Run and RunOnServer: Adding Parameters

by Julie Kadashevich
(with Lakshmi Annavajhala)

Level: Advanced
Works with: Designer 5.0
Updated 12/01/99

Inside this article:

Related links:

[Out of the Inbox: New R5 mail agents]
[Demystifying the Out of Office agent]
[Troubleshooting agents]
[Minimizing delays in the Agent Manager]
[Controlling the agents in your system]

Get the PDF:

Everyone (well, almost everyone) knows that agents can call other agents. A LotusScript agent can call another LotusScript agent using the Run method. You can use the same method to invoke a Java agent, or to invoke a LotusScript agent from a Java agent. When using the Run method, the calling and the called agents will be executed on the same machine (either client or server). An agent running on the client can execute a server-based agent using the RunOnServer method. Again, it doesn't matter whether the agent is written in Java or LotusScript. Great, right? What else could you want?

Well, release 5.0.2 of Notes/Domino introduces the ability to add parameters to the Run and RunOnServer methods, so that you can pass information between client agents and server agents. In this article, you'll learn how these methods worked prior to 5.0.2, why the new features solve many problems, and see examples of agents using the new parameters.

The problem: Passing information

Suppose you wanted to pass some information between agents. If you were passing information on the same machine using the Run method, you could use NOTES.INI settings, environment variables, profile documents, or regular documents to store the information you wanted to share. If you were using the RunOnServer method, things get a little more problematic because different clients could invoke the same server-based agent.

You could come up with a scheme by which both the calling agents and the called agents know how to find the same document, and if you wanted to have unique information associated with each user, your scheme would have to key off the user name. But this is not the end of your worries, you also have to worry about the race conditions.

Prior to release 5.0.2 of Notes/Domino, in order to pass information to the agent invoked by the RunOnServer method, you could have used:

- A profile document based on the identity of an EffectiveUser of each calling agent.
- A statically-named regular document, which will be used by all users, where each of the user names would be used as a key.
- A shared document where each user can have his or her own data in the shared document. A separate field is created for each user, with the field name being based on the user's name. The agent then goes through all the fields on the document, uses the field value as parameters, and sets a new field value to pass back to the invoker. When the invoker gets back control, it retrieves the return value from the same field, and removes the field to mark the job as completed. You can add Authors and Readers fields to the created fields for security.

Each of these approaches potentially has some pitfalls depending on exactly what you are trying to accomplish and how your agent will be used. For example:

- If the authority used in the calling agent is different from the authority used in the called agent, a simple scheme for finding the proper profile document will not work.

- If you are operating in the multiuser dynamic environment where many users can call the same server-based agent, you have to be concerned about replication conflicts if you are using a statically-named document for passing information.
- If the database resides in two locations, the parameter documents could replicate at a wrong time, and contain partial updated data. If the parameter document replicates before the agent clears the value, the agent on the other server will find out-of-date parameter field data when it is invoked, and run incorrectly.

So until now RunOnServer was best suited for invoking standardized processes on demand instead of on a schedule, or in an environment where you don't need any context.

The solution: Parameters in Run and RunOnServer methods

Release 5.0.2 of Notes/Domino eliminates the problems described above by allowing you to add parameters to both the Run and RunOnServer methods. Both of these methods now accept an optional Note ID parameter. The document identified by the Note ID is accessible from both the calling and the called agents, and can be used to pass parameters between the agents.

The parameter document can be used for both input and output parameters. You must save the document used for parameter passing before it is accessed by another agent. You can delete this document as soon as it is no longer needed. Also, a new agent property called ParameterDocID has been added. This property allows you to retrieve the parameter document ID in the called agent.

The security framework of how Run and RunOnServer works has not changed in this release -- the agents executing on the client run with the authority of the invoker and the agents running on the server run with the authority of the agent signer. For complete information on agent security, see the **Agent security at a glance** sidebar of the *Iris Today* article **Troubleshooting agents.**

The RunOnServer method continues to execute the calling agent on the client and the called agent on the server. So if the calling agent is being invoked by a person other than the signer of the called agent, it is possible for the calling agents and the called agents to run under different authorities. Note that the server-based agent is not executed by the Agent Manager process, but rather in a separate thread created by the client on the server for execution of a given RunOnServer agent.

The Run method is valid both on the client (foreground and background) and on the server (in scheduled, event driven, and Web browser agents). The RunOnServer method is only valid in the agent running on the client (foreground and background).

In Release 5.0.2, if the RunOnServer method is invoked on the server, it will be mapped to the Run method. In prior releases this scenario generated a run time error "RunOnServer must be used with a remote database."

The following table summarizes where Run and RunOnServer are valid:

	Server	Workstation	HTTP	DIIOP
Back-ground	Run, RunOnServer mapped into Run	Run, RunOnServer	Run, RunOnServer mapped into Run	Run, RunOnServer mapped into Run
Fore-ground	n/a	Run, RunOnServer	n/a	n/a

Release 5.0.2 also includes improved error handling for the RunOnServer method. This method returns a status -- zero in the case of success and a non-zero value in the case of failure. In all cases a status of non-zero generates a run-time error. This error can be caught and processed with an on-error method. You no longer need to check the value of the returned status. The non-zero value contains an internal code that does not map to the defined error messages.

Now, let's look at some examples to illustrate how the enhanced methods work.

Example 1

In the following LotusScript example, the calling agent named "Top agent" will call the "Bottom agent" using the Run method. The two agents pass information in the field called "status."

Top agent

```
Sub Initialize
    Dim sess As New NotesSession
    Dim db As NotesDatabase
    Dim agent As NotesAgent
    Dim doc As NotesDocument
    Dim item As NotesItem
    Dim paramid As String

    Set db = sess.CurrentDatabase
    Set agent = db.GetAgent("bottom agent")

    ' Create document that will be used for parameter passing
    Set doc = db.CreateDocument

    ' Add the field that will contain the value
    Set item = doc.AppendItemValue("status", "none")

    ' Save the document
    Call doc.save(True, False)

    ' Set the parameter value
    paramid = doc.Noteid

    Call Agent.Run(paramid)

    ' Delete in-memory document, it does not contain up-to-date info
    Delete doc

    ' Retrieve the updated version of the document
    Set doc = db.GetDocumentById(paramid)

    ' Obtain the value set by the other agent
    Set item = doc.GetFirstItem("status")
```

```
        status = item.text
        Msgbox "status of bottom agent = " & status

        ' Remove the document used for parameter passing
        Call doc.remove(True)
End Sub
```

Bottom agent

```
Sub Initialize
        Dim sess As NotesSession
        Dim db As NotesDatabase
        Dim doc As NotesDocument
        Dim it As NotesItem
        Dim agent As NotesAgent
        Dim status As String

        Set sess = New NotesSession
        Set db = sess.CurrentDatabase
        Set agent = sess.CurrentAgent

        ' Obtain the parameter passed by the calling agent
        NoteId = agent.ParameterDocID

        '  Get the document containing the parameters
        Set doc = db.GetDocumentById(NoteId)

        '  Here you can have some processing
        '  if error is encountered the status could be set to the error code
        '  if processing is successful status is set to "done"
        Set it = doc.ReplaceItemValue("status"," done")

        ' Save the document so the top agent can see the updated value
        Call doc.save(True, False)
End Sub
```

Example 2

The following LotusScript example illustrates parameter passing using the RunOnServer method. In this method a client-based agent will invoke an agent running LSDO on the server.

Top agent

```
Sub Initialize
        Dim sess As New NotesSession
        Dim db As NotesDatabase
        Dim agent As NotesAgent
        Dim doc As NotesDocument
        Dim item As NotesItem
        Dim paramid As String

        Set db = sess.CurrentDatabase
        Set agent = db.GetAgent("oracle-connect")

        'Create parameter document
        Set doc = db.CreateDocument

        ' Populate the parameters
        Set item = doc.AppendItemValue("status", "none")
        Set item = doc.AppendItemValue("datasource", "ORACLE")
        Set item = doc.AppendItemValue("userid", "tigger")
        Set item = doc.AppendItemValue("passwd", "pooh")

        ' Save the parameter document
        Call doc.save(True, False)
```

```
        paramid = doc.Noteid

        Call Agent.RunOnServer(paramid)

        ' Delete in-memory document that does not have updated data
        Delete doc

        ' Get the updated document
        Set doc = db.GetDocumentById(paramid)

        ' Get the updated status
        Set item = doc.GetFirstItem("status")
        status = item.text
        Msgbox "return status = " & status

        ' Delete the document from the disk
        Call doc.remove(True)
End Sub
```

Bottom agent

```
Sub Initialize
        Dim con As New odbcconnection
        Dim sess As New NotesSession

        Dim db As NotesDatabase
        Dim agent As NotesAgent
        Dim doc As NotesDocument
        Dim item As NotesItem
        Dim sdatasource As String
        Dim suserid As String
        Dim spasswd As String
        Dim status As String

        ' Get the current agent and the parameter associated with it
        Set db = sess.CurrentDatabase
        db.delayupdates = False
        Set agent = sess.CurrentAgent
        NoteId = agent.ParameterDocID

        ' Get the parameter document
        Set doc = db.GetDocumentById(NoteId)

        ' Obtain the parameters
        Set item = doc.GetFirstItem("datasource")
        sdatasource = item.Text
        Set item = doc.GetFirstItem("userid")
        suserid = item.Text
        Set item = doc.GetFirstItem("passwd")
        spasswd = item.Text

        Set item = doc.ReplaceItemValue("status", "before connect")
        status = item.text

        con.SilentMode = True
        If Not con.connectto( sdatasource, suserid, spasswd) Then
                nErrNum = con.GetError
                        strErrMsg = con.GetExtendedErrorMessage(nErrNum)
                        Call doc.ReplaceItemValue("status", "Connect Failed")
                        Goto Terminate
                Else
                        Messagebox "Connect was successful"
                        Call doc.ReplaceItemValue("status", "Connect Succeeded")
                End If
```

```
        ' Save the parameter doc
        Call doc.save(True, False)

        Delete doc
        con.disconnect

Terminate:
    Msgbox "terminating ..."
End Sub
```

Example 3

The following Java examples illustrate parameter passing using agent.runOnServer(NoteID).

JavaAgentROS

```
import lotus.domino.*;
import java.util.*;

public class JavaAgent extends AgentBase {

public void NotesMain() {

    try {
        Session session = getSession();
        AgentContext agentContext = session.getAgentContext();
        Database db = agentContext.getCurrentDatabase();
        Agent ag1 = agentContext.getCurrentAgent();
        Agent ag2 = db.getAgent("JavaAgentGetParameterDocID");
        Document doc = db.createDocument();
        // Document item "AgentList" will collect the names of
        agents called
        doc.replaceItemValue("AgentList", ag1.getName() + "
        performing agent.run(NoteID)");
        doc.save(true,true);
        String paramid = doc.getNoteID();
        ag2.runOnServer(paramid);
        // remove old doc object from db cache
        doc.recycle();
        // get updated document
        Document doc2 = db.getDocumentByID(paramid);
        Vector v = doc2.getItemValue("AgentList");
        String sAgList = (String)v.elementAt(0);
        System.out.println("Agent calling sequence: " + sAgList );
        // cleanup
        doc2.remove(true);
        } catch(Exception e) {
        e.printStackTrace();
    }
  }
}
```

JavaAgentGetParameterDocID

```
import lotus.domino.*;
import java.util.*;

public class JavaAgent extends AgentBase {

    public void NotesMain() {

    try {
        Session session = getSession();
        AgentContext agentContext = session.getAgentContext();
        Database db = agentContext.getCurrentDatabase();
```

```
            Agent ag1 = agentContext.getCurrentAgent();
            String paramid =  ag1.getParameterDocID();
            Document doc = db.getDocumentByID(paramid);
            Vector v = doc.getItemValue("AgentList");
            String sAgList = (String)v.elementAt(0);
            doc.replaceItemValue("AgentList", sAgList + " > " +
            ag1.getName());
            doc.save(true,true);
            } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Conclusion

By enhancing the Run and RunOnServer methods and introducing the ParameterDocID property in the Agent class, you can now pass information between agents.

Adding new features in a QMR (Quarterly Maintenance Release) is unusual, but we felt that this was one of those special cases where the rules had to broken to address customer needs. We hope this article helped you take advantage of the new features.

ABOUT JULIE

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 *Iris Today*. She has been focusing 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. Outside of work, her new hobby is kayaking. She recently bought her own kayak and is working on taming it.

ABOUT LAKSHMI

Lakshmi Annavajhala has been a User Assistance Writer at Lotus for the past two and a half years. She is part of the Programming documentation team for Domino Designer. This includes the conceptual and reference material for LotusScript objects, Formula Language, and XML for Domino.