A guided tour of the SAP NetWeaver Application Server upgrade tools

by Bert Vanstechelman, Mark Mergaerts, and Dirk Matthys

Bert Vanstechelman
Senior SAP Basis Consultant,
Logos Consulting

Mark Mergaerts
Principal Technology
Consultant, SAP Belgium

Dirk Matthys
Senior ABAP Developer,
Bekaert Group

(full bios appear on page 38)

This article is an excerpt from *SAP NetWeaver Application Server Upgrade Guide*, by Bert Vanstechelman, Mark Mergaerts, and Dirk Matthys, available from SAP PRESS. See the sidebar on page 38 for additional details and ordering information.

When dealing with a complex process such as an SAP upgrade — especially when things go awry — knowledge is power. That knowledge begins with becoming familiar with the upgrade programs and user interfaces. Being comfortable with these will greatly improve your effectiveness; you will do the right thing at the right moment, and you will be able to correctly identify and solve problems if and when they arise.

Compared with earlier versions, the current generation of SAP upgrade tools is either brand-new or at least comes equipped with a new look and feel. However, the tools are built upon operational concepts and methods of user interaction that have been around since early versions of SAP R/3 and have had plenty of time to stabilize and mature. The tools have been thoroughly tested and tweaked, and when the going gets tough, they will serve you reliably. They won't tie themselves into knots, spew gibberish at you (or go incommunicado), or vanish into thin air altogether.

This article introduces you to the SAP upgrade tools. You will learn how they work, where to find them, how to install them and get them going, and how to use their management and monitoring features so that you stay in control of the upgrade and know at all times what is happening.

The information presented here lays a common foundation for every upgrade. By "common," we mean those functions, mostly in the area of monitoring and managing the upgrade, that you can use in any upgrade scenario regardless of the component and release you are upgrading. Viewing the upgrade logs, changing the passwords of upgrade users,

or asking the upgrade process to trigger an alert when it stops are useful functions in every upgrade.

> ***Note!***
>
> This article applies to all SAP release upgrades to SAP NetWeaver 2004s or Application Server 7.00 ABAP and Java.

Introducing the tools

Since Basis release 6.20, SAP delivers and supports both the SAP NetWeaver Application Server ABAP and the SAP NetWeaver Application Server Java.

Whichever component you upgrade, if it is based on the SAP NetWeaver 2004s platform, you will always use the same upgrade programs:

- SAPup for the upgrade of ABAP
- SAPJup for the upgrade of Java

SAPup is the successor to the R3up program, which was used in all previous versions of SAP. SAPJup is new because SAP NetWeaver 2004s with AS 7.00 is the first release that provides an upgrade procedure for the AS Java.

To control the upgrade process and to interface with the user running the upgrade, the ABAP and Java components also provide applications of their own:

- The Upgrade Assistant Server and Upgrade Assistant GUI for the ABAP upgrade
- The SDT Server and SDT GUI for the Java upgrade (SDT stands for Software Delivery Tools)

If the system to be upgraded only runs one stack, then you only use the upgrade tools for that stack. For a pure ABAP system, you use the ABAP tools SAPup and the Upgrade Assistant. For a pure Java system, you use the Java tools SAPJup and the SDT Server and GUI. With double-stack systems (ABAP + Java), you use the two combined.

Synchronized upgrades

If the system runs both an ABAP and a Java stack, then you will need both sets of upgrade tools. The tools are designed in such a way that each knows of the other's existence; more than that, at several points during the upgrade, they will synchronize their activity, allowing the other upgrade to catch up or waiting until the other side has completed a critical task. For example, just before the ABAP upgrade hits the point where it shuts down the SAP central instance, marking the beginning of downtime, it will verify how far the concurrently running Java upgrade has progressed. If SAPJup has not yet reached its beginning of downtime, then ABAP will wait. The inverse is also true.

By running in a synchronized manner, the ABAP and Java upgrades can be executed simultaneously without the risk of conflicts (e.g., one side stopping the SAP system at a time when the other side is working in it). There is one precondition, however: you must always start the ABAP upgrade (SAPup) first. The ABAP side determines that the SAP system is double-stack and takes the necessary measures to make sure that the ABAP and Java upgrades will be able to run together.

The requirement to start the ABAP upgrade first has little or no effect on the overall time needed to upgrade the system. Unless it runs into some serious trouble, the runtime of the Java upgrade is much shorter than that of the ABAP upgrade. This is true both for the uptime part of the upgrade (especially if you artificially slow down the ABAP import phases) and for the downtime part (where the Java side mostly does software deployment, whereas the ABAP side also has to deal with things like table conversions and data conversion programs).

Upgrade program and control program

The general architecture of the ABAP and Java upgrade tools is very similar. Both are made up of the

actual upgrade program (SAPup, SAPJup) and a control program (Upgrade Assistant, SDT). The upgrade program drives the actual upgrade activities, such as data imports, structure conversions of objects in the database, Java deployment and ABAP transports, and so on. The control program does not do any technical upgrade work. Its main responsibility is to handle communication with the users (who are connected via their GUI) and to control and monitor the upgrade processes. Via the control program, you will instruct the upgrade program to start (and sometimes to stop). If errors occur, you will be notified of these through the control program; after investigating and fixing the error, you will, once more via the control program, order the upgrade process to retry the failed step.

The upgrade programs: SAPup and SAPJup

SAPup (for ABAP) and SAPJup (for Java) are responsible for running the entire upgrade process from start to — hopefully successful — finish. However, they do not do any of the dirty work themselves; this they leave to the appropriate utility, such as tp (to import transport requests into the AS ABAP) or JSPM (Java Support Package Manager, to deploy software on the AS Java). The main task of the upgrade programs is to start these worker utilities when needed, to monitor them and examine their result, to interrupt or slow down the upgrade when needed, to obtain input and instructions (not directly but via the control program) from the user who administers the upgrade, and to report the progress of the upgrade and alert the user in case of problems (again through the control program).

An upgrade is really a serial process made up of a long series of activity steps or phases. The total number of phases depends on the SAP release and the nature of the upgraded component, but in an ABAP upgrade, there are well over 200 phases and in a Java upgrade over 150. Each phase in the upgrade deals with one specific task. What the phase has to do can range from the simple and almost trivial, such as checking or modifying a flag in a database table, to the very complex and elaborate, such as activating the new dictionary or deploying the new versions of business applications. With such vast differences in the amount of work a phase must do, it is easy to understand that the runtime of the upgrade phases will also vary widely. Many phases will take just fractions of a second (which does not make them any less important!), but others may run for several hours. One group of phases, the ABAP database import, can even be slowed down artificially to minimize its impact on the server load (more on this later).

The fact that the upgrade process is serial in nature does not mean that it is also single-threaded. Some heavy-duty phases, such as dictionary activation, table conversion, or the huge data imports and data conversion runs during downtime, use parallel processing to benefit as much as possible from the capacity of the server and thus reduce the upgrade time.

PREPARE versus upgrade

Both the ABAP and Java upgrade processes are divided into two major parts: one is called PREPARE, and the other is the actual upgrade. Both the PREPARE and the upgrade are subdivided into phases, as explained previously.

PREPARE

As you can guess from the name, PREPARE deals with all the preparatory activities in the system to be upgraded. The first thing PREPARE will do is prompt you for the parameters of the upgrade, such as the location of the upgrade media, host names, passwords, and so on. These parameters will be used throughout the upgrade, and you will not have to enter them again (not that anything you enter is irreversible; you have the opportunity to change parameters later on if necessary).

After asking for the upgrade parameters, PREPARE copies data and programs into the upgrade directory, imports the upgrade tools into the database, and installs the shadow instance (for ABAP). Add-ons, support packages, and languages are integrated into the upgrade. After the initial configuration of the upgrade, PREPARE verifies that the source system meets the requirements of the upgrade process and of

the target release. This produces an action list, which must be taken care of before the actual upgrade can start. If the first pass of PREPARE detects errors (i.e., preconditions for the upgrade that are not met), then you must repeat the corresponding parts of PREPARE until no more problems are reported.

PREPARE is executed with the system up and running. End users are not affected. You normally run PREPARE several days before the actual upgrade. Runtimes vary, but for a "typical" ABAP PREPARE, you should expect a runtime between four and eight hours, mostly depending on factors such as the number of support packages to bind into the upgrade, the number of languages, and quite a few more.

Java PREPARE also runs during uptime. Like ABAP PREPARE, its first action is to prompt for the upgrade parameters. Its other tasks are also similar to those of ABAP PREPARE, but it normally has less work to do and therefore has a shorter runtime (typically between 1.5 and 3 hours, but again this is just an estimate based on our own upgrade experience).

Upgrade (ABAP)

When PREPARE has done its work and is satisfied with the condition of the system, you can start the actual upgrade. Way back yonder, this was the moment when you asked users of the SAP system to kindly log off and find some other occupation for the next few days. Today, that will happen only if you unwisely decided to run the upgrade in resource-minimized mode.

If you opted for the downtime-minimized method — as you ought to do — then the first (and longer) part of the upgrade happens while the SAP system is still in normal use. During this time, the upgrade performs all the necessary actions to build the shadow repository containing not only the new release but also a copy of all your own development. Early on in the process, the upgrade imports the data from the Upgrade Export media into the database. Because of the performance impact this data import may have and because of the number of database logs it will generate, you have the option of artificially slowing down the import. In this way, the upgrade takes as few resources as possible away from the production system. A slowed-down import also enables the backup procedures to keep up with the volume of database logs.

Not very long after the database import finishes, the upgrade starts up the shadow instance. After some preparatory work to make this instance fully usable, the upgrade will stop and ask you to carry out the dictionary modification adjustment using Transaction SPDD. When this is finished, the activation of the new dictionary begins (still in the shadow instance and shadow repository). After the activation and some other phases, the upgrade shuts down the shadow instance, which will not be needed again. Next, the main upgrade transports and the support packages that were bound into the upgrade are imported into the shadow repository. All the while, the main system remains productive.

Finally, with the shadow repository fully built up, the upgrade will inform you that all its uptime processing is done, and that it now waits for your permission to shut down the system and enter downtime. With correct planning and without accidents, this should happen some time before the start of the planned downtime window. For instance, the upgrade might reach the downtime point on Thursday afternoon while planned downtime begins at 7 p.m. Friday evening. In that case, you simply leave the upgrade waiting.

After the downtime window begins, you inform the upgrade via the upgrade GUI that it may stop the SAP system. Between then and a time quite close to the end of the upgrade, the upgrade process takes full control of the system, stopping and starting it as and when required. During this downtime phase, the system is placed in "upgrade lock" mode, which means that you can only log on as SAP* or DDIC. Except for problem solving, there is normally no need to log on to the system (and you are bound to be kicked out without warning anyway when the upgrade reaches a phase in which it has to stop the SAP system).

Near the end of its run, the upgrade process unlocks the system, which by now is fully on the new version, and restarts it for the last time. The upgrade

has only a few more phases to run (although some of these, e.g., the variant restore, can still be quite lengthy), but during this time, you can already log on to the system and start on the postprocessing tasks. After some final interaction, the upgrade kindly informs you that the upgrade is complete (somewhat optimistically because there is still a load of post-upgrade work waiting), produces a timing document and an evaluation form, and then stops.

Upgrade (Java)

The Java upgrade process (SAPJup) is on the whole a simpler affair than its ABAP counterpart and is likely to take less time. By far the most important task for the Java upgrade is the deployment of the new release on the AS Java.

After you start SAPJup, the process will soon stop to announce the beginning of downtime. Once you confirm this, SAPJup takes control of the AS Java, stopping and restarting it whenever necessary.

When it reaches the end of downtime, the upgrade will again stop, prompt you to back up the database and Java upgrade directory, and restart the AS Java. After producing an evaluation form, the upgrade process ends.

The Java upgrade does not use a shadow instance.

Synchronized Upgrade (ABAP and Java)

With a double-stack system, you run two PREPAREs and also two upgrade programs. The ABAP upgrade must always be started first, before the Java upgrade. Each side is aware of the other's existence. During much of the time, the two upgrades work independently, each on its own stack, but at some critical points, it is necessary that they synchronize with each other. For example, the ABAP upgrade will not enter downtime unless the Java upgrade is ready to do so as well (and vice versa). The one who reaches the downtime point first will wait for the other to catch up. Another example is that in a double-stack upgrade, the ABAP side is responsible for installing the new kernel. If the Java upgrade reaches the point where it needs the new kernel and sees that the ABAP side still has to carry out the kernel switch, it will wait for ABAP.

The control programs: Upgrade Assistant and SDT Server/GUI

Both control programs are designed as two-tier client-server applications, with a server program running on the upgrade (SAP central instance) host, and a GUI running on the user's workstation.

Despite this similarity in design, the ABAP and Java upgrade control programs are not completely identical. Their look-and-feel is not quite the same, nor are they used in exactly the same manner. In part, this goes back to their origins. The ancestor of the current ABAP upgrade tools was a program called R3up, which back in the old days was used for upgrades to SAP R/3 releases 2.x and 3.x. The old R3up was both upgrade and control program. It had no GUI client; you simply started it from the OS command line and interacted with it via this same OS session window. In a less than perfect world, this had obvious drawbacks: Even the briefest network failure between server and PC could kill the active upgrade process, not to mention a hanging PC, a Windows error, or user mistakes such as a badly aimed mouse click or Ctrl+C.

With the advent of release 4.0, SAP redesigned the upgrade program, separating the server activity (upgrading) from the user interaction (monitoring, parameter input, starting and stopping). The former was still handled by R3up[1], which had now become a strictly server-based program running in the background and thus invulnerable to whatever went wrong at the frontend. The interaction and communication part was moved to a new Upgrade Assistant (UA), which itself consisted of a server process running on the upgrade host — and like R3up shielded against frontend trouble — and a GUI, which could be used to connect to and disconnect from the upgrade at will.

[1] The program continued to be called R3up until SAP NetWeaver 2004 (Basis release 6.40). With SAP NetWeaver 2004s/Basis 7.00, the name changed to SAPup.

Note!

Even today, it is possible to run SAPup (the successor to R3up) in the so-called "scroll mode," where you run it in an OS session window like a DOS shell for Windows or a Telnet session for UNIX and interact with it directly. This can help if for some reason using the UA GUI is impossible, but it should only be used as a last resort.

SDT for Java is a very close relative of the installation utility SAPINST (the picture of the cable-stayed bridge on the progress screen will be familiar to you if you have ever done SAP installations for Basis 6.40 or higher). Like UA, SDT consists of a server and a GUI client. However, as you'll see next, its features are more limited and it does not provide the same flexibility as UA for ABAP.

The ABAP Upgrade Assistant (UA)

Chances are that you will be doing far more ABAP upgrades than Java upgrades. Therefore, let's look at the Upgrade Assistant (UA) for ABAP first.

UA is the "driver" program of the entire upgrade process. UA itself does not do any technical upgrade work. Its main responsibility is to handle communication with the users (who are connected via their UA GUI) and to control and monitor the upgrade processes.

UA is designed as a two-tier client-server application with an Upgrade Assistant Server (UA Server) running on the upgrade host and one or more UA GUI sessions on users' workstations.

An important characteristic of the ABAP UA (which it does not share with its Java counterpart) is its capability to have multiple users logged on via GUI to the running upgrade, and the distinction that it makes between two user roles: the "administrator" and the "observer." We'll discuss these upgrade roles first before taking a closer look at the UA Server and GUI.

Users: administrator and observer

When you use the UA GUI for the ABAP PREPARE and upgrade, you must log on to the GUI as a valid user; this is one of the main differences between the ABAP UA and the Java SDT. UA knows two users: the Upgrade administrator (user ID: admin) and the Upgrade observer (user ID: observer).

- The *administrator* has active control over the upgrade process. This user has the ability to start and stop upgrade processes and to interact with the upgrade via input screens in the GUI.
- An *observer* has only a passive role. Observers can monitor the progress of the upgrade via the GUI, but they cannot actively influence it. Their GUI screens are not enabled for input.

At any time during the upgrade, at most one administrator session can be logged on, but there can be any number of observers. While they are connected to UA, GUI users can dynamically change roles: administrators can demote themselves to observers, and an observer can promote himself to administrator (thereby automatically demoting the current administrator if one is logged on).

Administrators and observers are roles defined by the UA GUI. It is not necessary for a UA GUI to be active at any time; the upgrade can run nicely even without any user session connected. For instance, if a lengthy upgrade phase starts late in the evening and is likely to run through the best part of the night, the administrator and the present observers may decide to log off from their UA GUI and go home. At that moment, there is neither an administrator nor an observer, but that does not bother the running upgrade in any way. If the upgrade stops, for example, because of an error, then it will simply wait until an administrator comes online and communicates with the upgrade via the error screen in the UA GUI. If the alert service is enabled (discussed later when we look

at the UA GUI features), then UA will also invoke the alert script so that someone is notified of the stop.

Both user roles are protected by a password. You set the administrator password the first time you start the UA Server (discussed next). You do not create named users for UA; to log on with a certain role, you simply enter the password of that role. The logon screen of the UA GUI does give you the option to enter a name, but this is optional and purely informative; you may choose to enter your real name there or anything that helps others to identify you. You can also enter a contact telephone number on the logon screen.

The UA Server

The server side of UA is the Upgrade Assistant Server (UA Server). This process runs on the same host as the actual upgrade, that is, the host where the central instance of the SAP system resides. You do not interact directly with the UA Server except once: when you start the server for the first time, it will prompt you for the password of the administrator. After that, the server will never interact directly with you again, so you may then let it run as a background process (on OSs supporting this). You'll see the commands to run the UA Server later in the article. All further communication between the administrator and the UA Server goes via the UA GUI.

The PREPARE and the ABAP upgrade program (SAPup) run under the control of the UA Server. The server will start these processes when instructed to do so from the administrator's UA GUI session. From then on, it monitors the processes, forwarding their console output to all open UA GUI windows. When an upgrade stops for interaction, either because it needs parameter input or because an error has occurred, the server will handle the user interaction via the administrator's UA GUI session (if an administrator is currently logged on), and pass on this input to the upgrade process.

The server is started from the OS command line. It is stopped from inside the UA GUI, or you can also terminate it at the OS level when it is no longer needed. All this will be covered in detail next.

The UA GUI

The client is the UA GUI. Client and server communicate with each other via a TCP/IP port (port 4241). On a second port (4239), the UA Server is capable of receiving HTTP requests, allowing it to interact with a browser running on a user's PC. Access via a browser can be used to launch the UA GUI or to request information related to the upgrade, such as the phase list or the evaluation form. We'll take a closer look at the browser interface later when we examine how to start the UA GUI.

Although the UA GUI runs on the user's PC, it is not necessary to perform a separate installation of the upgrade tools on the PC. The UA GUI (like the UA Server) is a Java application that is uploaded to the server during the initial run of the PREPARE script (described later in the article). Both the UA Server and UA GUI are Java applications residing in Java Archive (JAR) files. When the user starts the UA GUI from the browser, the Java code for the UA GUI is loaded onto the PC under the control of Java Web Start.

Java Web Start is a Java application used to deploy and run client-side Java applications. The traditional method for client-side code is the use of applets, but applets face two major limitations that make them very difficult to use in many environments:

- Applets work under serious security restrictions; for example, they normally cannot access local files or external network addresses.
- Applets depend on the Java VM (virtual machine) built into the browser, which may cause version problems and incompatibilities.

Java Web Start overcomes these restrictions by deploying and starting applications on the client. At the same time it is also capable of uploading and installing the correct version of the Java Runtime Environment (JRE).

Figure 1 on the next page shows a simple diagram of the UA architecture. Here an administrator and several observers are logged on to the UA GUI; there are also active browser sessions communicating with the server via its HTTP interface.

SDT for Java

Like UA for ABAP, the upgrade tool for Java has a client-server design, consisting of the SDT Server and SDT GUI. SDT is the underlying framework that the Java upgrade tool has in common with the SAP installation utility SAPINST. Being built on the same platform, SAPINST and the Java upgrade GUI have much the same look and feel as we mentioned earlier.

The Java upgrade behaves in a way that is very similar to its ABAP counterpart. The server runs on the upgrade host and does not interact directly with the user. It controls the two major phases of the upgrade, the Java PREPARE and the Java upgrade (SAPJup). The server also handles all communication between the user and the upgrade process. Messages from the upgrade process, including requests for user input and error reports, are sent on to the SDT GUI. User commands and user input are transmitted from the GUI via the server to the upgrade process.

The SDT GUI is started in the same way as the UA GUI, namely from a browser. In the browser, you connect to the SDT Server via port 6239. The application is loaded from the server and started via the Java Web Start framework (see the UA GUI discussion in the previous section). The SDT GUI communicates with the server via port 6241.

Figure 2 shows the architecture of the Java upgrade.

The similarity with the ABAP upgrade is clear, but you may be struck by a major difference too. In **Figure 1**, several users were connected to the ABAP upgrade via their UA GUI; only one of them, the administrator, could be in control, but there could be any number of observers simultaneously logged on. In **Figure 2** for Java, you only see one SDT GUI connected. This is because the Java upgrade has no notion of different users. It does not make a distinction between active administrators and passive observers, and only allows one user to be logged on to the SDT GUI at any one time. Any attempt to open a second SDT GUI on a running Java upgrade will simply produce a rather terse error message (see **Figure 3**) and leave you no other choice but to close this GUI.

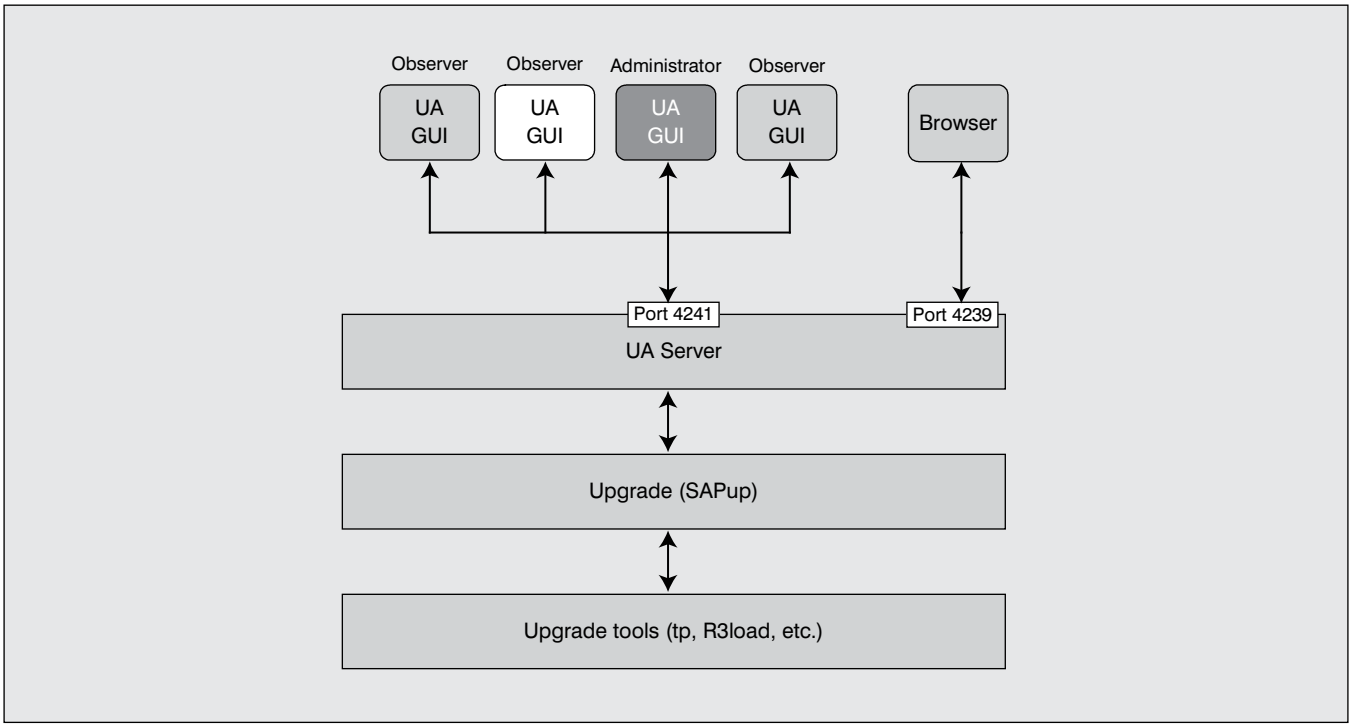

Figure 1 Upgrade Assistant (UA)

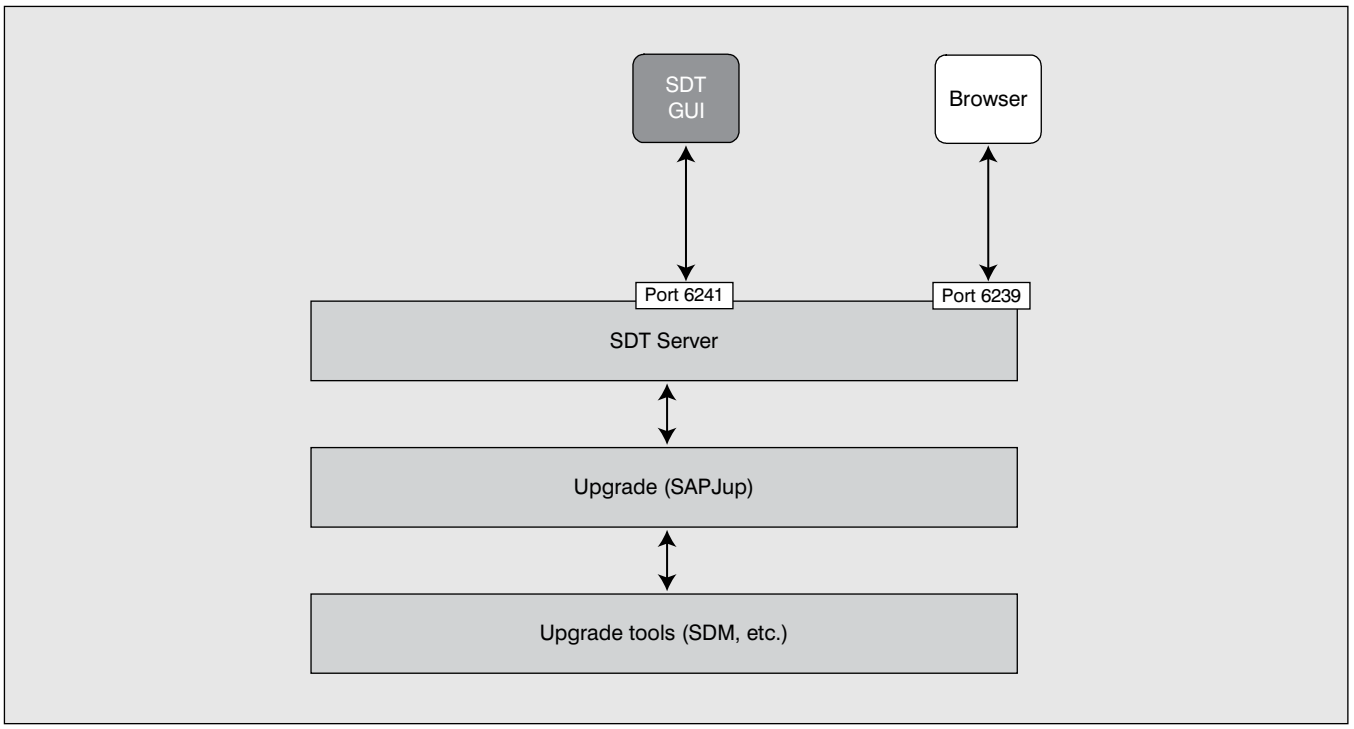

Figure 2 Java upgrade

Figure 3 Error when starting a second SDT GUI

Prerequisites for the upgrade tools

The upgrade tools for both ABAP and Java need a Java installation on the server and on the workstation where you run the GUI.

Java SDK or JRE for ABAP Upgrade Assistant

The ABAP UA is a Java application, which means that a JVM must be installed on the upgrade host (where the UA Server runs) and also on every workstation where you intend to run the UA GUI.

The Java requirement is strictest on the server side. The UA Server requires the Java Software Development Kit (SDK) version 1.4 or higher. An installation of the JRE is not sufficient.

The UA GUI is less demanding. Here the Java version must be 1.1 or higher, and both the Java SDK and JRE will do.

> ***Important!***
>
> If the SAP system to be upgraded contains a Java stack, then the host must also meet the requirements for AS Java. These will normally be stricter than those for the UA Server. With AS 7.0, the Java SDK version must be 1.4, not lower (1.3), nor higher (1.5). More precise restrictions, for example, a specific minimum version of the 1.4 Java SDK, may apply. Full details are in SAP Note 723909 and in a set of platform-dependent notes, which are all listed in SAP Note 723909. In practice, you should always use the latest available Java SDK within the range of supported versions (which is actually what SAP Note 723909 tells you to do).

To check the currently installed Java version, both on the server and on the GUI workstations, open a command shell, log in as the SAP administrator (user <sid>adm), and type the command:

```
java –version
```

Figure 4 shows an example on a Windows PC.

This workstation runs a Java 1.5 version. For UA (Server or GUI), this is all right. For AS 7.0 Java, this would not be all right (only 1.4 is supported).

Java SDK or JRE for SDT

SDT is also a Java application. Because SDT is used to upgrade AS Java systems, a Java SDK must obviously be installed on the server. As explained previously, version 1.4 SDK is required. At the time of writing, 1.5 SDK was not supported. 1.4 SDK also meets the requirements for the Java upgrade tools on the server side.

On the client side, the workstation must meet the requirements of the SAPINST tool, that is, a Java JRE must be present. We recommend that you use 1.4 JRE or SDK on the workstation like on the server, although 1.5 should also be okay.

Installation of the upgrade tools: ABAP

Making the upgrade tools ready for use is easy: All you need to do is run a script and, in the case of ABAP, replace the extracted version of the SAPup utility with the latest patch.

Initial run of PREPARE

The script for the initial extraction is located on the ABAP Upgrade Master DVD. This script is called PREPARE, which is a little confusing because the first major phase of the upgrade is also called PREPARE. Keep in mind that the two are not related.

The following section describes how to use the PREPARE script on UNIX/Linux and on Windows.

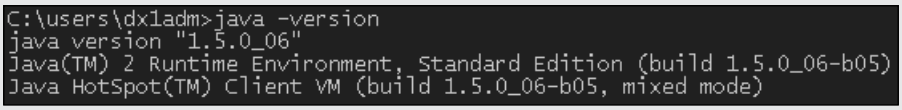

Figure 4 Checking the Java version

For IBM iSeries, the procedure is different; refer to the section, “Starting PREPARE for the First Time,” in the SAP Upgrade Guide for IBM iSeries for the exact commands. This guide is available in the Installation and Upgrade Guides Portal in the SAP Service Marketplace (http://service.sap.com/instguides).

Initial PREPARE run: UNIX and Linux

In this example, we assume that you have placed disk copies of the upgrade media in a server directory /sapcd. You have placed the ABAP Upgrade Master DVD in a subdirectory named /sapcd/UPGMSTR_ABAP.

To run PREPARE initially, proceed as follows:

1. Log on as user <sid>adm.
2. Switch to the upgrade directory:

```
cd /usr/sap/put
```

3. Type the command:

```
/sapcd/UPGMSTR_ABAP/PREPARE
```

If the name of the ABAP upgrade directory is not /usr/sap/put, then you must specify the path via a command-line argument, for example:

```
/sapcd/UPGMSTR_ABAP/PREPARE
 upgdir=/usr/sap/putPRD
```

Keep in mind that to use an upgrade directory with a name different from the default, you must also adapt the SAP profile parameter DIR_PUT.

4. The PREPARE script creates the required sub-directories in the upgrade directory, installs the initial set of upgrade tools (including the SAPup executable), and performs some basic tests, for example, checking the availability of the C++ runtime library.
5. Finally, PREPARE prompts you for an action:

```
Select operation mode:
            - “EXIT”
            - “SERVER”
            - “SCROLL”
Enter one of these options [EXIT] :=
```

Possible choices at this point are:

- EXIT — End the script. This is the default choice and also the one we recommend you use.
- SERVER — This starts the UA Server. It is preferable to start the UA Server manually later.
- SCROLL — Starts SAPup in scroll mode (see the earlier section, “The control programs: Upgrade Assistant and SDT Server/GUI,” for details on SAPup). Except in very special circumstances, you will never use this option.

Initial PREPARE run: Windows

The command file PREPARE.BAT does nothing other than invoke the JavaScript file PREPARE.JS. For this

method to work, Windows Script Host (WSH) must be available. WSH is a facility that provides powerful scripting capabilities. WSH is installed by default, so normally it will always be available. However, it is possible for system administrators to disable or uninstall WSH for security reasons (scripts have been abused as virus carriers). The WSH version on the SAP server might also be too low. The SAP upgrade requires at least WSH version 5.6.

You don't need to check anything related to WSH before starting the PREPARE script. If WSH is missing or its version is too low, then PREPARE will report this. If you do want to check the WSH version upfront, then do the following:

1. Open a DOS shell (CMD).
2. Type the command:

```
Cscript
```

3. The output shows the version banner followed by the command-line options.
4. If necessary, you can download WSH free of charge from the Microsoft Download Center at http://www.microsoft.com/downloads.
5. To run the PREPARE script, you must log on as the <sid>adm user.

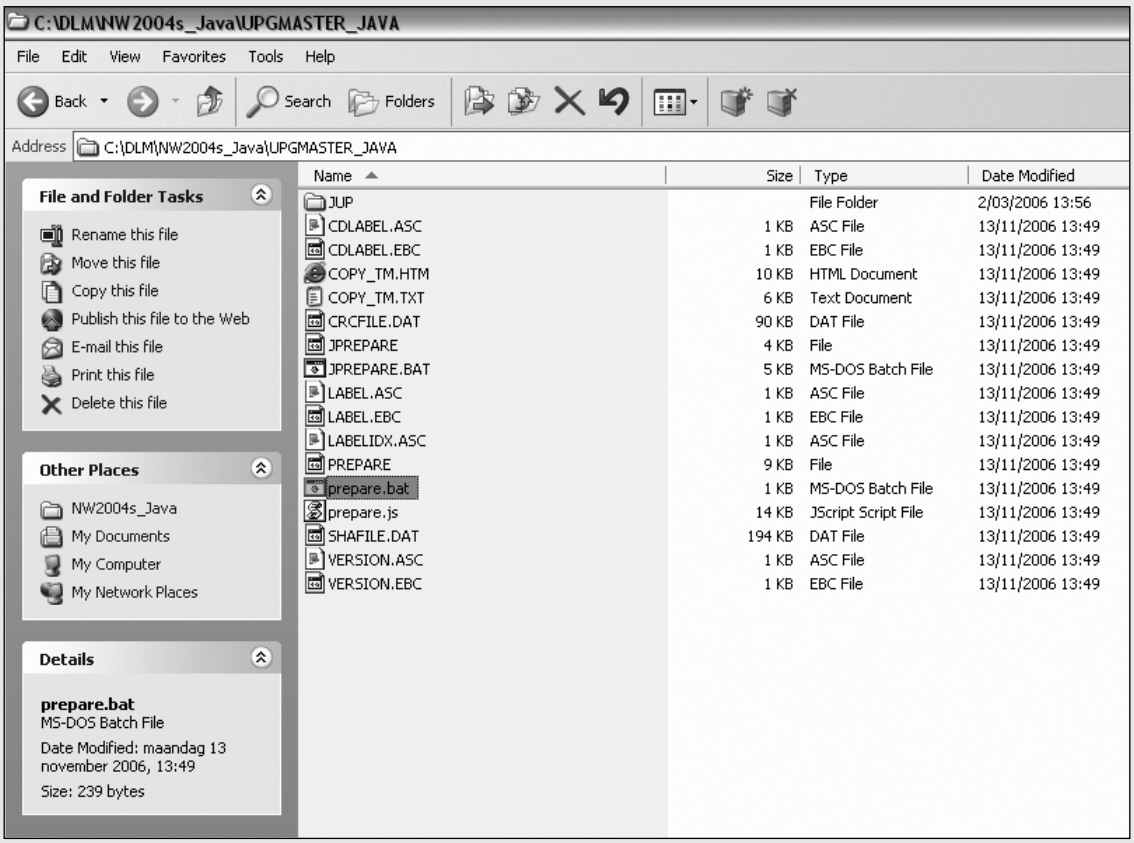

Figure 5 Select PREPARE.BAT in Windows Upgrade Master

6. To start the script, open Explorer, and browse to the directory of the Upgrade Master DVD (see **Figure 5**).

7. Start PREPARE.BAT.

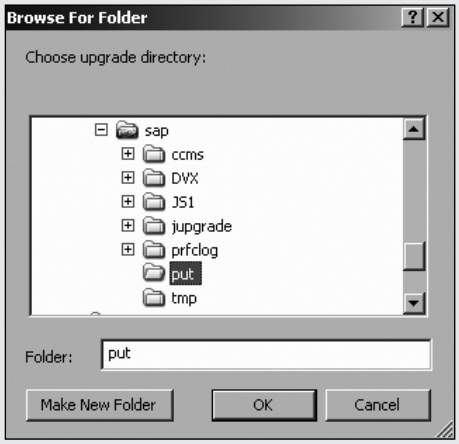

Figure 6 Select upgrade directory

8. A file selection box opens (see **Figure 6**). Browse to the ABAP upgrade directory.

9. Click on OK. After a few seconds, a DOS box opens prompting you for an action (see **Figure 7**).

The available choices are the same as with UNIX: EXIT (the default), SERVER, and SCROLL. See the previous section for a description of the choices.

Replace SAPup

The script has installed the SAPup executable (SAPUP.EXE in the case of Windows) in the subdirectory bin of your upgrade directory. Before starting the upgrade, you must now replace this version of SAPup with the latest patch available from the SAP Service Marketplace at http://service.sap.com/PATCHES.

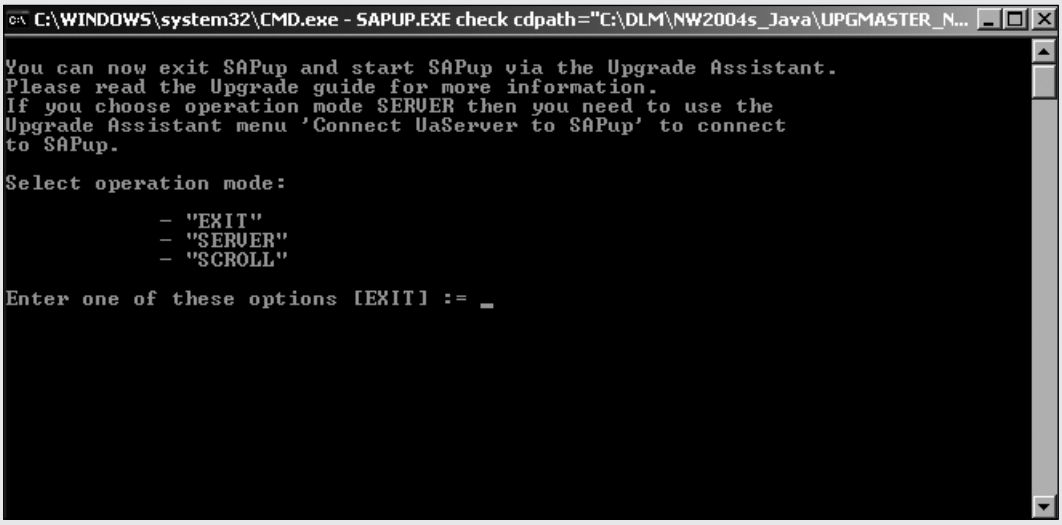

Figure 7 Action prompt after initial extraction

> ***Note!***
>
> Before starting an ABAP upgrade, you should replace the version of the SAPup executable delivered on the ABAP Upgrade Master DVD with the latest available patch. For information regarding the appropriate SAPup version to download, see SAP Note 821032.

Use the following instructions to install the SAPup patch after you have run the initial PREPARE extraction script:

1. Open a command shell on the server and log on as <sid>adm.

2. Determine the current version of SAPup.

   UNIX/Linux:

   ```
   ./SAPup -V
   ```

   Windows:

   ```
   .\SAPup -V
   ```

   This will also show the flavor. Example:

   ```
   This is SAPup version 7.00/2,
    build 24.067
   ```

3. Optionally, save a copy of the current SAPup.

   UNIX/Linux:

   ```
   cd /usr/sap/put/bin
   mv SAPup SAPup.orig
   ```

   Windows:

   ```
   cd \usr\sap\put\bin
   rename sapup.exe sapup_orig.exe
   ```

4. Extract the patch archive. The actual name changes with the patch level. <...> denotes the variable part of the name:

   ```
   SAPCAR -xvf SAPup7002_<...>.SAR
   ```

5. Repeat the version check. The version will be higher, but the flavor must be the same.

   UNIX/Linux:

   ```
   ./SAPup -V
   ```

   Windows:

   ```
   .\SAPup -V
   ```

> ***Caution!***
>
> Exchange the version of SAPup only *before* you start the upgrade. Never change SAPup *during* the upgrade unless SAP instructs you to do so!

Using the ABAP Upgrade Assistant

In this section you will learn how to start the ABAP UA and how to use the functions of the UA GUI.

Starting the UA Server

If possible, you should let the UA Server run as a background process to prevent unwanted terminations, for example, because you accidentally close the window it is running in, or because the network connection server and PC is interrupted. Windows does not give you this possibility, so you will have to run the UA Server in a DOS shell that normally remains open throughout the upgrade. On UNIX/Linux, you can start the UA Server as a background

process; there you can also use the nohup[2] facility to make sure the server process keeps running even after the session from where it was started logs off.

The very first time you run the UA Server, it will prompt you for the administrator password. User interaction is only possible with a process running in the foreground, so the first start of the server should always be in the foreground, even on UNIX and Linux.

For the first run, follow these steps:

1. Open a command shell on the upgrade host (DOS box, Telnet session), and log in as <sid>adm.
2. Type the command:

```
java -cp /usr/sap/put/ua/ua.jar UaServer
```

   For Windows, use the same command with the slashes reversed.

   The name of the Java class (UaServer) is case-sensitive, also on Windows.

3. The server prompts for the password of the administrator user. Enter and confirm the password:

```
Please enter administrator password
Enter password:*********
Confirm password:*********
```

4. On Windows, you can leave this server process running. The following steps are for UNIX/Linux only.
5. On UNIX/Linux, wait until you see the message:

```
UaServer> Ready
```

6. At that point, terminate the running server with Ctrl+C.
7. Restart the UA Server as a background and no-hangup process:

```
nohup java -
cp /usr/sap/put/ua/ua.jar UaServer &
```

[2] nohup (no-hangup) is a feature of UNIX that allows you to start a background process inside a command session and keeps this process alive when your session ends (normal background processes terminate along with the shell from where they were started). For example, to run a command “comm” in this way, type “nohup comm &” (the ampersand specifies that comm must run in the background, and the nohup prefix ensures the command does not get terminated when the session ends).

Starting the UA GUI

To start the UA GUI, you first connect to the UA Server using an Internet browser. The Java Web Start framework will then launch the GUI on your workstation.

Connect to the UA Server via a browser

1. To connect to the UA Server, open a browser on your PC.
2. Type the URL http://<server>:4239 where <server> is the host name (or possibly the full domain name, depending on how networking is set up) of the host where the UA Server is running. Example:

```
http://sapdev.mycomp.com:4239
```

   If you decide to start the browser on the upgrade host itself rather than on a local PC, you may also use the URL http://localhost:4239.

3. The main page of the ABAP UA now opens (see **Figure 8** on the next page).
4. If you do not see this window, check the following:
   - Is the host name you specified correct and can it be resolved to an IP address from your PC? Use ping to check this.
   - Is the UA Server running and ready to receive browser requests? Open the command window where you started the UA Server and make sure the server has displayed the messages:

```
UaServer> Starting HTTP server
UaServer> HTTP server started
UaServer> Ready
```

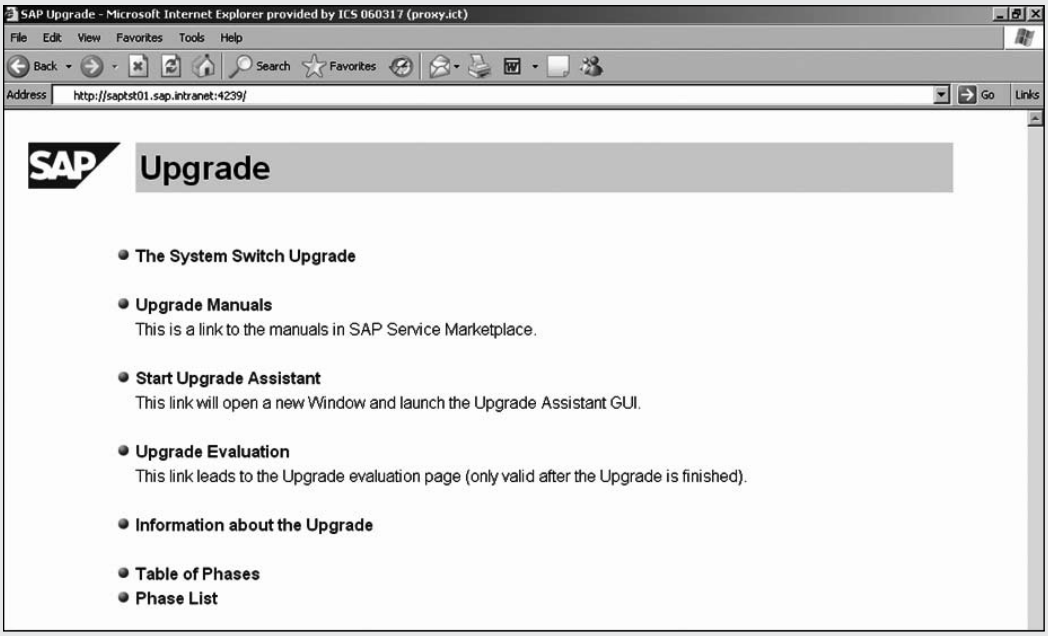

Figure 8 ABAP UA main page

- If you started the server as a nohup background process (UNIX and Linux), then you might not see these messages in the command window. In that case, look at the standard output file (typically nohup.out in the directory from where you started the server).

If the server encounters an error, it writes a Java stack trace and terminates. This trace can be quite long, but the first few messages will usually reveal what the problem was.

In some cases, the port 4239, which is used for communication between the browser and the UA Server, may be unusable, for instance, because it is blocked by a firewall or because it is already in use by another application (which could even be another SAP upgrade running on the same host). In that case, ask the network administrator either to open the ports for UA (if blocking by the firewall is the problem) or to determine alternative port numbers you can use. See the upcoming section "SAPup command options" for information about the port numbers used by UA.

Start the UA GUI

1. On the main upgrade page, click on Start Upgrade Assistant.

2. The Java Web Start window pops up, and the loading of the application begins (see **Figure 9**).

   Sometimes Java Web Start encounters an error while it loads the UA GUI to the workstation; see the next section.

Figure 9 Java Web Start loading the UA GUI

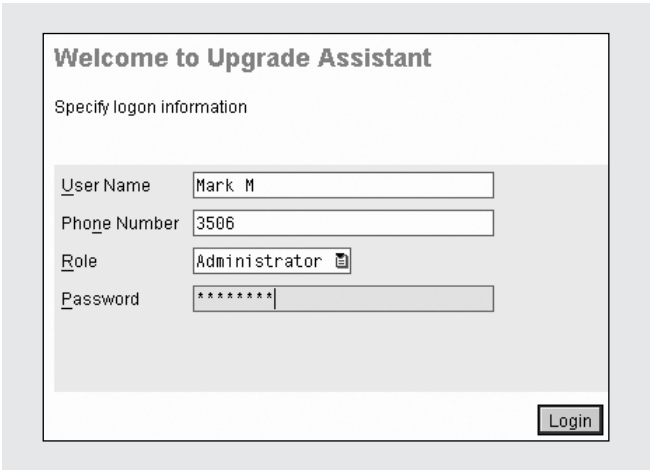

Figure 10 Logon screen of the UA GUI

3. When the loading is complete, the logon screen of the UA GUI appears (see **Figure 10**).

4. Log on as an administrator. The user name and password fields are mandatory. What you enter for the user name is entirely up to you; anything that identifies you is acceptable. The UA GUI only demands that the user name be at least two characters long.

   If you expect that several users will access the UA GUI during the upgrade, then it is convenient that you also specify your telephone number.

5. Click on Login.

6. On the next screen, you can choose functions from the menu. We describe the available functions in more detail in the section "Features of the UA GUI." These functions are common to all ABAP upgrades (the UA GUI menu is always the same regardless of the product you are upgrading).

Java Web Start errors while loading the UA GUI

If Java Web Start is unable to upload and start the UA GUI to your workstation, an error Unable to launch Upgrade Assistant appears (see **Figure 11** on the next page). Click on the Details button to see more information about the problem. The most useful information is probably in the upper parts of the Java stack trace, which you find under the Exceptions tab.

Java stack traces are not gems of readability, but most of the time you will be able to find at least one interesting term pointing to the cause of the error. In this case UnknownHostException followed by a server name indicates that Java Web Start was unable to resolve the host name of the server. In our experience, this is the most common problem you can encounter with the Java Web Start mechanism. The usual cause is that the host name of the SAP server is not known in the network configuration (DNS). Adding an entry in the local HOSTS file on the PC often — but not always — helps; otherwise, you must report the problem to the network administrator.

Features of the UA GUI

Let's begin with a brief tour of the menu functions, and then we'll turn our attention to the different services UA provides for controlling the upgrade and, whenever necessary, changing its behavior.

The File menu:

- **File → Change Role:** Switches the UA GUI session between administrator and observer roles. If you switch from observer to administrator, the current administrator (if there is one) is automatically demoted to observer. In **Figure 12** on the next page, user Mark changes his current role from administrator to observer.

- **File → List of users:** Lists all users currently logged on to UA. A pop-up window opens showing the name, telephone number, and current role of all connected users (see **Figure 13** on the next page).

- **File → Exit:** Ends your UA GUI session. Leaving the UA GUI has no effect at all on the UA Server (and thus the upgrade), which simply continues running.

The Administrator menu:

- **Administrator → Start PREPARE:** Starts SAPup in PREPARE mode.
- **Administrator → Start SAPup:** Starts SAPup in upgrade mode. PREPARE must be complete for this.
- **Administrator → Start SAPup with options:** Starts SAPup with a command option. You can run SAPup interactively to perform some special operations. You can do this directly on the host (via the command line) or via the function "Start SAPup with options" inside the UA GUI. The default command option here is "set stdpar,"

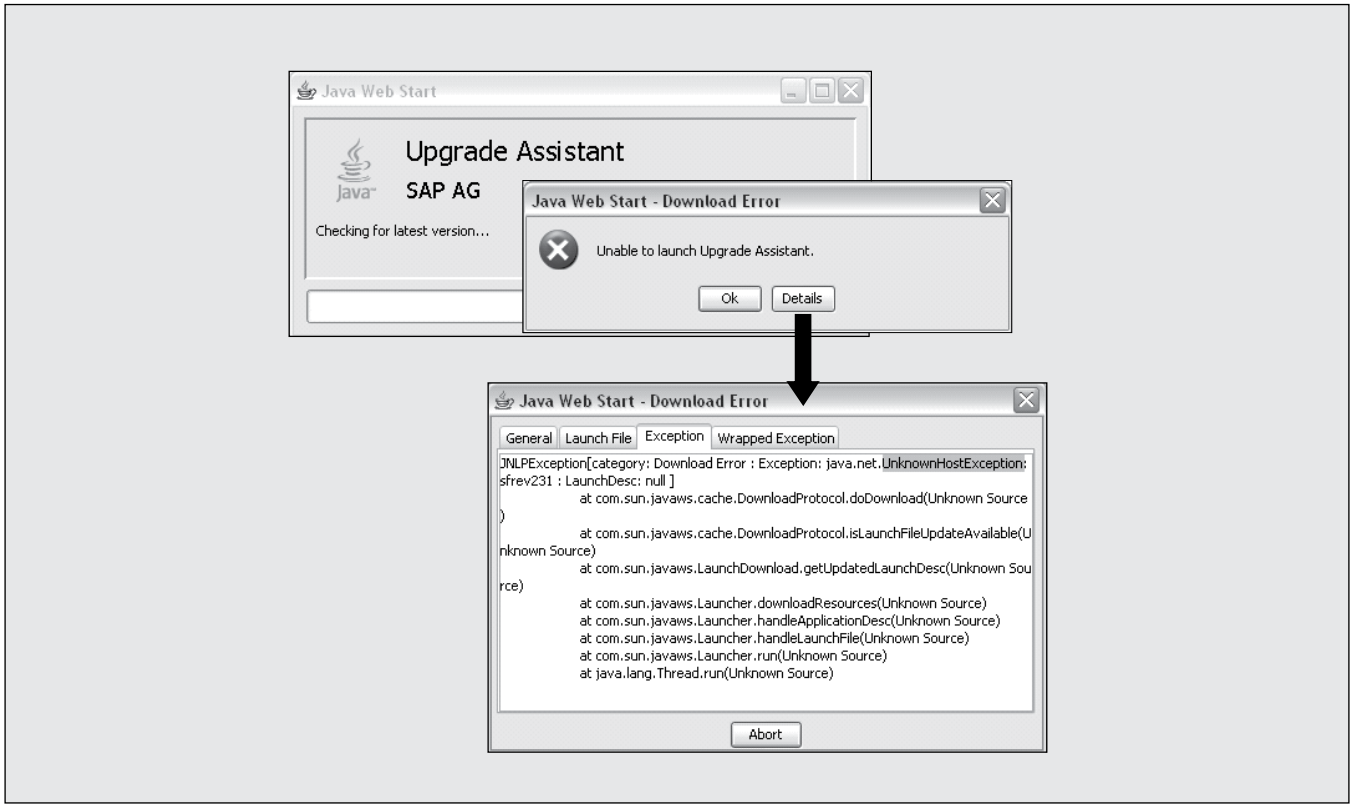

Figure 11 Java Web Start error

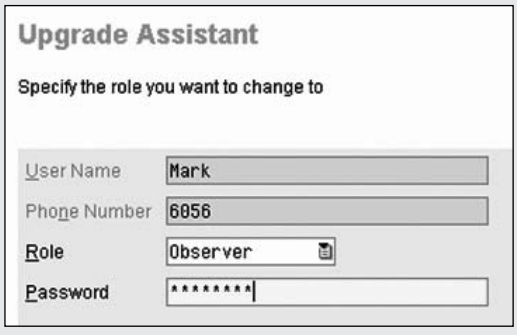

Figure 12 Change role in the UA GUI

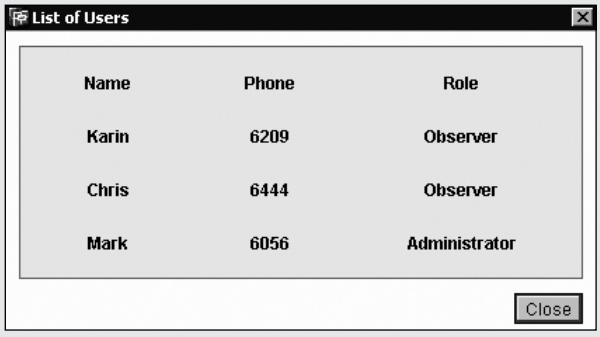

Figure 13 Users logged on to UA

which is the option you need if for some reason you want to change the upgrade parameters. See the upcoming section "SAPup command options" for more information.

- **Administrator → Stop SAPup after current phase:** This instructs a running SAPup process (in PREPARE or upgrade mode) to halt after it finishes the current upgrade phase. This option is useful if for some reason you need to interrupt the upgrade, for example, to reboot the server, to change database or SAP profile parameters, and so on.

- **Administrator → Terminate SAPup:** Terminates a running SAPup process immediately. This is the emergency brake, which you should only use in exceptional circumstances.

- **Administrator → Connect UaServer to SAPup:** With this function, which is rarely used, you can instruct a newly started UA Server to connect to an already running SAPup process. You could use this, for instance, when the UA Server stopped abnormally and had to be restarted. The new UA Server would then connect to SAPup (which was not affected by the abnormal stop of the server).

  This is, to put it mildly, not the most reliable of UA features; in fact, our experience is that this rarely — if ever — works. If you lose the UA Server while SAPup is running, your best bet is probably to wait until a "natural" stop of SAPup, for example, to request user input, and then stop and restart both SAPup and the UA Server. In the meantime, you can monitor the upgrade via the logs in the upgrade directory.

- **Administrator → Disconnect UaServer from SAPup:** This breaks the connection between the UA Server and SAPup. SAPup will continue until it needs to interact with the user and will then stop. We have never used this function and, given the quirky behavior of its Connect counterpart, we're not likely to try.

- **Administrator → Set Alert:** Use this function to make the UA GUI invoke a script or command of your choice when it stops for interaction. This is a very useful feature if you want to let the upgrade run unattended, but you want to be informed when your intervention is needed. The alert function is described in the next section.

- **Administrator → Change Passwords:** This function lets you change the administrator and observer passwords. If you forget the administrator password, then you need a special procedure because you cannot log on to the UA GUI; see the "Tips and tricks" section later in the article for the solution.

- **Administrator → Terminate Upgrade Assistant Server:** This function stops the UA Server. It is a clean alternative to killing the server from the OS.

  The Services menu:

- **Services → File Service:** Lets you select and display files on the upgrade host. This is normally used to display upgrade logs. See the upcoming section "The File Service" for details.

- **Services → Upgrade Monitor:** Brings up a progress window showing the upgrade phases. See the upcoming section "The Upgrade Monitor."

- **Services → Console Window:** Displays the main upgrade log. See the upcoming section "The Console Service."

- **Services → SAP Notes Search:** Opens a browser session in the SAP Service Marketplace to search notes. See the upcoming section "The Notes Service."

  The Help menu:

- **Help → Introduction:** Opens a browser session with information about the upgrade tools. Interesting reading but unfortunately not always up to date. At the time of writing, the information provided here was for the upgrade tools of Basis 6.40 and not of Basis 7.00, which could be confusing.

- **Help → About:** Shows the UA version and the copyright notice.

The Alert Service

Unless you work in a round-the-clock team and an administrator is constantly watching, chances are that the upgrade will at certain times run unattended. Although this may give you a welcome opportunity to get some sleep or to do some other important task, you don't want to come back to the upgrade to find out that it has stopped for user input or because of an error and has been in that idle state for many hours. To prevent these nasty surprises and the resulting loss of time, you can instruct the UA Server to trigger an alert whenever it stops. When the alert goes off, the UA Server invokes a command of your choice. This could be a shell script to send an email to the members of the technical upgrade team or a script that sends an SMS (Short Message Service) message to your mobile phone, for instance.

To enable alerts, do the following:

1. Choose Administrator → Set Alert in the UA GUI menu.
2. In Alert Command, type the full path name of the command or script you want the UA Server to call.
3. In Alert Text File, you can specify the name of a file on the host where the UA Server will write the same information that it sent to the UA GUI when the upgrade stopped.
4. In Alert Delay, enter a delay the UA Server must observe between the moment the upgrade stops and the moment the alert is triggered. The default is 500 seconds. You might want to reduce this somewhat, for example, to 2 or 3 minutes, but do not set the delay to 0 because this will cause many unnecessary alerts.
5. With the Set active flag, you can enable and disable the alert configuration at will.

If alerts are active, then the following happens when the upgrade stops for user interaction (this may be normal interaction, i.e., prompting for user input, or an error stop):

1. The UA Server waits for the amount of time set in the Alert Delay.
2. If the user does not restart the upgrade before this delay expires, the UA Server writes the text of the upgrade message to the alert text file.
3. The UA Server then invokes the alert command.

In the example given in **Figure 14**, the UA Server is instructed to call a script upgrade_sms whenever the upgrade stops for more than 2 minutes. Judging by the name, the script creates an SMS message, which it sends to some administrator's mobile phone. More sophisticated scripts are, of course, possible; you could, for instance, design a shell script that "noisily" alerts the administrator by sending a brief

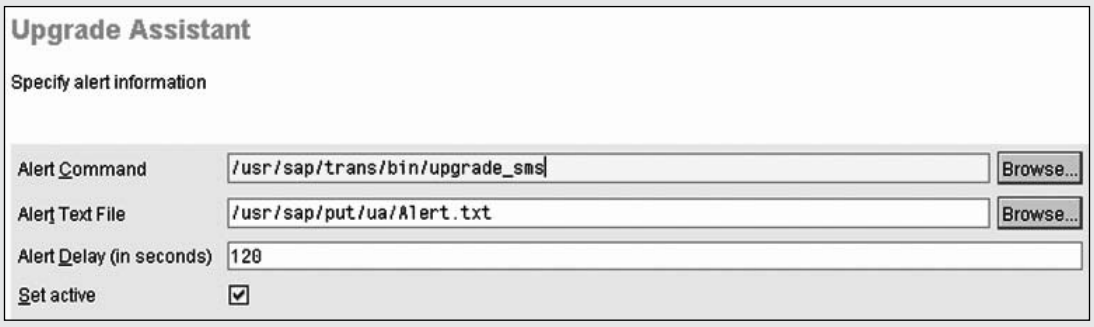

Figure 14 Defining an alert

SMS and at the same time also sends an email with the contents of the alert text file.

Figure 15 shows an example of an alert text file. Here the upgrade stopped not because of an error but to notify the user that action is needed, which, in this case, is upgrading the liveCache during an APO (Advanced Planning and Optimization) upgrade.

The File Service

The File Service in the UA GUI lets you select and display files on the upgrade host. A file selection window opens, showing the directory structure of the host. You can use this window to navigate through the directory tree. Alternatively, if you know the full path name of the file you want to display, you can type this into the File input field at the bottom (on the left of **Figure 16**, on the next page). The file is displayed directly. Unfortunately, you can only enter file names and not directory names into this field.

If the directory contains files, the file chooser window lists them (right, **Figure 16**).

Double-clicking on a file opens a new window with the contents of that file (see **Figure 17** on the next page). With long files, use the More and End of File buttons to navigate.

The Upgrade Monitor

Via Services → Upgrade Monitor, you can see how far the upgrade has progressed. Click on Close to leave the monitor screen and return to the main UA GUI window.

Figure 18 (on page 25) shows the Upgrade Monitor during the actual upgrade. Here the progress screen shows all the upgrade phases.

During PREPARE, which is subdivided into modules, you only see the phases of the currently executing module. You see an example of this in **Figure 19**, on page 25.

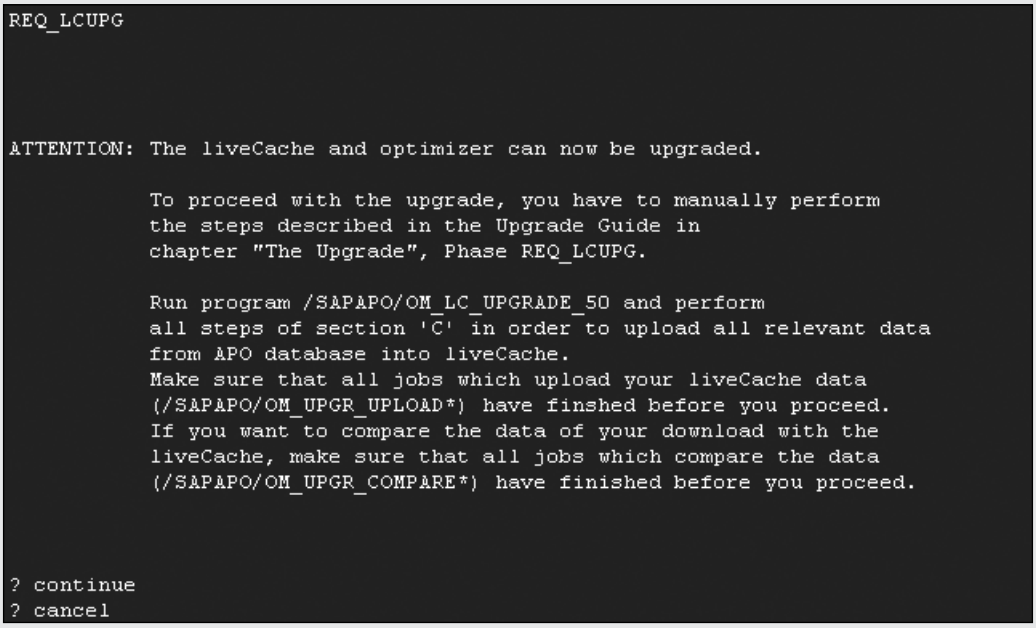

Figure 15 Alert text file

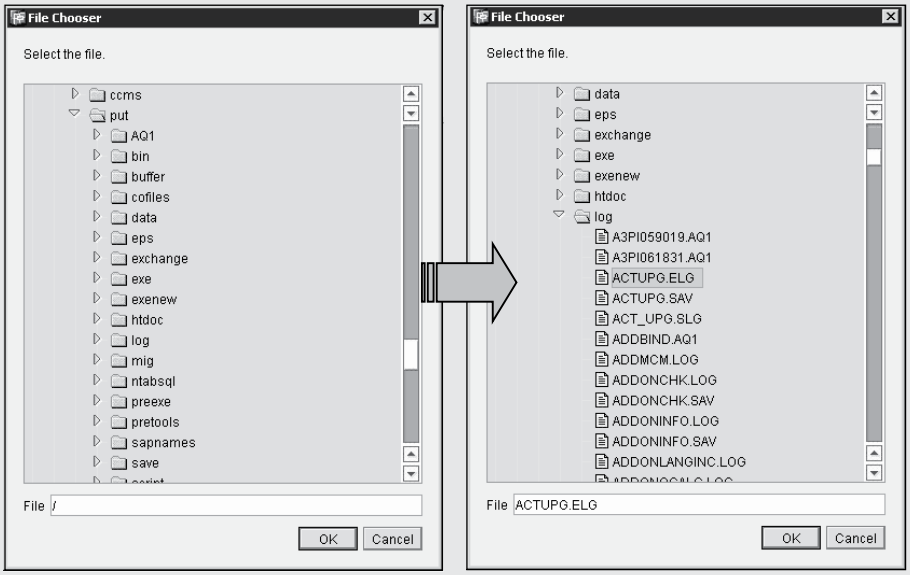

Figure 16 UA File Service — File Chooser

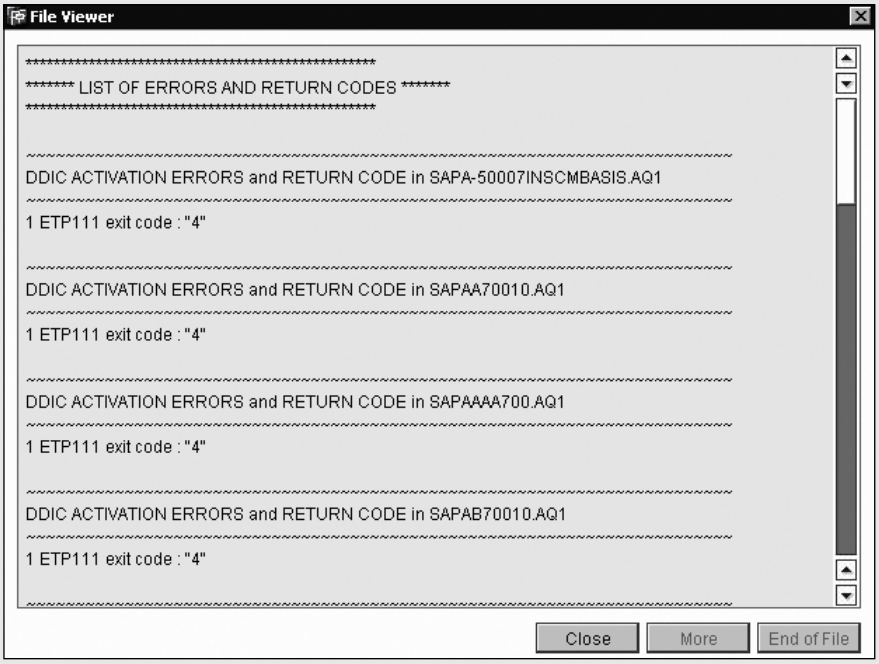

Figure 17 UA File Service — Display File

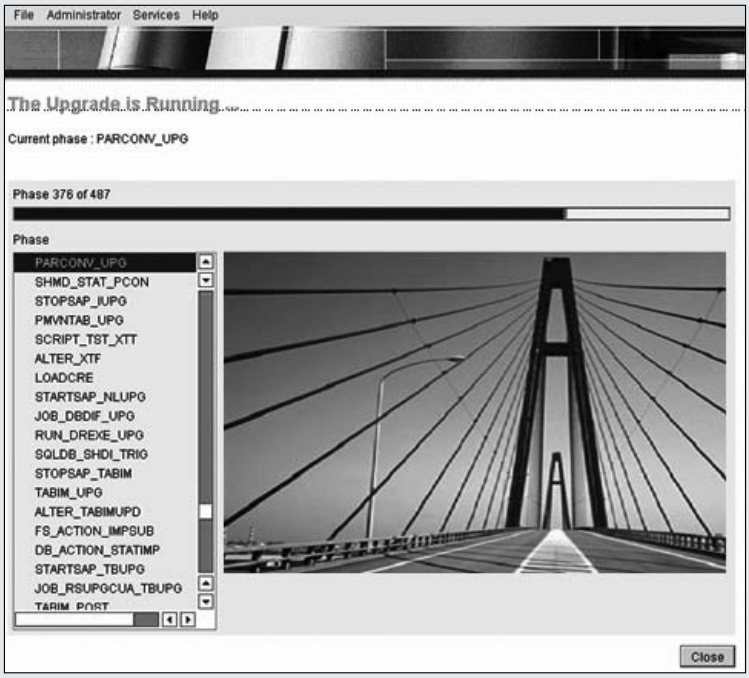

Figure 18 Upgrade Monitor in the UA GUI

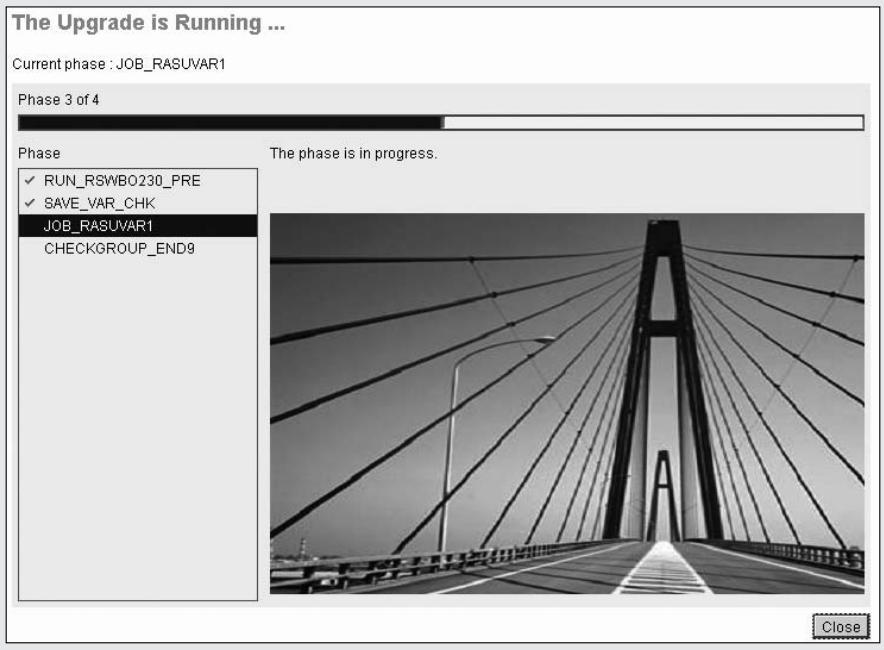

Figure 19 Upgrade Monitor during PREPARE

> ***Note!***
>
> Older versions of UA also showed an estimate for the remaining runtime, both for the current phase and for the entire upgrade. In practice, so many factors affect the runtime that a sensible prediction is very difficult to make. Because the time estimates were often wide off the mark, this feature was removed.

The Console Service

While the upgrade is running, the UA GUI window displays a progress screen (see **Figure 20**) that contains messages for the beginning and end of each phase, other messages shown to the user (e.g., to signal errors), and all interaction with the upgrade administrator (prompts and replies). You can scroll back inside this window to see all the messages and user input since the beginning of the upgrade.

When the upgrade stops to interact with the user, this progress screen is temporarily replaced with an input window. If you want to look at the progress screen at this point, for example, because you want to check some input you entered in an earlier phase, then choose Services → Console window. The progress screen will then reappear.

Another advantage of the console window is that copying text is enabled here. This is useful if you want to copy/paste parts of the upgrade dialog into your documentation.

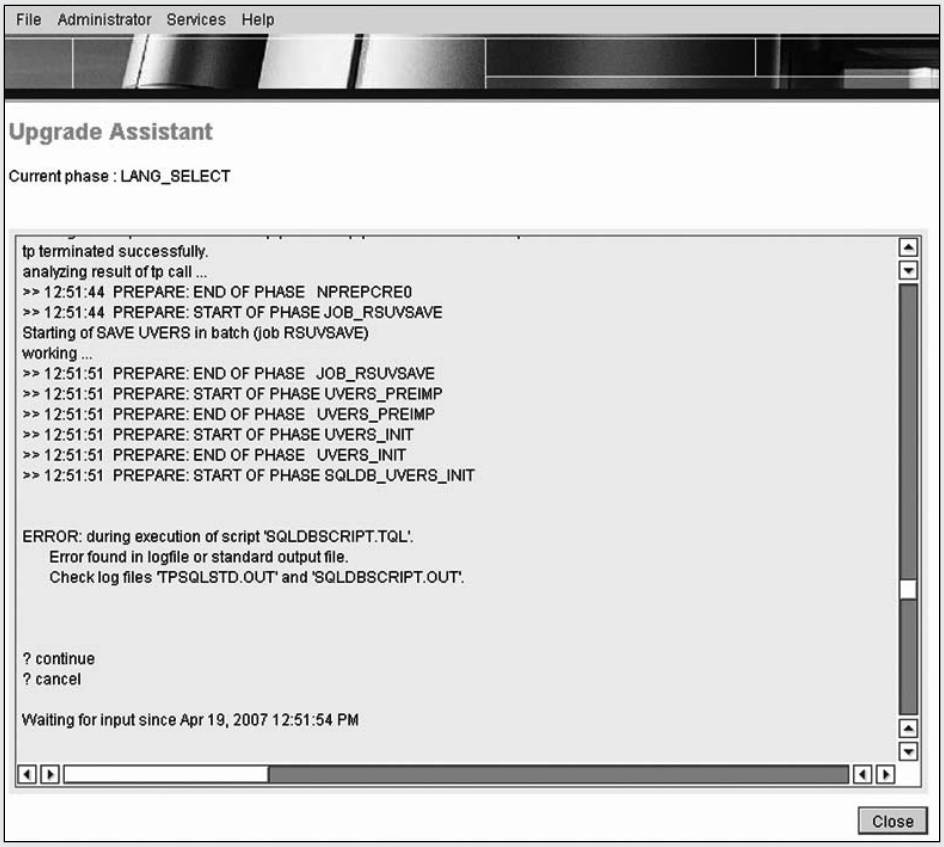

Figure 20 Console Service

Click on the Close button to return to the input screen.

> ***Note!***
>
> The complete upgrade dialog is also saved in the logfile /usr/sap/put/log/UpgDialog.log. Always keep this file after an upgrade because it contains information that will be very useful for your next upgrades.

The Notes Service

The Notes Service function starts a browser on your PC and brings up a page for searching SAP Notes in the SAP Service Marketplace (see **Figure 21**). As you can see, this is not the usual page for searching notes (http://service.sap.com/NOTES).

Providing the ability to search SAP Notes directly from the GUI is a nice idea, but, in practice, there is just one tiny problem: It doesn't work. No matter what you enter in the search fields, the search result is always "no notes found." This is a bit silly but can hardly be called a major irritant. Just open a separate browser and go to the normal notes search page.

SAPup command options

The principal function of SAPup is to run in the background and to control the ABAP PREPARE and the ABAP upgrade. In addition, SAPup also offers a command interface, which you can use to perform certain special operations.

You can invoke the command mode of SAPup in two ways:

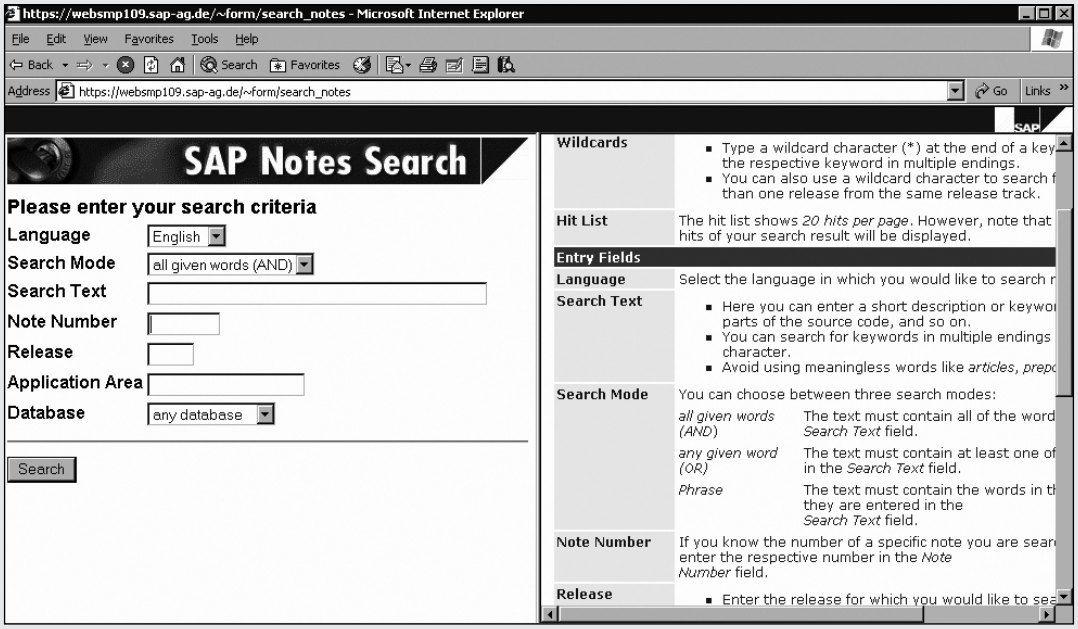

Figure 21 Notes Service

- On the upgrade host via the command line
- From inside the UA GUI

To see the available functions and command options, type the following command:

```
/usr/sap/put/bin/SAPup –h
```

SAPup will prompt you for the path to the upgrade directory (press Enter if the default path shown is correct).

Most of the functions are not intended for your direct use. They are either called internally during the upgrade process or needed only in exceptional circumstances. The functions that you will need most often are the ones that allow you to change parameters of the upgrade and those that control the shadow instance. These and some other useful functions of SAPup are described next.

Specifying the upgrade directory

If you run SAPup from the command line, SAPup will prompt you for the path to the upgrade directory. To avoid this extra prompt, use the upgdir option. Example:

```
SAPup -h upgdir=/usr/sap/put
```

Show the SAPup version

This command displays the version. Example:

```
> ./SAPup -V upgdir=/usr/sap/put
This is SAPup version 7.00/2,
build 24.081
```

Start and stop the shadow instance

Use the startshd and stopshd functions to start and stop the shadow instance. This is normally not necessary because SAPup takes care of starting and stopping the shadow instance at the right moment. However, it is sometimes necessary to do a manual stop/start, for instance, if you have to change profile parameters of the shadow instance.

Commands:

```
SAPup [upgdir=<upgrade_dir>] startshd
SAPup [upgdir=<upgrade_dir>] stopshd
```

Unlock and lock the shadow instance

Like the main instance during downtime, the shadow instance normally runs in upgrade lock mode, meaning that logging on to the instance is only possible for the users SAP* and DDIC, and that ABAP Workbench objects cannot be changed. At the beginning of the modification adjustment (Transaction SPDD), SAPup will unlock the shadow instance automatically. When activation begins after you have finished SPDD, SAPup also locks the shadow instance again.

If, however, as is often the case, the activation phase reports errors for some dictionary objects, then you must manually unlock the shadow instance to be able to log on as a user with development authority and to modify objects in the ABAP Workbench. When you are finished with correcting the activation errors, lock the shadow instance manually before resuming the upgrade.

Commands:

```
SAPup [upgdir=<upgrade_dir>] unlockshd
SAPup [upgdir=<upgrade_dir>] lockshd
```

Change parameters of the upgrade

Early on in the PREPARE process, you will be prompted for all the parameters that the upgrade process needs, such as host names, locations of the upgrade media, and so on. These parameters are then used throughout the entire upgrade, and normally you will not change them again. If you do need to change a parameter later in the upgrade (for example, because you had to move the upgrade DVDs to a different disk or file system), then use the following function:

```
SAPup [upgdir=<upgrade_dir>] set stdpar
```


As always, you can execute this command from inside the UA GUI or directly via the command line of the OS. In this case, calling the command in the UA GUI is more convenient because you are presented with the same graphical input windows as during PREPARE.

Change parameters of the shadow instance

PREPARE also prompts you for the shadow instance parameters (instance number, port numbers, etc.). Again, you can change these parameters later if necessary by using the function:

```
SAPup [upgdir=<upgrade_dir>] set shdpar
```

Change parameters for the upgrade strategy and parallel processing

SAPup (not PREPARE) prompts you for the upgrade strategy you want to use (downtime-minimized or resource-minimized) as well as for some other parameters that affect the resource usage by the upgrade:

- Allotted time for the database import
- Upgrade phase from where database archiving is to be disabled
- Number of parallel batch processes

If you need to change these parameters, you can do so with the function:

```
./SAPup upgdir=/usr/sap/put set rswpar
```

Change the DDIC password

During the initial parameter input, PREPARE prompts for the password of user DDIC in client 000. DDIC in 000 is used throughout the entire upgrade for all phases that need a connection to the SAP system. It is good practice to set the DDIC password at the beginning of the upgrade and leave it unchanged until the end, but this is not always possible. The password might expire during the upgrade, or you might need to change it for any other unexpected reason.

If you alter the DDIC password, then you must inform the upgrade process of this with the following SAPup function:

```
./SAPup upgdir=/usr/sap/put set ddicpwd
```

Note that this command by itself does not change the password (you need Transaction SU01 for that); it simply informs SAPup of the password change.

Change the DDIC password in the shadow instance

In the shadow instance, which has its own user configuration, user DDIC in client 000 is created with the same password as in the source system. If you change the DDIC password in the shadow instance for whatever reason, then you must again make SAPup aware of this:

```
./SAPup upgdir=/usr/sap/put set shdddicpwd
```

(Be careful: It’s three d’s in a row!)

Resetting PREPARE

Situations may arise — hopefully not too often — in which you decide that you want to abandon the active PREPARE altogether and start over. What could also happen is that you run PREPARE in a system but then decide not to upgrade that system after all. Leaving traces of this unfinished business in the system is not a good idea because it would prevent upgrading the system in the future. To reset PREPARE and thus erase any trace of the attempted upgrade, use the command:

```
./SAPup upgdir=/usr/sap/put reset prepare
```

Other functions

SAPup -h will list numerous other functions than the ones described in the previous paragraphs. You

normally do not need these functions and unless SAP instructs you to use one of them, it is best to keep to the straight and narrow. Don't start experimenting with them in a live upgrade!

Tips and tricks

We end this section with two useful tips:

- How to make UA use different network ports
- How to change the administrator password

Change the port numbers for the Upgrade Assistant

UA uses a total of four ports for all its communication. By default, these ports are 4238, 4239, 4240, and 4241. These numbers are defined in the configuration file /usr/sap/put/ua/UaServer.properties:

```
UA_R3UP_PORT     = 4240
UA_GUI_PORT      = 4241
UA_MONITOR_PORT  = 4238
(...)
UA_HTTP_PORT     = 4239
```

The first three entries are grouped together near the beginning of the file. The UA_HTTP_PORT entry can be found farther down in the section for the HTTP server settings.

Normally there is no reason to change the port numbers, but there are situations where this becomes necessary. One or more of the ports might be in use by another application or access to these ports might be blocked by a firewall. If you run several ABAP upgrades simultaneously on the same host, then you will have to change the port numbers for all but one of these upgrades; otherwise, the respective UA Servers will attempt to open the same ports, which results in an error stop.

Before you assign new port numbers, make sure that these new numbers will be usable:

- The ports must not be blocked by the firewall.
- The ports must not be used by other applications. To check whether it is currently in use, type the following command.

  UNIX/Linux:

  ```
  netstat -an | grep <portnr>
  ```

  Windows:

  ```
  netstat -an | findstr <portnr>
  ```

  In the following example, you want to use the port 4339, but this port turns out to be in use. The entry *.4339 indicates that some local program is listening on the port (see **Figure 22**). The other entries show the IP address and port of the host in the first column, and the IP address and connected port of all the clients currently using this port.

You must also check whether the ports have not been assigned for use by other applications, even if those applications are not active at present. You do this by looking for an entry with this port number in the services file on the host. The name of this file is /etc/services (UNIX/Linux) or C:\WINDOWS\SYSTEM32\DRIVERS\ETC\SERVICES (Windows).

> ***Note!***
>
> Depending on the version of Windows, the top directory might be called WINDOWS or WINNT.

The entry in the services file in **Figure 23** indicates that port 4339 is destined for use by another (in this case non-SAP) application.

After choosing a new set of port numbers and ascertaining that these ports can be used for UA, you can configure the new ports:

1. Stop the UA Server if it is still running.

2. Make a backup of the current properties file, for example:

```
cd /usr/sap/put/ua
mv UaServer.properties
   UaServer.properties.default
```

3. Use a text editor to change the four port entries in the UaServer.properties file. Let's suppose that you chose to use ports 8238 to 8241:

```
UA_R3UP_PORT     = 8240
UA_GUI_PORT      = 8241
UA_MONITOR_PORT  = 8238
(...)
UA_HTTP_PORT     = 8239
```

4. Restart the UA Server, and wait for the "Ready" message.

5. Call the main upgrade page, using the new port number 8239:

```
http://hostname:8239
```

6. Start the UA GUI, and specify the new port number on the logon screen.

Set a new administrator password

As explained earlier, you specify the password for the administrator role in UA when you first start the UA Server. If you intend to use the observer role, then you set its password in the UA GUI.

To change the administrator and observer passwords, you also use the UA GUI, and you must obviously be logged on as the administrator.

But what if you forgot the administrator password, and you do not have an open UA GUI administrator session? Stopping and restarting the UA Server won't help you; the UA Server will not prompt for the password again. Looking up the answer in the SAP Upgrade Guides or in the SAP Notes won't do you any good either; both are silent on the subject. Fortunately, the solution is not difficult:

1. If the UA Server is currently running, then terminate it with OS means, such as the kill command in UNIX or the End Process function in the Windows Task Manager, at the first convenient moment (preferably not in the middle of a critical or long-running upgrade phase).

2. Remove the file /usr/sap/put/UaState.

```
$ netstat -an | grep 4339
10.13.194.79.4339          *.*                    0      0 49152      0 LISTEN
10.13.194.79.4339    10.13.194.91.49679   49152      0 49152      0 ESTABLISHED
10.13.194.79.4339    10.13.194.96.49644   49152      0 49152      0 ESTABLISHED
```

Figure 22 Network port in use

```
mmecomm         4339/tcp                        # MMECOMMS server
```

Figure 23 Port reserved in services file

3. Remove the file /usr/sap/put/ua/ks (the "keystore").
4. Start the UA Server, which will now prompt for the administrator password like it did the first time you ran it.

Note!

Depending on the UA release, you might also need to delete the following files in the UA subdirectory:

- .sdt_storage
- .sdt_keystore

The UA Server prompts for the password if it does not find the file ua/UaState. If you only remove this file but leave the keystore untouched, then the next start of the UA Server will prompt for the password but will then abort with an exception stack trace that includes the message:

```
java.io.IOException
Keystore was tampered with, or password
was incorrect
```

Installation of the upgrade tools: Java

Compared with the ABAP side, installing and operating the upgrade tools for Java is a pretty simple affair. This is not necessarily an advantage because the interface for Java also offers fewer facilities than its ABAP counterpart, UA. As we mentioned earlier, the Java SDT has no notion of different user roles (administrator/observer) and does not allow more than one GUI to be connected at any one time. These are characteristics the upgrade GUI for Java shares with its very close relative, the SAPINST installation utility.

Before you start, you must have downloaded the Java fix buffer and extracted the fix to the Java upgrade directory. The next step is to extract the tools to the upgrade directory. Like on the ABAP side, this is done with a script located on the Java Upgrade Master DVD (for UNIX/Linux and Windows; for IBM iSeries, see the instructions in the SAP Upgrade Guide for that platform).

Note!

SAP will continuously correct problems in the upgrade programs and control files that were discovered after the initial release of the upgrade media. These corrections are made available in the form of fix buffers available from http://service.sap.com/PATCHES. For information regarding the appropriate fix buffer to download, see SAP Note 813658.

Initial PREPARE run: UNIX and Linux

In this example, we assume that you have placed disk copies of the upgrade media in a server directory /sapmedia. You have placed the Java Upgrade Master DVD in a subdirectory named /sapmedia/UPGMSTR_JAVA. The Java Upgrade Master contains the installation script for the Java upgrade tools.

For the Java upgrade directory, we assume that you use the default path name /usr/sap/jupgrade.

Follow these steps:

1. Log on as user <sid>adm.
2. Switch to the upgrade directory:

```
cd /usr/sap/jupgrade
```


3. Type the command:

```
/sapmedia/UPGMSTR_JAVA/JPREPARE
```

   If the name of the Java upgrade directory is not /usr/sap/jupgrade, then you must specify the path via a command-line argument, for example:

```
/sapmedia/UPGMSTR_JAVA/JPREPARE
 /usr/sap/jupgPRD
```

   Note that here you simply specify the directory path as the command argument; you do not add the upgdir= keyword as with ABAP.

   If the upgrade directory does not exist, then JPREPARE will create it.

4. If the fix buffer was already placed in the upgrade directory, then JPREPARE will also unpack the UPG file. This is a minor difference with ABAP, where the buffer is extracted during PREPARE and not by the initial script. If JPREPARE creates the upgrade directory, or the fix buffer is not yet present, then the PREPARE process will take care of the fix buffer later; appropriate screens will appear in the SDT GUI.

5. JPREPARE then unpacks the upgrade files to /usr/sap/jupgrade and finally displays the message:

```
Waiting for SDTServer to connect on host
name <host>/<address> socket 6240 ...
```

6. At this point, you can start the GUI, but it is a good idea not to let the process run in the foreground. To free up the terminal window and start a background process, break with Ctrl+C and then restart as follows (as <sid>adm):

```
cd /usr/sap/jupgrade/exe
nohup ./PREPARE &
```

   Note that now the name is just PREPARE, not JPREPARE.

Initial PREPARE run: Windows

For the Windows example, we assume that the upgrade media are in the folder D:\SAPMEDIA, with the Upgrade Master in the subfolder UPGMSTR_JAVA.

1. Go to the Upgrade Master folder, and start the script PREPARE.BAT:

```
cd /D D:\SAPMEDIA\UPGMSTR_JAVA
jprepare
```

   This assumes that you use the default path for the Java upgrade directory. This default path is \usr\sap\jupgrade on the drive pointed at by the SAPLOC share. If you want to use a different path name and/or a different drive for the upgrade directory, then you must specify the path on the command line, for example:

```
jprepare F:\sapadmin\jupgrade
```

> ***Caution!***
>
> You cannot specify a path in UNC format (\\host\share\path).

2. If the upgrade directory does not exist, JPREPARE creates it. It then creates the subfolders (see **Figure 24** on the next page) and unpacks the upgrade tools.

3. Finally, JPREPARE launches the PREPARE process. At this point, you will see the following message:

```
call exe\jump -
cddir=D:\sapmedia\UPGMSTR_JAVA\\JUP -
 run prepare
Waiting for SDTServer to connect on host
name <server>/<ip> socket 6240
```

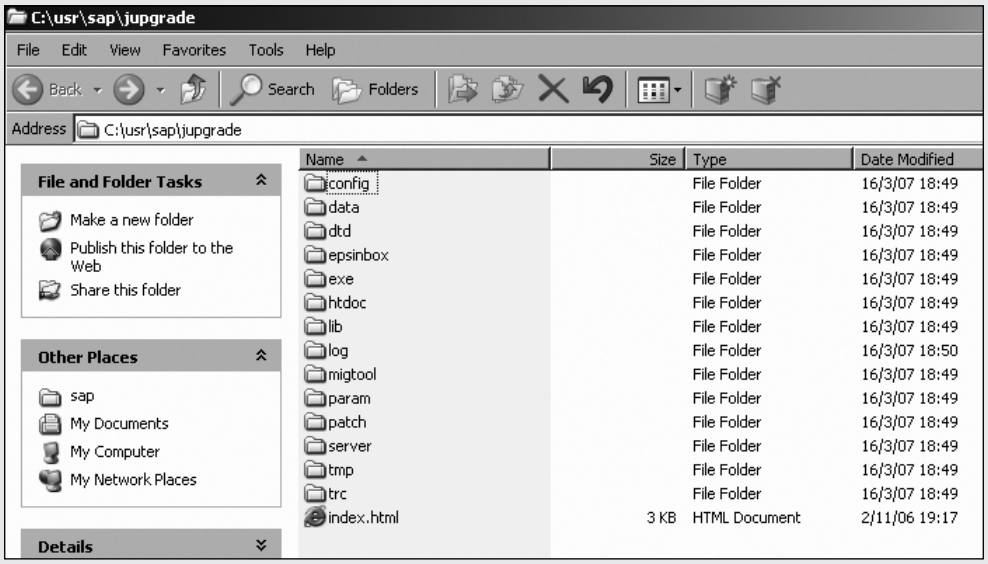

Figure 24 Java upgrade directory after initial JPREPARE run

4. If you already placed the fix buffer into the upgrade directory, then PREPARE will immediately unpack the UPG file:

```
Unpacking fix archive
 'FIX_J_NW04SSR2.UPG'...
```

5. If JPREPARE creates the upgrade directory, or the fix buffer is not yet present, then the PREPARE process will take care of the fix buffer later; appropriate screens will appear in the SDT GUI.

6. Minimize but do not close the command window where PREPARE is waiting. You are now ready to start the GUI.

7. If you have to restart the PREPARE process later, then proceed as follows:

```
cd \usr\sap\jupgrade\exe
prepare
```

   There is no need to specify a path here even if you are not using the default path name for the upgrade directory.

Using the Java SDT GUI

Starting the Java upgrade involves an Internet browser and Java Web Start, but that is where the similarity with the ABAP UA ends. Let's see how it works.

Starting the SDT Server

You do not start the SDT Server explicitly. When you start the SDT GUI as described next, the server process also starts and connects to the waiting PREPARE process. At this point, the command window of PREPARE displays a "connected" message:

```
Waiting for SDTServer to connect on host
name <server>/<ip> socket 6240
 ... connected.
```

Starting the SDT GUI

To start the SDT GUI, follow these steps:

1. Open a browser on your PC.

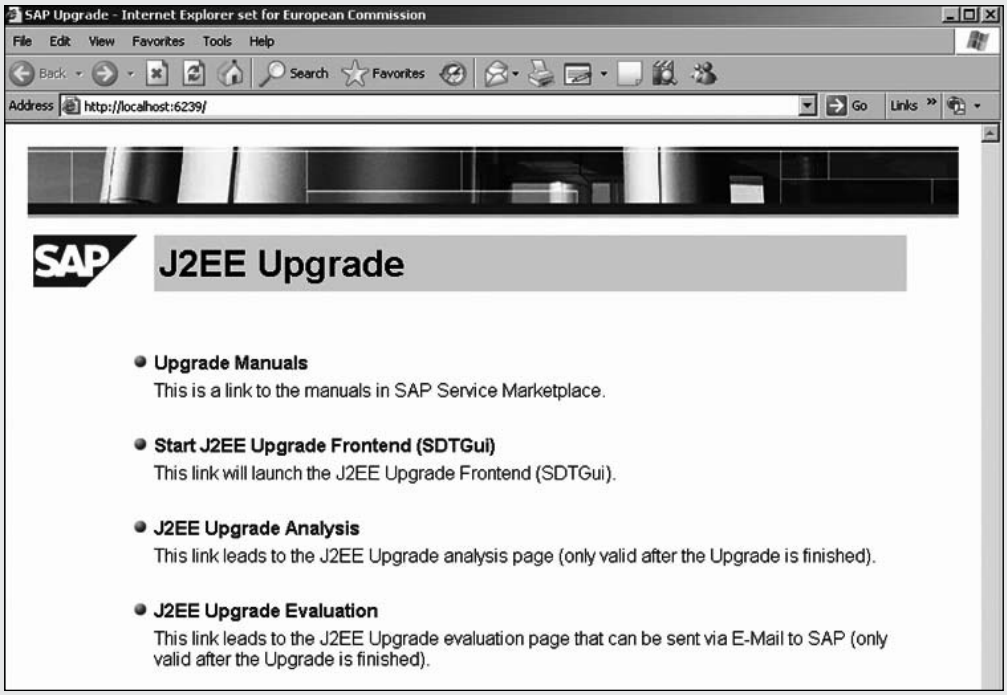

Figure 25 Java upgrade — main page

2. Type the URL http://<server>:6239 where <server> is the host name (or possibly the full domain name, depending on how networking is set up) of the host where the UA Server is running. Example:

   ```
   http://sapdev.mycomp.com:6239
   ```

3. If you decide to start the browser on the upgrade host itself rather than on a local PC, you may also use the URL:

   ```
   http://localhost:6239
   ```

4. The main Java Upgrade page now opens (see **Figure 25**).

5. The Java Web Start window pops up, and the application begins to load (see **Figure 26**).

6. See the earlier section “Java Web Start errors while loading the UA GUI” if Java Web Start fails to load and start the application.

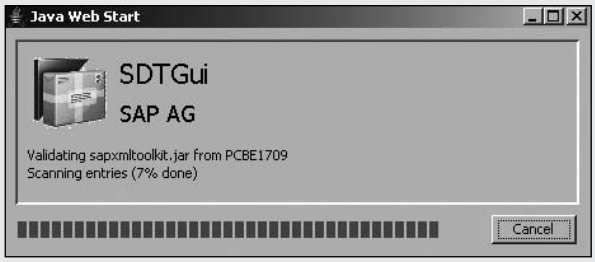

Figure 26 Java Web Start loading the SDT GUI

7. The Welcome screen of the SDT GUI appears (see **Figure 27** on the next page). Remember that the Java upgrade does not have the concept of administrator and observer users, so there is no logon screen.

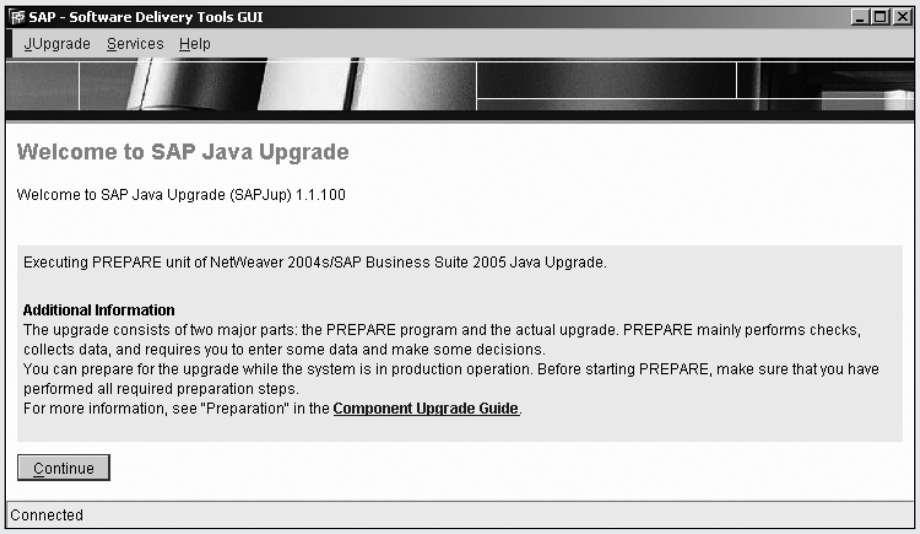

Figure 27 SDT GUI Welcome screen

Features of the SDT GUI

As we did for the ABAP UA GUI, let's again take a brief tour of the menu functions, of which there are only a few.

The JUpgrade menu:

- **JUpgrade → Exit:** Ends the SDT GUI session. Leaving the SDT GUI does not end the active PREPARE or upgrade process, so you can safely do this.

The Services menu:

- **Services → File Service:** Lets you select and display files on the upgrade host. This is normally used to display upgrade logs. This function is identical to the File Service in the UA GUI. See the earlier section "The File Service" for a description.
- **Services → Log Service:** This service provides immediate access to the Java upgrade logs. The left pane lists the existing log and error files in /usr/sap/jupgrade/log. By clicking on a file, its contents are displayed in the right-hand pane. **Figure 28** shows the log service for a normal text log, which is displayed in free-format fashion.

  Many of the upgrade logs, including the main SAPJup.LOG, are displayed in a more structured, tabular form showing the severity, date and time, message text, and call location. You can use the arrows in the toolbar at the top of the screen to move through the file. You can also filter messages by choosing the minimum severity level, as shown in **Figure 29**.

The Help menu:

- **Help → About:** Shows the version of the Java Upgrade GUI and the copyright notice.

Tips and tricks

Like on the ABAP side, you can change the network ports used for the Java upgrade if necessary. The default ports used by the Java upgrade are 6239 (for HTTP connections from the browser) and 6241 (for the SDT GUI). You find these port numbers in the file:

```
/usr/sap/jupgrade/server/sdtserver.xml
```


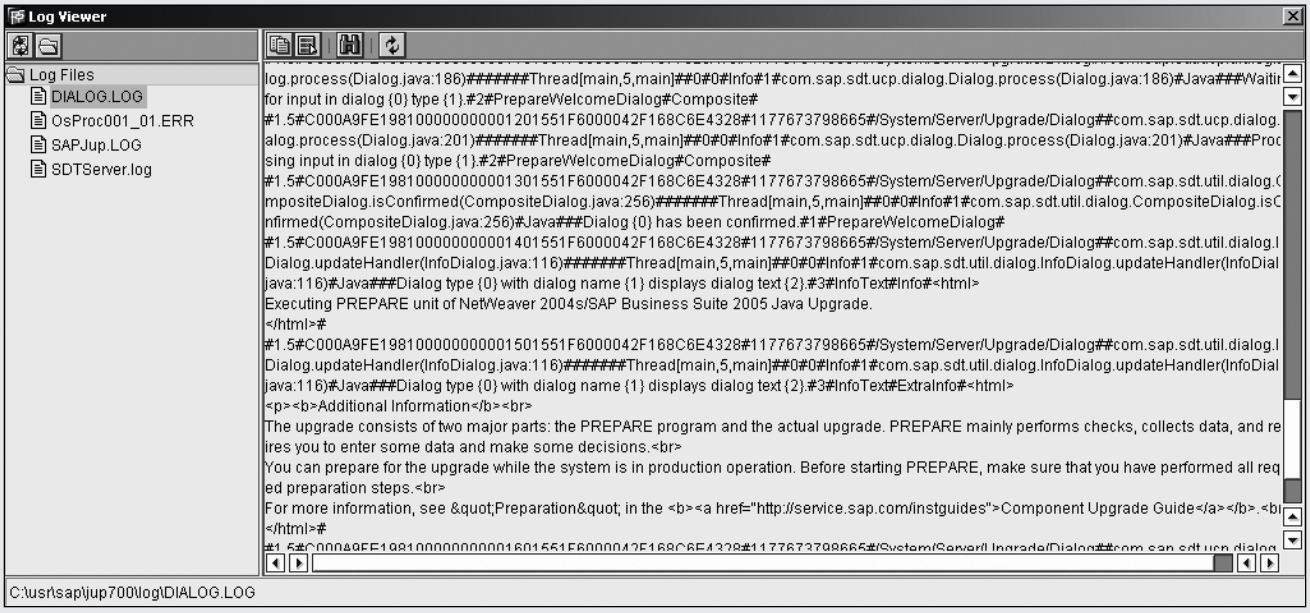

Figure 28 SDT GUI Log Service (free-format display)

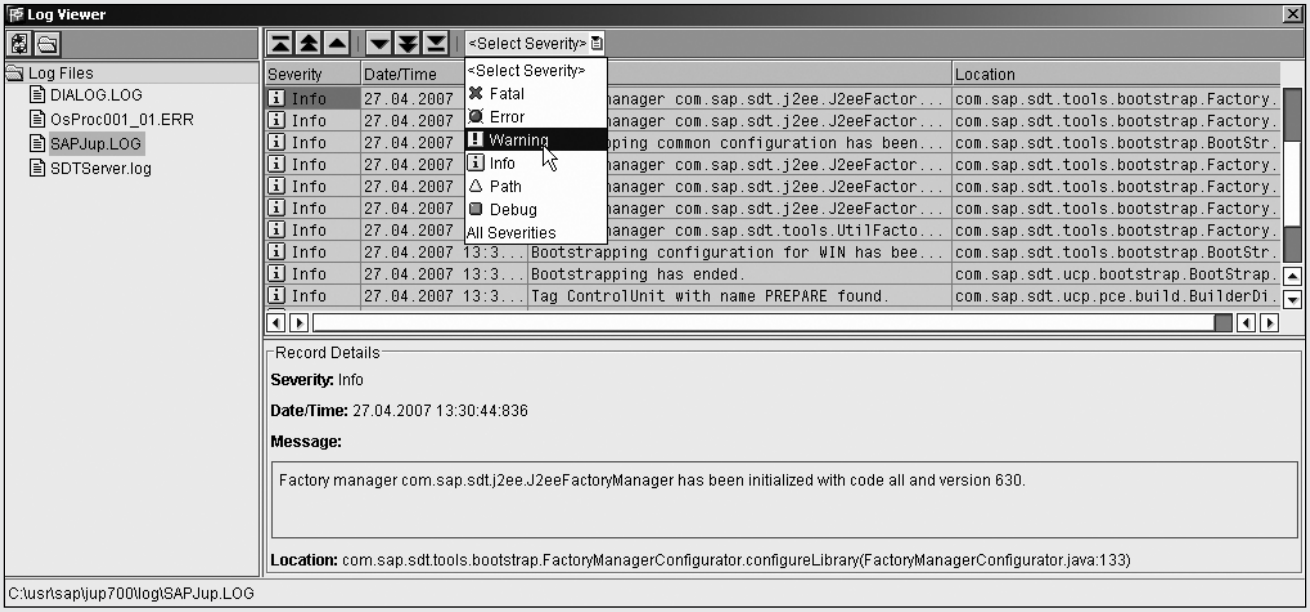

Figure 29 SDT GUI Log Service (structured display)

Note that here we are not dealing with a simple text file but with an XML file. To edit the file, you could use a specialized XML editor, but a standard text editor such as Notepad or vi will do just as well. Simply search for the strings "6239" and "6241"; each appears only once in the file, with the respective tags <HTTPPort> and <GuiPort>.

The same warnings and restrictions apply as for the ABAP UA:

- Don't change the port numbers unless you have to.
- Make sure the new port numbers are not already used by another application.
- Also make sure the server ports are accessible (not blocked by a firewall) from the workstation you intend to use during the upgrade.

See the earlier section "Change the port numbers for the Upgrade Assistant" for instructions on how to check these requirements.

Ready to go

Our hope is that you now have a solid foundational knowledge of the SAP upgrade tools that are available to you — what they are; how to start, stop, and control them; and how to monitor their activities. With this knowledge, you are now ready to attack the upgrade at hand.

Bert Vanstechelman works as an independent SAP Basis Consultant and has approximately 12 years of SAP experience. His most recent long-term assignments have been in SAP Basis consulting roles running all kinds of SAP versions in combination with all possible databases and operating systems supported by SAP. Bert specializes in SAP administration, upgrades, installations, operating system and database migrations, and Unicode conversions. Bert is the author of the SAP Essentials the SAP OS/DB Migration Project Guide and the mySAP ERP Upgrade Project Guide, both published by SAP PRESS. He is a frequent contributor to SAP Professional Journal and panel expert for SAP Release Upgrades and OS/DB migrations on SearchSAP.com's Ask the Expert feature. Bert can be reached at bert@logosconsulting.be.

Mark Mergaerts is Principal Technology Consultant at SAP Belgium and has more than 12 years experience with SAP Basis. His consulting activities, mainly with large SAP accounts, concentrate on the key areas of system administration, database management, upgrades and installations, performance and workload analysis, and OS/DB and Unicode migrations. He also teaches advanced database administration, upgrade, and SAP performance classes to an international audience. Mark can be reached at mark.mergaerts@sap.com.

After seven years of R/2, Dirk Matthys started with R/3 back in 1997. Dirk is a Senior ABAP developer working for one of the biggest Belgian Metal Transformation companies. In the past few years, his main activities have diverted to the twilight zone between the SAP technical and application world. He is responsible for the coordination of SAP Basis application change projects, such as upgrades, migrations, mergers, splits, and the installation of support packages. Dirk can be reached at dirk.matthys@bekaert.com.

