agent waiting in the S queue for its scheduled time to arrive. Agent3 is an event-triggered agent waiting in the V queue for a document to be created or modified.

Agents typically require a minute or so to appear in queues or to move from one queue to another. For information on the different factors that control scheduling, see the *LDD Today* article "[Minimizing delays in the Agent Manager](#)."

Tell amgr status

The Tell amgr status command gives you a snapshot of the Agent Manager status. You can check the status of Agent Manager queues and control parameters. This lets you review which parameters are currently in effect. For example:

```
12/26/02 10:30:15 AM  AMgr: Status report at '12/26/02 10:30:15 AM'
12/26/02 10:30:15 AM  Agent Manager has been running since '12/22/02 02:18:25 PM'
12/26/02 10:30:15 AM  There are currently '1' Agent Executives running
12/26/02 10:30:15 AM  There are currently '4' agents in the Scheduled Task Queue
12/26/02 10:30:15 AM  There are currently '0' agents in the Eligible Queue
12/26/02 10:30:15 AM  There are currently '0' databases containing agents triggered by new mail
12/26/02 10:30:15 AM  There are currently '0' agents in the New Mail Event Queue
12/26/02 10:30:15 AM  There are currently '3' databases containing agents triggered by document updates
12/26/02 10:30:15 AM  There are currently '3' agents in the Document Update Event Queue
12/26/02 10:30:15 AM  AMgr: Current control parameters in effect:
```



```
12/26/02 10:30:15 AM AMgr: Daily agent cache refresh is performed at '12:00:00 AM'
12/26/02 10:30:15 AM AMgr: Currently in Daytime period
12/26/02 10:30:15 AM AMgr: The maximum number of concurrently executing agents is '1'
12/26/02 10:30:15 AM AMgr: The maximum number of minutes a LotusScript/Java agent is allowed to run is '10'
12/26/02 10:30:15 AM AMgr: The maximum percentage of time agents are allowed to execute is '90'
12/26/02 10:30:15 AM AMgr: Executive '1', total agent runs: 414
12/26/02 10:30:15 AM AMgr: Executive '1', total elapsed run time: 273
```

Tell amgr debug

Tell amgr debug lets you display current debug settings for the Agent Manager or set new ones. When setting debug values with this command, you can use the same flags we described earlier for the Notes.ini setting Debug_AMgr. These settings take effect immediately without restarting Agent Manager or the server.

To set debug settings, type:

```
>tell amgr debug *
02/12/98 02:03:14 PM AMgr: Current debug control setting is 'mecvrspl'
```

To see the current debug setting you can enter the command without arguments:

```
>tell amgr debug
02/12/98 02:03:15 PM AMgr: Current debug control setting is 'mecvrspl'
```

There is no command to turn off the debugging mode, but you can achieve the same result in one of two ways:

- Reset the debug mode to other than verbose (in this example, we chose c to output agent control parameters):

```
>tell amgr debug c
10/21/02 21:00:01 PM Amgr: Current debug control setting is 'c'
```

- Quit and reload amgr

Tell amgr run "database name" 'agent name' (Notes/Domino 6 only)

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 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 "Run on behalf of" field is present (which would take precedent). This command is a handy way to perform administrative tasks, as well as to test server-based agents. (Note that the agent name has to be in single quotes.)

Tell amgr cancel "database name" 'agent name' (Notes/Domino 6 only)

This command aborts a scheduled agent that is currently being run by the Agent Manager. 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" (Notes/Domino 6 only)

The abbreviated form of this command (without the v) displays the names of the agents in the database, listing private and shared agents separately with totals. The verbose format of the command (with the v) shows all agents and script libraries with additional information for each. It displays the language the agent is written in, who it is signed by, and the effective user of the agent. This data can be helpful in debugging security issues.

Load amgr -?' (Notes/Domino 6 only)

This command returns a help index for the Agent Manager.

Common agent problems

Now let's turn our attention to typical problems you may run into when developing agents and possible solutions for these problems.

My agent runs from the UI, but not as a scheduled agent

This is usually caused by: (1) a security issue, or (2) a UI class in the agent. For information on agent security, see the sidebars "[Notes/Domino 6 agent security at a glance](#)" and "[Notes/Domino 5 agent security at a glance](#)."

Back-end classes perform operations on Notes databases, and front-end classes handle UI (NotesUIWorkspace, NotesUIDatabase, NotesUIView, NotesUIDocument). Back-end classes can run anywhere (in the background or foreground,

server or workstation). Front-end classes can run only in the foreground on the workstation.

Prior to Notes/Domino 6, when any UI classes were used in a background agent (even if only a Dim statement for NotesUIWorkspace), the agent would not run in the background. 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 caused the agent to fail on loading. Load errors are difficult to debug because they cannot be caught programmatically. The errors appear on the server console and in the server log (Unknown LotusScript error or "Error loading USE or USELSX module: XXX"), but some users either don't have access to the server log or don't think of looking at the log.

Notes/Domino 6 handles UI classes in background agents in a way that is 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 that it is acceptable to have a Dim statement referencing a UI class. The runtime error (error 217 ErrAdtCreateError) is generated when the UI object is being created. Because this is a runtime error, a simple error handling routine allows you to catch this condition. This means that you can write one agent that runs on both 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
NOYOU MAYNOT:
MayIUSEFEClass = False
Exit Function
End Function
```

My unrestricted agent does not run (Notes/Domino 6 only)

You may have an agent that was signed by an unrestricted signer and that runs fine in R5, but after editing it with Domino Designer 6, the agent stops running. In this case, check the security tab of the Agent Properties box to see the runtime security setting:



An agent that has not been edited with Domino Designer 6 runs on a Domino 6 server with the rights of the signer just as it would

in R5. So in the case of an unrestricted signer, the agent runs with unrestricted rights. But if the agent is edited by Domino Designer 6, the default setting is "Do not allow restricted operations" (the safer setting). If you want to perform unrestricted operations in the agent after you have edited it with Domino Designer 6, you need to explicitly set this security setting.

My scheduled agent does not run in a client replica

Locally scheduled agents are disabled by default. To enable scheduled agents on the client, select File - Preferences - User Preferences, then check Enable scheduled local agents.

My mail agents do not run

"After new mail has arrived" agents are designed to run on the home mail server of the agent owner (signer). If the agent is replicated from your home mail server to another system, or if you are trying to run an agent that was written by someone else, your home mail server will not match the server on which you are attempting to run the mail agent, and the agent will not run. You can suppress the check for the home mail server by adding the following setting to the server's Notes.ini file (supported in R4.5 and later):

AMgr_DisableMailLookup =value

where *value* can be 0 (check for the home mail server) or 1 (don't check for the home mail server). The default is 0.

By suppressing the check that forces the agent to run only on the home server, you can have the agent run on multiple servers. Depending on the logic of your agent, allowing it to run on more than one server may cause replication conflicts and processing of the same logic more than once.

"Before new mail has arrived" agents executed by the router run on the server on which the mail is delivered; the concept of home mail server does not apply in the case of this trigger. AMgr_DisableMailLookup has no effect on pre-delivery mail agents.

Both before and after mail delivery agents do not use the server name that may be stored in the agent to determine which server to run on. For that reason, the option to set the server name is not available via the Agent Properties box.

What does "Unsupported trigger and search in the background or embedded agent" mean?

This error is generated in a server-based agent (which can be a Web agent or any other agent invoked on the server) or embedded agent (an agent called by another agent) which references UI elements as trigger/search target settings in the agent builder. For example, suppose you create an agent that has action menu selection as a trigger and all selected documents as a search target. When you invoke the agent from the browser, you receive the unsupported trigger... error. The "all selected documents" concept is not understood within a Web agent; this setting is meaningful only in a Notes client. To fix the problem, change the target setting, for instance, to all documents or none. The following four options are not supported in this situation:

- All unread documents in view
- All documents in view
- All selected documents
- When documents are pasted documents

My scheduled agents do not run in MAIL.BOX

If scheduled or event-triggered agents are created in the MAIL.BOX database, these agents are never automatically scheduled to be executed. This happens because MAIL.BOX is an internal database—it is not treated as a regular database, and the Agent Manager does not scan it for scheduled tasks. Documents in the MAIL.BOX database are transient by nature; they stay in the database for very short period of time. The Agent Manager examines its queue no more frequently than once a minute, so in all probability some documents in MAIL.BOX would never be seen by a scheduled agent allowed to run on ".BOX" databases.

I assigned execution rights to users, but they're still rejected

As you may know, you assign execution rights for users in the Programmability Restrictions fields in the Security section of the Server document. These fields have a maximum limit of 256 characters. Any names entered beyond 256 characters are not seen by the server, and these users are rejected for not having the proper execution rights. If you exceed this limit, you receive a warning while editing the field. You also see this warning periodically while the Agent Manager is running. You receive the following warning on the server console:

12/12/02 14:05:45 AMgr: Agent execution ACL is longer than 256 characters. Use groups.

To solve this problem, use group names in the Agent Restrictions fields.

Something is disabling my agent

The Design Update task is a system process that updates database design from a database template. This task's goal is to keep the database and the template in synch. When either the database or the template have been changed, and the user did not select the "Do not update" design property for specific design elements, the Design Update task updates the database to match

the template. This sometimes causes problems for agents, typically described as "something is turning off my enabled agents in the middle of the night!"

This happens because the user has made a significant change in the agent in the database. By significant change, we mean something other than enable/disable status of the agent or the name of the server on which the agent is scheduled to run. The most common change that causes this problem is the change to the schedule of the agent, but it also could be a change to the agent code or some other part of the agent design.

Prior to Notes/Domino 6, the Design Update task recognized changes in the database as well as the template. In Notes/Domino 6, Design Update now keeps track of changes only in the template. This lets you make changes to the database and keep them, unless the template itself had some significant changes. Changes you make to the template take precedence (because typically the changes in the template indicate bug fixes or some other administrative change). If the template is used to update the database, agents revert to the versions in the template, which typically are disabled.

You can suppress the agent design update by selecting the option "Do not allow design refresh/replace to modify" in the Agent Properties box.

It is necessary to disable the agent because when the agent is updated from the template, the agent from the template (including the signature in the template) replaces the agent in the user database. The agent is disabled to force the user to sign it and thus, to make it run with the appropriate security rights. (Otherwise, the mail would come from Lotus Notes Template Developers among other undesirable things.) Because the user ID file does not exist on the server, it is not possible to re-sign the agent with the user ID automatically. The only ID that is available on the server is the server ID, but having the agent run under the server ID is also undesirable.

I want my server-based agent to run immediately

You can write a foreground agent that invokes a server-based agent, using the method `RunOnServer` in the Agent class. You can then run the agent through the client interface, which runs the agent on the server. To do this, use the following code fragment:

```
agent.RunOnServer(),
```

where *agent* is the agent you want to execute on the server. The call is synchronous, so the workstation waits for the agent to complete before returning. This agent runs on the same server as the database on which it resides. One of the benefits of this method is that if you have an agent that performs database operations, you do not need to have a copy of the database software on the client.

The workstation component of this type of agent has different security from the server component. Each component follows one of these rules:

- For the calling agent running on the workstation, the ACL rights are determined by the rights of the invoker and agent restrictions are not used.
- For the called agent running on the server, the ACL rights are based on the signer (and the "Run on behalf of" identity in Notes/Domino 6) and the agent restrictions are determined based on the signer of that agent.

For more information on agent security and how it can affect when your agents run, see the sidebars ["Notes/Domino 5 agent security at a glance"](#) and ["Notes/Domino 6 agent security at a glance."](#)

Another option new to Notes/Domino 6 is the ability to issue a server console command that executes an agent. This command can be issued either from the server console or programmatically:

```
Tell amgr run "database name" 'agent name'
```

How can I change the apparent sender of agent generated mail?

Sometimes you need to generate mail from an identity different from your own. For example, you may want to generate mail from the user Domino Administrator. There are three ways to do this:

- Create a special ID for user Domino Administrator and sign the agent with that ID.
- Use the Principal field in the agent code to override the From field.
- Use the Run on behalf of field (new in Notes/Domino 6).

The first method is the simplest because it requires no coding. It does, however, involve licensing and maintaining IDs. The second method is more flexible, but requires some coding. You don't need any special rights, but the information about the original sender is maintained for auditing purposes. By default, the mail template displays the original sender's name in the sent by field under the From field. There are three ways to specify the Principal field:

- `doc.Principal="Joe User/Org@NotesDomain"` where Joe User/Org has a Person record with an InternetAddress of your choosing. (Note that the string @NotesDomain must be present.)

- doc.Principal="CN=Joe User/O=Org" where Joe User/Org has a Person record with an InternetAddress of your choosing.
- doc.Principal=User@acme.org@NotesDomain <mailto:User@acme.org@NotesDomain>. (Note that the string @NotesDomain must be present.)

If you only care about the ReplyTo address, you can use it instead of the Principal, which changes both the From and ReplyTo:

```
Sub Initialize
Dim session As New NotesSession
Dim db As NotesDatabase
Dim view As NotesView
Dim doc As NotesDocument
Dim item As NotesItem
Dim ToWho(40) As String
Dim FirstName(40) As String
Dim Msg As String
Set db = session.CurrentDatabase
Set doc = New notesdocument(db)
*****

Count = 30
ToWho(0)="joe@yahoo.com"
FirstName(0)="Joe"

ToWho(1) = "teresa@yahoo.com"
FirstName(1)="Teresa"

<more initialization code removed>

doc.Principal = "julie@yahoo.com@NotesDomain"
doc.Form = "Memo"

doc.Subject=" Happy Holidays!"
For i=0 To count
Greet = "Hi " +FirstName(i) +", Happy Holidays!"
Msg1 ="additional text"
doc.Body=Greet + Msg1
Print toWho(i)
Call doc.Send(True,toWho(i))
Next
```

As we show in the preceding code, @NotesDomain is a string that is expected in the syntax. Omitting this is often the problem when people report that the Principal field is not working.

The third method (Run on behalf of) is new to Notes/Domino 6. This field in Agent Properties allows one user to run an agent as though it were invoked by another user. You need special rights to run agents on behalf of other users. These rights are controlled through the "Sign agents to run on behalf of someone else" field in the Security tab of the Server document. If they have this permission, mail appears to come from the name specified in the Run on behalf of field. This is a very high level of rights and should only be granted carefully, as it allows one user to run agents on behalf of other users, including having their ACL rights when accessing databases.

How do I access databases on a server other than the database where my agent is running?

Prior to Notes/Domino 6, reaching remote servers through an agent was supported in the client agents, in locally scheduled agents, and through CORBA-remoted Java backend classes and DIIOP. However, Domino did not have a security protocol to properly authenticate agents running on the server under user rights with the remote server. So even though it was possible to write the code that accessed a remote server with API routines, this could not be done securely, so we didn't allow it in the agents.

In Notes/Domino 6, we have added a new security protocol to make this possible. The server whose data the agent wants to access needs to grant this permission to the server where the agent is running. This level of trust is set up in a new Domino Directory field called Trusted servers.

What is the difference between "after new mail has arrived" and "after documents are created or modified"?

The difference is that mail-based triggers know what constitutes mail and do not select items such as drafts and sent mail. These

triggers pick up newly modified, previously delivered mail because in some situations it is necessary (for example, when mail is replicated to your mail database rather than delivered via the router). If this is not what you want, you can create an agent that marks a document with a flag when you process it the first time, then ignores it on the next round. (This is necessary only if you anticipate modifying newly delivered mail.)

Note that "before delivery agents" only work for mail delivered via the router.

Why do "after new mail has arrived" and "after documents are created or modified" agents process more documents than I think they should?

An agent of this type reprocesses documents if:

- The agent itself has been changed and re-saved. (It's possible that the logic of the agent completely changed.)
- Documents the agent is processing have been changed.

My agents are creating replication conflicts

Replication conflicts created by agents are no different from replication conflicts created by other means. They happen because more than one entity (a person, a process, or an agent) modified the same note at the same time. Some common situations to check for:

- Agents with the wild card Any Server specified as the server to run on
- "After new mail has arrived" agents running on the server whose Notes.ini file contains the variable AMgr_DisableMailLookup, which suppresses the check for the home mail server (thus, making it possible for an agent to run on more than one server)
- A local replica with an agent modifying the same documents as the server-based agent
- Different events triggering agents that modify the same document (for example, QueryOpen and QuerySave)

My agent queues are backing up (Notes/Domino 6 only)

Does it seem like your Web agents take longer than normal to run? Or perhaps the agent queue is backing up? Check to see whether or not the remote debugger is turned on. When you turn on remote debugging, each agent pauses before it begins executing. (This pause is one minute by default; you can set it longer or shorter.) If you forget that you turned on remote debugging, your agents may appear inexplicably slow. Currently the server log doesn't report when the remote debugger is pausing agents before execution; we plan to add this message in an upcoming maintenance release.

Deploying on different servers

If you create an agent you intend to deploy on multiple systems, you may find:

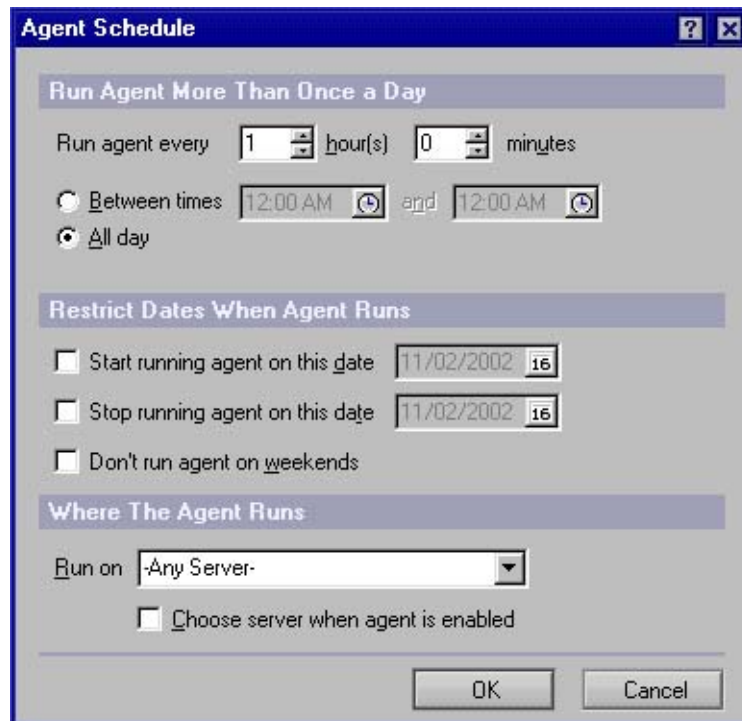
- The server names on which you need to deploy are different from the name of your server. In addition, you may not know the server names in advance.
- The agent developer's signature is different from the end user's signature.
- The development server does not have a certificate in common with the production server.

We will look at several approaches to solving the first two issues, which in turn, will make the certification issue irrelevant.

Changing the server name

When you create an agent, the name of the server on which the agent is supposed to run is stored in the agent. By default, the server name is the server on which you're developing the agent. For the agent to run on a different server, the server name needs to change. There are three ways to do this. The first approach is to deploy your agent as disabled. Click the Schedule button in the Agent Properties box and select the option "Choose server when agent is enabled." When the user first enables the agent, a prompt appears from which the user can choose a server name for running the agent.

The second approach is to specify that the agent can run on any server. To do this, you can click the Schedule button in the Agent Properties box, and select Any Server in the Run on field. Note that if you have replication set up between different servers and the agent modifies the same documents on those servers, you may end up with replication conflicts. The following screen illustrates both of these options.



The image shows the 'Agent Schedule' dialog box in Lotus Notes/Domino. It has a title bar with a question mark and a close button. The dialog is divided into three sections: 'Run Agent More Than Once a Day', 'Restrict Dates When Agent Runs', and 'Where The Agent Runs'. In the first section, 'Run agent every' is set to 1 hour and 0 minutes. The 'Between times' section shows 12:00 AM and 12:00 AM, with 'All day' selected. The second section has checkboxes for 'Start running agent on this date' (11/02/2002), 'Stop running agent on this date' (11/02/2002), and 'Don't run agent on weekends'. The third section has a dropdown menu set to 'Any Server-' and a checkbox for 'Choose server when agent is enabled'. At the bottom are 'OK' and 'Cancel' buttons.

The third approach is to write an agent that programmatically sets the server name of another agent. To do this, you can use the following code fragment:

```
agent.ServerName="ServerName"  
Call agent.save()
```

where *ServerName* is the new server name, which you can read from a database or obtain from the end user. You need to save the agent to update the server name.

Prior to Notes/Domino 6, agents could not manipulate and save other agents running on the server. An agent saving another agent was only allowed to run on the client. If the agent ran on the server and attempted to modify and then save the agent, the following error message was generated:

11/04/02 05:14:35 PM AMGr: Agent ('DoEnable' in 'test1.nsf') error message: Restricted operation on a server

This was because we did not have a way to preserve the user identity associated with the agent being changed on the server. (The user ID was not present on the server to re-sign the agent with that ID.) In Notes/Domino 6, it is now possible to do this, if you have the proper rights. An agent can change and save another agent if both 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 (a new field that has been added to the Programmability Restrictions section of the Server document's Security tab). In other words, users can change their own agents without any special rights, but need special authority to manipulate agents written by others.

Signing the agent

When you develop an agent, your signature is stored in the agent. You need the user's signature to replace yours in the agent, so the agent can run in the background with the user's rights. There are two ways to do this. Both are based on the fact that enabling an agent re-signs that agent with the signature of the person who enabled it. In addition, both approaches require that you deploy the agent as disabled. The first way relies on the end user to manually enable the agent by clicking on the enable checkbox.

The second approach is to enable the agents programmatically. This is more appropriate if you need to deploy a large number of agents. In that case, you may prefer to write an agent that enables other agents. The end user will run this agent, which then enables all the other agents. Here is the code that you need to include in your agent to programmatically enable other agents:

```
agent.IsEnabled = True  
Call agent.save()
```

You need to save the agent in order to update the IsEnabled property. As mentioned previously, prior to Notes/Domino 6 agents could not manipulate and save other agents running on the server. An agent saving another agent was only allowed to run on the client. This restriction has been removed in Notes/Domino 6, as long as the user has proper rights.

Conclusion

It was true four years ago, and it's still true today: Troubleshooting agents is an important topic to many Notes/Domino developers, administrators, and users. As we've shown, many of the tips and techniques used in R4.6 are still valuable. In addition, Notes/Domino 5 and 6 bring new issues to consider—as well as additional tools you can use to find out why your agent isn't doing what you want it to. We hope you find this article every bit as useful as the original when analyzing problems with your own agents—which hopefully you won't have to do *too* often!

ABOUT THE AUTHOR

Julie Kadashevich came to Iris in March of 1997 after working at FTP Software on Java and C++ mobile agent technology. 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. Outside of the Agent Manager, her two main interests are photography and quilting.



Notes/Domino 5 agent security at a glance

Figuring out agent security is not easy because there are many different situations you need to consider while troubleshooting a problem. The following tables should help you identify when the agent security is enforced and whose rights are used to authenticate the access level. Note that these tables apply to personal and shared agents.

Agent security consists of two parts:

- Agent restrictions control who can run the agent with what level of rights.
- Database ACLs control the level of access to the data the agent's effective user has.

Whether or not agent restrictions apply depends on how the agent is invoked. If invoked on the client, restrictions do not apply. If invoked on the server, agent restrictions do apply. Agent restrictions are always determined based on the agent signer:

	Client			Server	
How agent is invoked	User initiated	Scheduled	HTTP "run as Web user"	HTTP "run as signer"	Scheduled
Restrictions	N/A	N/A	Signer	Signer	Signer

Database ACLs always apply no matter how the agent is invoked. The identity 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 on to 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 R5 scheduled agents on the server and agents invoked from the Web running in Run as agent signer mode, the effective user is the agent signer:

	Client			Server	
How agent is invoked	User initiated	Scheduled	HTTP "run as Web user"	HTTP "run as signer"	Scheduled
ACL checks	Invoker	Invoker	Invoker	Signer	Signer

This table summarizes the rules for where R5 agents can access databases on other servers.

Type of agent/Where agent runs	Server	Workstation
Scheduled	Only via CORBA*	yes
User initiated	—	yes

*Accessing databases on other Domino servers is possible using CORBA-remoted Java back-end classes and DIIOP.



Notes/Domino 6 agent security at a glance

Figuring out agent security is not easy because there are many different situations you need to consider while troubleshooting a problem. The following tables should help you identify when the agent security is enforced and whose rights are used to authenticate the access level. Note that these tables apply to personal and shared agents.

Agent security consists of two parts:

- Agent restrictions control who can run the agent with what level of rights.
- Database ACLs control the level of access to the data the agent's effective user has.

Whether or not agent restrictions apply depends on how the agent is invoked. If invoked on the client, restrictions do not apply. If invoked on the server, agent restrictions do apply. Agent restrictions are always determined based on the agent signer:

	Client		Server		
How agent is invoked	User initiated	Scheduled	HTTP "run as Web user"	HTTP "run as signer"	Scheduled
Restrictions	N/A	N/A	Signer	Signer	Signer

Database ACLs always apply no matter how the agent is invoked. The identity 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 on to 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 Notes/Domino 6 scheduled agents on the server and Notes/Domino 6 agents invoked from the Web running in "Run as agent signer" mode, the effective user is the "Run on behalf of" identity if this field is populated (and the signer has proper rights). Otherwise the effective user is the agent signer:

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

This table summarizes the rules for where R5 agents can access databases on other servers:

Type of agent/Where agent runs	Server	Workstation
Scheduled	yes	yes
User initiated	—	yes