The Complexity of Software Update Services and What it means for AIMS

The Software Update Services (SUS) Server that is used in DPEND LAN for automated operating system updating, is functionally complex software that holds many leads for our Assets and Infrastructure Management System (AIMS) software.

In addition, this is an excellent example of how Network Computing & Administration is closely related to the modern concepts of Database conceptualization, development, deployment and administration.

In order to appreciate this relationship, we must first understand how SUS functions in a Windows 2000 / 2003 Active Directory Domain, and then relate its functionality with what AIMS does and can do.

How does the SUS operate?

Let us see how SUS operates in our DPEND Local Area Network (LAN):

DPEND LAN has a number of Windows 2000 machines. Most of them have Service Pack 4 installed.

However, between the release of two Service Packs, Microsoft releases a number of patches for the operating system, some of them critical. These patches are small programs that update specific sub-sections of the Windows 2000 operating system. Examples of such sub-sections include the Internet Explorer, the Windows Media Player, the Outlook Express, MS Office and so on.

Each of the Windows 2000 clients will have slightly different configurations, depending upon the software installed in it for the User. As User requirements vary, the installed software too will be non-uniform. In addition, it is quite possible that (at least initially) each of the Windows 2000 clients are not upgraded to the same level as every other instance of a given sub-section of the software. For example, some may be using Outlook Express 6.0, others Outlook Express 6.0 with Service Pack 1, and so on. Some clients may be running under Service Pack 3 – others may be using Service Pack 4.

So, the challenges of bringing each client to the same level are:

- Identifying the machine's current list of software / sub-sections
- Identifying if they are up to date
- Identifying if there are any updates available
- Downloading those updates from the Internet
- Checking and approving the downloaded updates
- Applying those updates – in fact, applying only those differential updates to each client so that all come up to the same level
- Preventing unauthorized Users from other Divisions from applying updates using our Servers
- Applying those updates, asynchronously, so that all the clients do not hog the bandwidth at the same time

- Applying the updates with a special account having elevated privileges so that the Clients are updated even if normal Users are using the machine
- Applying Server specific updates to the Servers
- Applying the updates for the Server and the Clients at different times so that when a Client is ready to enter the LAN, the Server is not down
- Applying the updates in such a fashion that, if required, the updates can be uninstalled
- Logging all the update information so that the User / Administrator knows which update has been applied where

and so on.

The SUS Server installed in DPEND LAN meets these challenges through the following steps (*these are in-built in SUS; we have not created these; we just configure the SUS correctly*):

- The SUS Server is installed in DPENDFOUR, the fastest Server in DPEND
- Whenever Microsoft releases updates (which is known through emails from the MS SUS Department), we connect DPENDFOUR to Microsoft SUS Server and "Synchronize" the two Servers – at all other times, DPENDFOUR is kept disconnected from the Internet for security reasons
- During the process of "synchronizing", if it is the first synchronization, DPENDFOUR downloads ***all*** the updates from the MS SUS Server; on all subsequent synchronization, only the ***differential*** updates are downloaded
- The downloaded updates are then approved on a case-by-case basis
- These operations (synchronizing, downloading, approving) take place through active server pages within the .NET framework (ASP.NET) as the front-end with Internet Information Server (IIS) as the middleware; the downloaded updates form the database. So the connection is like this: ASP.NET ➔ IIS ➔ Downloaded Updates Database
- In our LAN the IIS and the downloaded updates reside in the same Server, DPENDFOUR. ASP.NET can be used from either DPENDFOUR itself, or from any other Client. Note that even though the ASP.NET processes, IIS and the Updates Database reside in the machine, they are different entities and will require appropriate authentication and authorization mechanisms to work together.
- In this case, ASP.NET pages make anonymous access calls to the IIS, which in turn invokes the IUSR_<servername> and IWAM_<servername> accounts to access the downloaded databases
- The Domain Controller (DC) of the DPEND LAN – in our case, it is DPENDTHREE which runs Microsoft Windows 2003 Enterprise Server – has been programmed through Group Policy, to instruct each Client in DPEND LAN to (a) allow updates to be performed, (b) allow updates at 1300 hours, (c) to download the relevant updates from DPENDFOUR, (d) allow updates without any User / Administrator intervention, and (e) allow the Clients to restart themselves automatically.
- At random times during the day (or whenever the Client is ON and connected to the SUS Server in DPENDFOUR), each client calls SUS Server and finds out if there are any updates that needs to be applied. Basically, the Client makes an anonymous access contact to the IIS, which in turn invokes

the IUSR_<servername> and IWAM_<servername> accounts to access the downloaded databases.

- If there is any patch that is new or updated, that is approved and ready for application, that patch / patches are downloaded from DPENDFOUR to the Client. The Client downloads only those patches that are appropriate for itself, from the list of approved patches.
- At a preset time, (set through Group Policy, in the DC, i.e., DPENDTHREE), all the Clients apply their respective patches, and restart themselves as required.
- The other Servers and DPENDFOUR update themselves too, in the same fashion

Note that in this procedure there are THREE types of machines and THREE types of processes involved. The THREE types of machines are (1) the domain controller which has the Group Policy for the whole domain, (2) the SUS Server that has the downloaded updates, and (3) the clients. The THREE processes are (1) the ASP.NET which acts as a front end web page for various operations, (2) the IIS that acts as a middleware to supply the updates, and (3) the downloaded update database itself.

How does this relate to AIMS?

For a quick understanding of the relationship between SUS model and AIMS, one can just associate the SUS' downloaded updates, with the AIMS' various databases [such as Assets (assets.dat), Systems (systems.dat), Components (components.dat), Parts (parts.dat), etc.]. The IIS and ASP.NET roles are common. But that's not all; there's a much deeper connection between SUS and AIMS from the viewpoints of (a) authentication and authorization, (b) distributed, asynchronous updating / editing of the AIMS' databases, (c) distributed, secure access, and (d) distributed decision-making.

A Possible AIMS Scenario

Completely distributed AIMS Software, could have the following structure:

- Each of the AIMS Software databases is located in a different Server (i.e., assets.dat is present in one server, systems.dat in the second, and so on); and each of these Servers are geographically separated, connected across the Internet using a Virtual Private Network (VPN). Note that this is an *extreme* case – a more *probable* case could be where each of the AIMS Software databases are loaded in different Servers, located within an Intranet. However, the proposed solution would cover both the extreme and the probable cases.
- The databases are present in a Relational Database Management Server (RDBMS) such as SQL 2000 Server, which is accessed using IIS, present in some other physical server. Note that, if an RDBMS is used, each of the AIMS database is a Table.
- The AIMS Software databases are accessed for reading, editing and updating through ASP.NET from clients that might vary from simple browsers to mobile devices.

Now, in such a scenario, the common threads between SUS and truly distributed AIMS Software emerge more clearly.

For example, in the case of a browser access to the AIMS database, the following steps would take place:

- The browser contacts the IIS, at the designated Server, and tries to access, say, aims.aspx (ASP.NET environment). Depending upon where the IIS is located with respect to the browser (within the Intranet, Extranet or Internet), the communication channel between the browser and the IIS must be secured.
- This contact must be authenticated by the IIS, depending upon how it (IIS) is configured (Windows Integrated authentication, anonymous access, etc.)
- If it is Windows Integrated authentication, the accessing User must have the permission to enter the IIS with his credentials and to access aims.aspx with his credentials (these two access controls could be differently set).
- If it is anonymous access, then the built-in anonymous User accounts IUSR_<servername> and IWAM_<servername> must have permission to access the IIS and the file, aims.aspx (including NTFS permissions)
- Assuming the browser User has the required permissions and accesses the aims.aspx page, the User must then have permission to execute the active code present in the aims.aspx page
- That guaranteed, the code in the aims.aspx page will access the various Tables in the SQL Server 2000 – each Table, as we saw earlier, is a database in this distributed mode of AIMS software. Depending upon the type of authentication that is set in the IIS, either the browser User's actual identity, or the anonymous access builtin accounts namely IUSR_<servername> and IWAM_<servername> must have the necessary permission to access the SQL Server 2000 Tables. Again, the mutual locations of the IIS and the SQL Server and their communication channel (communication within the Intranet, Extranet or Internet) will come into play here.
- The database access can further be fine tuned further, by providing access controls at the Table or Row/Column level (by defining the roles of the accessing identities) or in a broader sense, for reading / writing, etc.
- Once the Tables are accessed, the information must flow back to the browser in the same way, without compromising either data quality or security.
- At every interface (browser-IIS, IIS-SQLServer, SQLServer-Database), the transactions must be logged so as to allow accountability reviews later.

From this account, one can see the confluence of technologies between the SUS mode of operation and that of distributed AIMS Software. In a way, the AIMS case is even more challenging, as SUS uses anonymous access throughout, thereby resolving a number of issues easily (e.g., SUS need not flow the identity of the "caller" down to the database level).

This is also an exercise that highlights the commonalities between network computing, administration and management on the one hand, and the development of a modern, distributed database access for device-agnostic access on the other.