Notes.net

Iris Today

Mallareddy Karra on the Domino Console

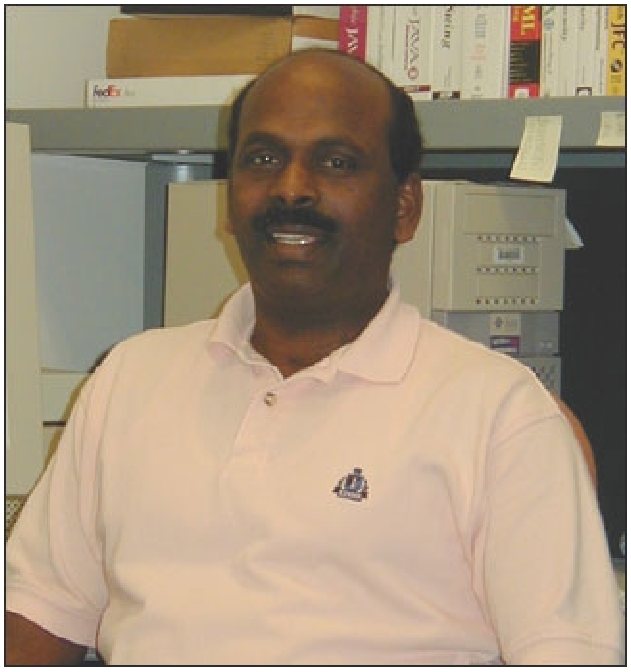
by
Laura
Rutherford

Level: All
Works with: Domino Rnext
Updated: 12/04/2001

Giving administrators the gift of flexibility when it comes to viewing or monitoring servers is what the Domino Console is all about. Here, we talk to software consultant Mallareddy Karra about the benefits of the Console, what prompted its development, and why its creators decided to develop it in Java. He also talks about some of the particular challenges faced in developing a console that needed to mimic some of the functionality and look-and-feel of other Domino console tools, yet also contain some of its own unique benefits.

The Domino Console is new in Rnext. What is it and what is its purpose?
The Domino Console is a Java-based Graphical User Interface (GUI) used to manage the Domino server console from anywhere, providing the same functionality on all platforms with a user-selectable look-and-feel. The purpose of the Domino Console is to give administrators the ability to view or monitor the server console from anywhere, and from any platform. Administrators can also simultaneously monitor different servers that are in the same domain or in different domains. Another purpose is to give administrators the ability to connect or disconnect from server consoles as and when needed.

What prompted you to develop it?
Originally, I was assigned to fix a problem with the existing server console on UNIX platforms. The problem was caused by the fact that the same terminal was used for command input and server output messages that work asynchronously. As a user types commands for the server console, the user also got messages from the server—so the messages and commands got intermixed. That's when we said "Why don't we have a consistent console look on all the platforms, where the functionality is the same so that the user does not have to think differently for each platform?" I proposed having a console that is GUI-based and that is actually cross-platform, with the same functionality—at least minimally—as the other consoles.



Can you tell us how the Domino Console works?
To meet one of the primary goals (that is, remote access to the server console), we developed the Domino Console using client/server architecture. By using the client/server model, we could separate the presentation (the GUI) from the actual data. This model also enables multiple connections to a single server from different users or a single user connecting to multiple servers.

The architecture has three modules. One of them is the existing Domino server and another is the Controller, which controls the Domino server. The Controller is a very small piece of the code that resides on the server platform along with the server. The last piece is the Domino Console, a GUI-based program, which could be running on the same server machine or it could be running remotely on any platform.

The Domino server runs under the control of the Controller on a server machine. The administrator runs Domino Console and connects to the Controller running on the same machine or a remote machine. There are two levels of authentication we use to achieve the highest level of security. The connection between the Domino Console and the Controller is done through a secure connection—we use an SSL connection for that. Apart from that level of security, which does data encryption, we also have a user authentication mechanism to access the server console. So, the first level of security performs transfer of the data in a secure fashion so nobody can see the details. The second level of security is the ability to access the Domino Console based on user authentication with a user name and password.

You can start any of the three modules in any fashion. For example, you can start the Controller by itself or you can start the Controller and Domino, and so forth. This feature gives administrators the ability to monitor remotely. The advantage of doing this, especially in big companies where they have lots of machines running, is that they can start the Controller and Domino without a Domino Console on the server machines and run the Console on an administrator's desktop—thus eliminating the need for monitors for each server while still providing accessibility from the administrator's desktop.

How does the Domino Console differ from R5 Admin options—in

particular, the Admin client and the Web Admin client?
The Domino Console is not a one hundred percent admin tool. The Domino Console is more like a server console that allows minimal administration related to the console, such as starting and stopping the server.

The other difference is because the Admin client basically works with the Notes RPC mechanism and the Web Admin client works through the HTTP server on the Domino server, these two tools cannot work when the Domino server is not responding or is hung or crashed. When it comes to the Domino Console, that is not the situation. Even when the server is down, the Controller is up and running because it is a separate piece. So you can talk to the Controller and find out what went wrong or tell the Controller to kill and/or restart Domino. Or you could do some minimal system administration things using the Controller's native shell command access.

Also, the Domino Console can connect or "talk" to multiple servers at same time. That gives an administrator the ability to, say, bring down all the servers at the same time. Instead of an administrator having to physically go to each machine or connect to each machine and bring it down, the administrator is already connected to all of them. Similarly if an administrator wants to broadcast a message to the people connected to all the servers, he or she can broadcast a message to all these users simultaneously.

And normally when you connect to the Admin client, you need a Notes ID to bring up the Admin client and also to connect to the server. If you want to connect to a different server in a different domain, you may need a different ID if you are not an administrator. But sitting at one Domino Console, you can connect to different servers on different domains with different user names. The Web Admin client works the same way—conceptually you can connect to only one server at a time. Whereas with the Domino Console, since you can connect to multiple servers, you can connect to them as different users. So the user doesn't have to carry multiple IDs or anything like that.

In terms of similarities, when we designed this product we tried to minimize differences in the functionality and the look-and-feel of the console from the Admin client's remote console, the Web Admin client's remote console, and the Domino Console. We did this in hopes of eliminating the confusion and learning time for administrators.

In what scenarios would an administrator use the Domino Console as opposed to the Admin client or Web Admin client?
As explained, pretty much what you can do with Domino Console you can do with the other clients' remote consoles as well. But there are restrictions; for instance, the Admin client is available only on a Windows platform. And if you are to use the Web Admin client, then the server should be up and running. If the server is down or not responding, you cannot use it. So those are the major advantages to using the Domino Console. You can use it on any platform and you can use it even if the servers are not running.

What are some of its other benefits?
The flexibility of it. The Domino Console can work on any machine and can work when you don't have access to servers. You have the native command access and the ability to start and stop the servers even when the servers are not in a reachable state. The Domino Console has the ability to enable or disable a user for the length of the server session. This is a great advantage for the customers who want to let external administrators or developers look at their server remotely. And yet the customer has the capability to disable any user at any time if they find the person is not doing the right thing in the interest of the server.

What were your goals when designing the Domino Console?
We wanted to eliminate the problem I mentioned earlier on UNIX servers and, at the same time, give more flexibility and consistency across all platforms,

provide remote connection capability, and display the data in an easy and manageable way. When designing the GUI, we wanted it to have a user-definable look-and-feel while normalizing the functionality to make sure that administrators have no confusion between the Domino Console and the Admin client remote console or the Web Admin client console.

The next big design goal was to make sure the Controller is up and running even when the Domino servers are down—thus giving the ability to administer Domino to stop and/or start. And the ability for administrators to access the console from anywhere was another big goal.

Can you tell us why you decided to develop the Domino Console in Java?
Java is a platform where you have a lot of advantages. We decided to develop in Java to achieve the primary goals we had for the console—for example, making the console available across all platforms. If we developed this product in other languages like C/C++ and native Windowing platforms, then we would have to write a lot of platform-dependant code. This process would have required a lot of code and caused a lot of problems. Java is one-time coding. You sit on one platform, write on one platform, and it is available on all the platforms.

Java also has the ability to switch its look-and-feel. For example, when you are in Windows, it can show you the Windows style. When you are in UNIX, it can show you the UNIX style. Java solved all the big problems we may have faced in trying to develop this product for multiple platforms.

Did you face any particular challenges in developing the Domino Console?
We wanted the console to be independent in the sense that we did not want to depend on the Domino code—so that the Console could be still up and running even if Domino was down. Because each platform works differently, we had some difficulty in achieving this goal. Another big challenge was making the Console fully internationalized. Java helped in some ways in this regard.

Can you share some of the customer feedback on this new feature?
We've shown the Domino Console to some customers, especially the ASP kind of customers. They like this idea because typically they have hundreds of machines, and in order to bring each one down, they have to go to each machine and do the operations. With the Domino Console, they can sit in their office and connect to any machine in the lab whenever they need to and do the operations from their office. That’s a huge advantage for them both financially and, because it eliminates the need for monitors, space-wise.

In a scenario where you have a big company that has multiple offices and locations, right now they need an administrator to be here and there and in several locations. With the Domino Console, they can do things remotely, and they won’t need to have multiple administrators in multiple locations.

Do you feel the Domino Console is going to be difficult for administrators to learn?
No. It's very easy. We tried to normalize the feature set on the three consoles so that knowing one is pretty much like knowing all of them.

What is on the horizon for the Domino Console beyond Rnext?
We are looking into allowing administrators to do remote debugging. For example, let’s say Domino is down. Right now someone would have to physically go there and run the debugger and do the debugging and find out the reason the server is down. We want to give the ability to do the debugging from a remote location.

We also want to give the ability to connect through firewalls in a secure way. And we will also be adding some ways to check the health of the server, such as how well a server is responding to the clients' connections, how much resources the server consumes, and the ability to alert administrators about the changes in the critical parameters set by users.

About Mallareddy Karra
Mallareddy is a consultant in the Core Services Team at Iris. He started at Iris in 1991 and was one of the first to do R3 migration to UNIX. Mallareddy has been working since then on R4, R5, and Rnext. Prior to coming to Iris, he consulted with CLSI (working on Library Automation software), Computer Vision (working on CADDS5), Sun Microsystems, and Fidelity Investments (developing Fixed Income Automated Trading System). In 1994, he co-founded a consulting company, Software Experts, Inc. Mallareddy's hobbies are outdoor games such as tennis, cricket, and volleyball and movies. Outside of work, he juggles his time between his company, family (his wife and two lovely daughters), and social activities.