SIEMENS

Teamcenter 11.2 Systems Engineering and Requirements Management

Systems Architect/ Requirements Management System Administrator's Manual

REQ00004 • U

Manual History

Manual Revision	Teamcenter Requirements Version	Publication Date
A	3.0.1	May 2003
B	4.0	December 2003
C	4.1	February 2004
D	5.0	July 2004
E	6.0	March 2005

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
F	2005	September 2005
G	2005 SR1	June 2006
H	2007	December 2006
I	2007.1	April 2007
J	2007.2	September 2007
K	2007.3	January 2008
L	8	January 2009
M	8.0.1	June 2009
N	8.1	October 2009
O	8.2	October 2010
P	9	July 2011
Q	9.1	May 2012
Q1	9.1.5	January 2014
R	10.0	January 2015

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
S	10.1	September 2016
T	11.1	March 2018
U	11.2	September 2019

This edition obsoletes all previous editions.

System Administration Contents

Preface

This manual is a system administrator's reference for Teamcenter Systems Architect/Requirements Management . Systems Architect/Requirements Management belongs to the Siemens PLM Software portfolio of digital product lifecycle management software and services.

Audience

This manual is for Systems Architect/Requirements Management system administrators. The manual provides both conceptual information and step-by-step instructions for specific tasks.

This manual assumes that you are familiar with the general system administration tasks of operating systems.

Conventions

This manual uses the conventions described in the following sections:

Revision Marks

Technical changes are marked by a bar adjacent to the changed text.

Names and Values

This manual represents system names, file names, and values in fonts that help you interpret the name or value. For example:

Change or add the parameter to the **init***sid***.ora** file.

The conventions are:

Bold	Bold font represents unvarying text or numbers within a name or value. Capitalization is as it appears.
Italic	Italic font represents text or numbers that vary within a name or value. The characters in italic text describe the entry. Letters are shown in lowercase, but the varying text may include uppercase letters. In **init***sid***.ora**, *sid* identifies a varying portion of the name (a unique system ID). For example, the name of the file might be:

	initBlue5.ora
text-text	A hyphen separates two words that describe a single entry.

Command Line Entries, File Contents, and Code

This manual represents command line input and output, the contents of system files, and computer code in fonts that help you understand how to enter text or to interpret displayed text. For example, the following line represents a command entry:

```
msqlora -u system/system-password
```

The conventions are:

`Monospace`	Monospace font represents text or numbers you enter on a command line, the computer's response, the contents of system files, and computer code. Capitalization and spacing are shown exactly as you must enter the characters or as the computer displays the characters.
`Italic`	Italic font represents text or numbers that vary. The words in italic text describe the entry. The words are shown in lowercase letters, but the varying text may include uppercase letters. When entering text, use the case required by the system. For the preceding example, you might substitute the following for *system-password*: `KLH3b`
`text-text`	A hyphen separates two words that describe a single entry.

Submitting Comments

Portions of Teamcenter software are provided by third-party vendors. Special agreements with these vendors require Siemens PLM Software to handle all problem reports concerning the software they provide. Please submit all comments directly to Siemens PLM Software.

Please feel free to share with us your opinion on the usability of this manual, to suggest specific improvements, and to report errors. Mail your comments to:

Siemens PLM Software Technical Communications
5939 Rice Creek Parkway
Shoreview, MN 55126
U.S.A.

To submit an incident report, you can use the Siemens PLM Software GTAC online support tools at the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/

Proprietary and Restricted Rights Notice

Chapter 1: Introduction to System Administration

This chapter presents an overview of system administration in Systems Architect/Requirements Management.

A Systems Architect/Requirements Management system administrator has the following main responsibilities:

- Configuring Systems Architect/Requirements Management in a Web application server environment.
- Tuning the Systems Architect/Requirements Management database.
- Maintaining the Systems Architect/Requirements Management database in the back end.
- Managing the Systems Architect/Requirements Management license.

For information about Systems Architect/Requirements Management project administration, such as creating projects, maintaining project security, customizing objects and properties, and managing users, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

For information about installing Systems Architect/Requirements Management, see the *Systems Architect/Requirements Management Server Installation Manual*.

Systems Architect/Requirements Management Architecture

Designed for multitier Web deployment, the Systems Architect/Requirements Management architecture employs four logical tiers:

- The *client* tier, which presents the Systems Architect/Requirements Management user interface on a local computer.
- The *Web* tier, which handles the HTTP/HTTPS traffic between the client and enterprise tiers.
- The *enterprise* tier, where the Systems Architect/Requirements Management business logic resides.
- The *resource* tier, where the Versant object database resides.

Although the Web and enterprise tiers are logically separate, both tiers run in the same Java™ environment of the J2EE Application Server, or Web application server with servlet engine. Therefore, only three independent Systems Architect/Requirements Management components are deployed:

- Client workstation (where the client runs). For example, a Windows 10 computer.
- Web application server (Requires servlet engine only. Does not require or use EJB support.) For example, WebLogic, Tomcat.
- Database server (Versant database application).

In enterprise production deployments, the Web application server and database server typically run on different hosts. In smaller installations, both components can run on the same host. For standalone demonstration environments, all three components can be installed on one workstation or laptop computer.

The components in all tiers are multithreaded and are able to maximize performance on hardware with multiple CPUs.

Figure 1-1 shows two examples of multitier Web deployment.

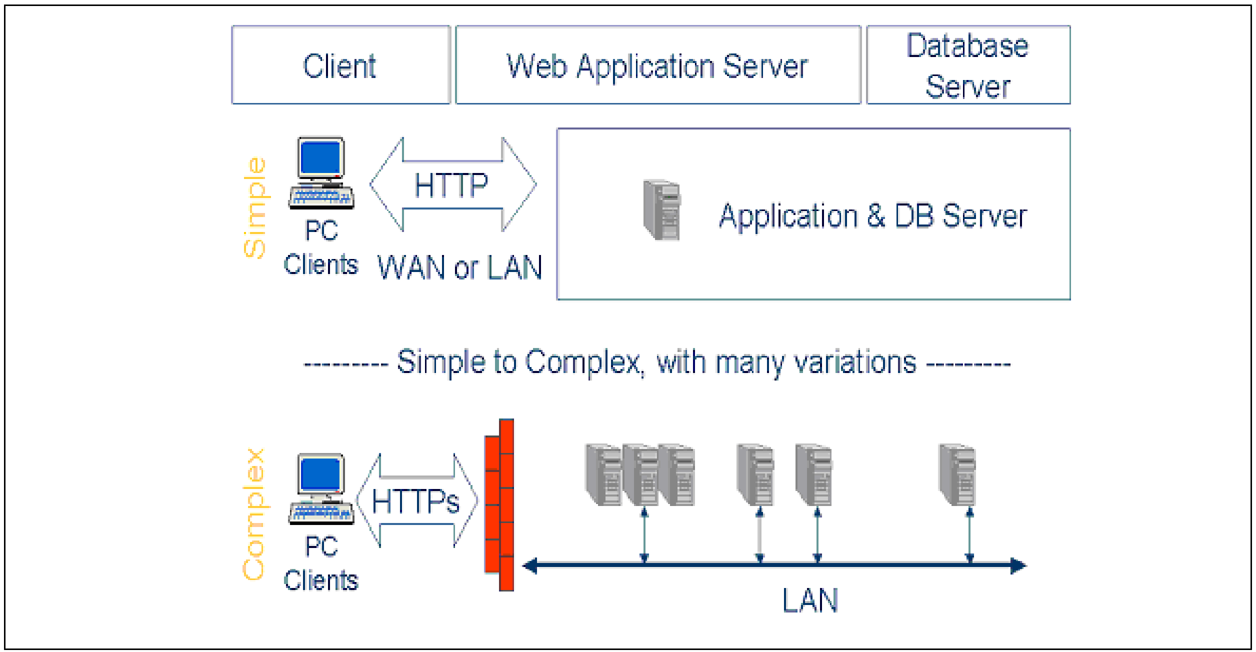

Figure 1-1. Systems Architect/Requirements Management Multitier Web Deployment

Systems Architect/Requirements Management Components

The Web application server and database server components are initially installed from the Systems Architect/Requirements Management installation package. The files installed on the Web application server include the client components. These client files are automatically downloaded to a user's workstation the first time the user logs in to Systems Architect/Requirements Management.

The components installed from the installation package fall into three categories:

- Client components, which are unique to Systems Architect/Requirements Management.
- Web components, which are common to other Teamcenter products.
- Database component, the Versant database server supplied by Siemens PLM Software.

Figure 1-2 shows the components included in the Systems Architect/Requirements Management installation package.

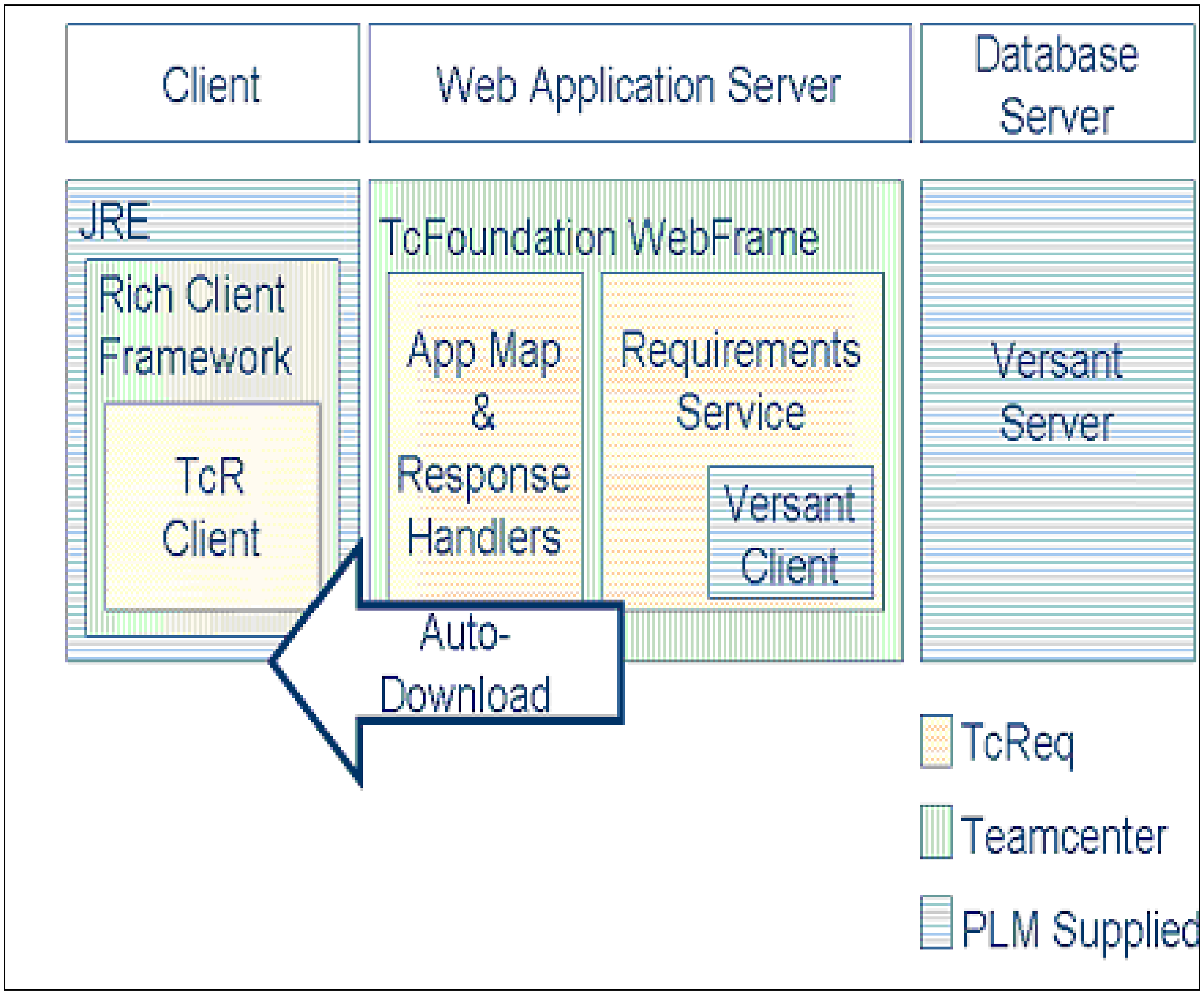

Figure 1-2. Components Included in the Systems Architect/Requirements Management Installation Package

Client Components

The Systems Architect/Requirements Management installation package contains the following client components:

- **Systems Architect/Requirements Management client**

  This component presents the Systems Architect/Requirements Management user interface on a local computer, and communicates with the Web tier via HTTP/HTTPS. This Java client is downloaded automatically from the Web application server when a user logs in to Systems Architect/Requirements Management the first time. It is also downloaded whenever a user logs in and a newer version is found at the server.

- **Rich Client Framework**

  The Rich Client Framework (RCF) is shared by certain other Teamcenter products. This Java client authoring framework and the Systems Architect/Requirements Management client are installed and run together as one application. At this time, there is no need for users or administrators to deal separately with RCF, but in the future more than one Teamcenter product may run within the same RCF instance and certain RCF customization points will be exposed.

- **Java Runtime Environment (JRE)**

  The Systems Architect/Requirements Management client requires a Java Runtime Environment. If a compatible version is already present on the client workstation, that existing JRE installation is used. Otherwise, the JRE version delivered with Systems Architect/Requirements Management is automatically downloaded to the client and installed.

Web Components

The Systems Architect/Requirements Management installation package contains the following Web components:

- **TcFoundation WebFrame**

  This is a model-view-controller Java environment that manages incoming HTTP/HTTPS requests and dispatches them according to the *application map*. *TcFoundation WebFrame* is also used by Teamcenter Enterprise, but at this time Systems Architect/Requirements Management installs and runs its own copy.

- **Application map and response handlers**

  The application map tells TcFoundation WebFrame which *response handlers* to execute for each incoming URL. The response handlers then communicate the incoming user request to the requirements service and format the results for return to the client via the HTTP/HTTPS response.

- **Requirements service**

  The Systems Architect/Requirements Management business logic resides in the *requirements service*. Incoming requests are verified against access rights of the user, actions are performed on objects residing in the database, and results assembled for return to the client. Each request to the requirements service is bounded by a database transaction that is rolled back if any part of the request fails to complete successfully.

- **Versant client**

  The requirements service is a client to the Versant database, managed through a pool of open database sessions. Each incoming request from a user is temporarily allocated one of those sessions to perform its work, and then releases the session. This pool avoids the overhead of establishing a new database connection for each user request, and avoids the database server load of maintaining a dedicated open connection for each user.

Database Server Component

The Versant database server is supplied with Systems Architect/Requirements Management. The database server can be installed on the same host as the Web application server, or another host within the same local area network segment. In large installations, more than one Web application server can connect to the same database server.

⚠ For best performance, avoid firewalls or routers between the database server and the Web application servers connected to it. Wide-area network segments between the database server and the Web application servers must never be used.

Chapter 2: Configuring Architect/Requirements

This chapter contains instructions for configuring Architect/Requirements in a Web server environment.

Configuring Architect/Requirements involves the following:

- Configuring for linking to other Teamcenter products
- Configuring for license
- Customizing the Architect/Requirements server
- Configuring for Security Services

Use the Configuration Web page to configure Architect/Requirements parameters. The remaining sections in this chapter provide details on the specific parameters to update.

Installing Architect/Requirements Client in Silent Mode

The Architect/Requirements client installation program provides an option to install the client in silent mode. The IT departments or enterprise administrators can use the silent mode option to roll out command line installations of Architect/Requirements Client with Office Integration on end-user machines.

- The installation must be performed as a user in the Power Users or Administrators group.
- The client installer does not check for the user privilege level. If the installation is not performed as a privileged user, the registry entries are not created and the client fails to run when launched by a normal user.

To install the Architect/Requirements client in silent mode:

1. After a successful installation of the TcSE server, the client installation program is located in the TcSE_Client_Installer sufolder on the TcSE server, which should be available via a Windows share referenced as follows:

   \\<TcSE-Windows-server>\TcSE_Client_Installer .

   For details, please refer to the post installation task *Adding a Shared Folder for the TcSE Client Installer* in Chapter 4.

2. From each Windows 10 client sytem, run the following command:

   ```
   \\<TcSE-Windows-server>\TcSE_Client_Installer\setup.exe –i silent LAX_VM
   "%TCR_JRE32_HOME%\bin\java.exe"
   ```

   If you install the 32-bit Java 8 JRE using Oracle's installer, the the last two arguments (*LAX_VM "%TCR_JRE32_HOME%\bin\java.exe"*) are not required. Otherwise, without these argurments, the installer will fail to locate a Java 8 JRE on its own.

The client is installed in the default folder, "*C:\Program Files\SiemensPLM\Teamcenter\SystemsEngineering\Release_<version>*" .

Using the Configuration Web Page

A Configuration Web page provides a graphical user interface for you to configure server parameters. This section describes how to access the Configuration Web page and explains how to use it to configure Architect/Requirements parameters.

To access the Configuration Web page:

1. In Internet Explorer, open the Architect/Requirements home page, and then click the **Administrative Tools** link.

   Depending on whether Security Services is enabled on the Architect/Requirements server, one of two login pages appears:

   - The Architect/Requirements login page appears if Security Services is not enabled.
   - The Teamcenter Login page appears if Security Services is enabled.

2. Do one of the following:

   - On the Architect/Requirements login page, enter your Architect/Requirements user name and password, select a language, and click **Log In**.
   - On the Teamcenter Login page, enter your Security Services user name and password, select a language, and click **Log In**.

3. From the Administrative Tools page, select **Web Application Configuration**.

   This brings you to the following URL, which can also be accessed directly:

   `http://.../tcr/ugs/tc/req/configtcr.jsp`

   For example:

   http://*myhost*:8080/tcr/ugs/tc/req/configtcr.jsp

   The Architect/Requirements Configuration Web page is displayed.

To enter information in the Configuration Web page:

1. Under **Parameter Name**, select the parameter you want to update.

   If a parameter value exists in the database, it appears in bold in this column.

   The **Description** column provides a description of the corresponding parameter.

2. Under **Current Database Value**, enter the proposed value for the parameter.

   If a parameter value exists in the database, that value appears in this column.

3. After you make all your updates, you have the following choices:

   For information about installing the Architect/Requirements security files, see the *Systems Architect/Requirements Management Server Installation Manual*.

   - To reset the value of a parameter that has been changed to its default value, before clicking the **Update** button, click **Clear**.

     The default value appears in the **Default Value** column.

   - To reset the value of selected parameters to their default values, click **Reset to Default**.
   - To update your changes and save them to the database, click **Update**.
   - To cancel your changes, click **Back**.

4. Click **Update**.

   The Confirm Deployment Descriptor Update page is displayed, with the proposed changes in a table.

5. From the Confirm Deployment Descriptor Update page, click **OK** to accept the changes and update the database.

   If you need to change the information, click **Back** to return to the Configuration Web page.

6. Restart your Web server after modifying a value.

Configuring the Office Live Interface and Teamcenter Linking

To support Office Live interface configuration and object linking between Architect/Requirements and other Teamcenter products, modify the following parameters using the Web Application Configuration page (**http://.../tcr/ugs/tc/req/configtcr.jsp**):

If there is a problem, run the Office Live diagnostic utility, which is located at **Systems Architect/Requirements Management Home Page|Administrative Tools|Diagnostic Tools|Live Office diagnosis**.

WOLF.AppIP (Office Live Interface and Teamcenter Linking)

The application address, used when registering with the application registry service.

Change to the DNS name or IP address of the server where the Architect/Requirements Web application is running. For example:

```
server1.ugs.com
```

WOLF.AppPort (Office Live Interface and Teamcenter Linking)

The application port, used when registering with the application registry service.

Change to the port where the Architect/Requirements Web application is running. For example:

```
8080
```

WOLF.AppProtocol (Office Live Interface and Teamcenter Linking)

The Architect/Requirements Web application protocol, used when registering with the application registry service.

The value is **http** unless you are running Architect/Requirements under Security Services, in which case the value may be **https**.

WOLF.AppRegistryURL (Teamcenter Linking only)

The URL of the application registry service.

WOLF.AppGUID (Teamcenter Linking only)

The Architect/Requirements unique global unique identifier (GUID) for the Architect/Requirements database being registered. Enter a new value. The GUID must be unique.

WOLF.ChooserURL (Teamcenter Linking)

URL for the WOLF object chooser. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.IconURL (Teamcenter Linking)

URL for an icon that represents the Architect/Requirements application. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.LauncherURL (Teamcenter Linking)

URL that identifies the Architect/Requirements Object Launcher. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.LinkHandlerURL (Teamcenter Linking)

URL to transfers the control to Architect/Requirements link handler during WOLF linking. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.SoapURL (Teamcenter Linking)

Web Service URL for SOAP communication in PROXY linking. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

Configuring License Information

When you register your Architect/Requirements application, you receive the following from Siemens PLM Software Customer Support:

- Your Architect/Requirements license file, named **tcr.lic**.
- Your Architect/Requirements customer number.

To configure a license from the Configuration Web page:

-

  Locate the **LIC. Customer Number** parameter and enter your customer number in the corresponding **Current Database Value** box. You are given a customer number with your Architect/Requirements license.

  For more information, see Managing Licenses.

Customizing the Architect/Requirements Server

You can customize the Architect/Requirements server by modifying the following parameters on the Web Application Configuration page (**http://.../tcr/ugs/tc/req/configtcr.jsp**):

DB.LockTimeOut

The length of time (in minutes) during which reservations on requirement objects are held after a timeout occurs before being released. The first user to open a requirement for editing obtains a reservation, or lock, on that requirement. No other user can change the requirement until the reservation is released. The existing reservation is renewed each time the user saves his or her work, extending the time the object is held.

The reservation is released when the user closes the requirement or when the HTTP session is terminated.

If your HTTP session terminates (for example, if the client session times out) your reservation is lost. Another user can open the requirement for editing during this time. If you then try to edit the requirement, the client session reestablishes a connection with the database and sets a new reservation if another user has not opened the requirement for editing.

If the Web server is not aware that your HTTP session has terminated (for example, if the Web server crashes), the reservation on the requirement is released at the end of the lock timeout period, allowing a user to access the requirement.

If your system is configured so that multiple Web servers access a single database, the lock timeout period is checked as soon as a second user (on a different Web server from the first user) tries to access the requirement. This can cause the first user to lose his or her reservation. For such a system configuration, Siemens PLM Software recommends that you set a higher lock timeout period (for example, 60 minutes) to avoid having multiple users accessing the requirement.

You can change the reservation duration in the **Changed to Value** field for this parameter.

DB.MaxSession

The maximum connections to the database. You must restart the Web server after changing this parameter.

DB.MaxSessionAge and DB.MaxSessionSize

Architect/Requirements uses a session pool to manage Versant (database) sessions. These sessions are used by Versant for any database operations. Sometimes, the sessions may grow large and continue to hold the resources, such as memory, even after they are released to the session pool. In such cases, it is better to close the sessions to release the resources they hold to the system. Closing Versant sessions has no effect on the active user sessions.

Two Web Application Configuration parameters enable you to control how and when these sessions are closed. The parameters are:

- **DB.MaxSessionAge** specifies the maximum time in hours that a Versant database session may exist before it is closed. The default value for this parameter is 4. Any unused Versant session is closed by the system if it is found to be more than 4 hours old. You should avoid using a value less than 20 because it may lead to premature session closure and extra overhead for starting new sessions. If you do configure a value less than 20, then monitor session closures in the Architect/Requirements server log. A value of 0 (zero) turns off this behavior, and the session is not closed, irrespective of how old it gets.
- **DB.MaxSessionSize** specifies the maximum size in mega bytes (MB) that a Versant database session may grow to before it is closed. The default value for this parameter is 20. Any unused Versant session that has grown to be more than 20 MB is closed. A value of 0 (zero) turns off this behavior, and the session is not closed irrespective of how big it grows.

The following general formula is suggested for adjusting the **DB.MaxSessionSize** parameter for your site. Divide the total cache size by the maximum number of sessions, with a reservation for especially large actions.

Let us assume:

C is the memory configured for the Versant front end object cache. This is controlled by the Versant tuning parameter **swap_threshold**. The default value is 128 MB, which is sufficient for basic installations. Higher values may be recommended for high user counts and 64-bit servers. **swap_threshold** can be modified in the Versant front end profile.

R is the reserved memory for large transactions (use 100 MB for this value).

N is the number of Versant sessions (the likely maximum concurrent server load, typically 10 to 20% of the number of logged in users).

Then, **DB.MaxSessionSize** = (C minus R) divided by N

For example, let the Versant sessions have 500 MB to work with, and let the server typically need 20 concurrent sessions. This example uses 100 MB for the reserved memory.

Then, **DB.MaxSessionSize** = (500-100)/20 = 20 MB (which is the default setting).

If your calculation yields a value significantly less than 20 MB, you may be at risk of poor performance by expecting to service too many active users with insufficient memory.

DB.UndoLevels

The number of undo levels.

DB.RecycleBinTimeout

Determines whether a user's recycle bin is automatically emptied when the user logs out. This parameter is also used to specify how long a deleted item can remain in the recycle bin before being automatically deleted. Set this value to **0** to enable the automatic emptying of the recycle bin. Set this value to **-1** to disable it. Setting this value to a positive integer indicates the number of days after which the items in the recycle bin are automatically deleted.

DB.FastNumberRefresh

Determines the speed of the operations that modify a hierarchy. **DB.FastNumberRefresh** can be set to **True** or **False**. Operations that modify a hierarchy are faster if this parameter is set to **True**. In some cases, the number property displayed for an object may not be updated immediately. The number properties are always updated on setting this parameter to **False** but the operation takes longer.

Project.Cleanup.Now

Determines whether the database space should be recovered upon deleting a project in the Architect/Requirements client. Note that after a project is deleted, it is immediately removed from the list of projects.

By default, **Project.Cleanup.Now** is false. If **Project.Cleanup.Now** is true, after a project is deleted, the database space is cleaned up immediately by spawning an external process on the Application Server. If **Project.Cleanup.Now** is false, the project cleanup does not happen immediately. A subsequent administrative **projectCleanup** action is required to destroy the project completely, or upon the next complete run of **maintainDB**.

ExternalImportExport

When **True**, data imports and exports in the Architect/Requirements client run in separate Java processes outside the Web application server's JVM. Each import and export request from the client generates a process with its own JVM.

Imports and exports of large files can consume significant amounts of memory in the Web server. Running separate processes can help to avoid performance problems and failures due to out-of-memory errors.

External processes consume the Web server machine's memory. Set the **ExternalImportExport** parameter to **True** only if the server machine has sufficient memory for importing and exporting large files.

Not all imports and exports use a separate process when the value is **True**. Because the overhead can be significant, the server may determine that an import or export is too small to benefit from running in a separate process. For example, when importing a file smaller than two megabytes, an external process is not used.

Each external process automatically runs the **tcradmin** script to perform the import or export. The **ImportExportScript** configuration parameter controls the location of the **tcradmin** script file.

When **false**, the default value, imports and exports run within the Web server.

ImportExportDir

Complete path to the directory on the server where Architect/Requirements temporarily stores import and export files. The directory must exist and be writable by the Web server process.

If this parameter is not set, document export and import, XML export and import, and some features of the live Office interface do not work.

Some temporary files remain in the directory. The application server deletes temporary files created by Architect/Requirements that are more than two hours old. Architect/Requirements temporary files have the **TcR-** prefix.

ImportExportScript

Complete path to the **tcradmin** script file. The script file is created automatically by the Architect/Requirements installer. The **ImportExportScript** parameter is initialized to the file's location.

If the **tcradmin** script file is moved or you want to use a customized version, you must update the **ImportExportScript** parameter to point to the new location.

By editing the **tcradmin** script, you can configure Java command line arguments such as *maximum heap size* (*-Xmx*).

Custom.Folder.Path

Complete path to the directory on the server where user wants to place files related to Architect/Requirements configuration, such as locale-specific date styles. The directory must exist and be writable by the Web server process. The default value of this is blank.

A permanent location can be used for **Custom.Folder.Path** because its value is stored in the database and does not change during an upgrade.

MailServerIP

This is the IP address of the Mail server that is used to send E-mail. If this IP address is not set, send E-mail functionality does not work. Additionally, the change approval feature does not work

MessageQueue.Master

Determines if this application server Instance is the master for processing objects in the message queue. The message queue must be set as **Master** on one server to integrate with Engineering

Process Management. Setting this to **Master** also enables background processing, which results in improved performance when adding or removing properties for a type definition.

For more information, see Modifying the Properties of a Type Definition in the *Systems Architect/Requirements Management Project Administrator's Manual*.

Property updates may take a few seconds to many minutes, depending on the number of objects of that type, and how often the background queue is set to run. Also, the objects are collected for update using a query, and not all objects in a folder are processed at the same time. So, you may open a folder and see some objects that have the property changes applied and some objects where the changes are not applied yet.

Only one server (instance) should be designated as **Master**. The Master server runs a thread to process all tasks submitted to the message queue.

The possible values are **True**, **False**, or **port** number.

- **True**: Indicates that the machine is a single application server instance and is the master server for managing the message queue.
- **port**: Indicates that the machine is also the master server and has multiple instances running. The port designates the master instance.
- **False**: Indicates that message queue thread not run on this machine.

MessageQueue.Start

Determines if an application server instance participates in message queuing or not.

If background task processing is enabled (see **MessageQueue.Master**) all application server instances (including **MessageQueue.Master**) must have **MessageQueue.Start** set to **True**.

Setting **MessageQueue.Start** to **True** allows the application server instance to submit tasks to the message queue.

MessageQueue.Thread.SleepTime

The Message Queue server checks the queue at interval specified in this parameter.

PortRangeStart

Start of the port range the client searches when trying to find an available port on which to start its socket service.

PortRangeEnd

End of the port range the client searches when trying to find an available port on which to start its socket service.

In most installations the **PortRangeEnd** can be ~5 greater than the **PortRangeStart** value. But, when Citrix or Terminal Server are in use, the range should be 2x the expected maximum number of concurrent users running Architect/Requirements on that shared Microsoft Windows environment.

EnableAditionalSecurity

If **True**, Architect/Requirements validates all JSP parameters against cross site scripting attack.

Monitor.PollTime

Amount of time (in seconds) that the client waits before it polls the server for a monitored task. 0 indicates no polling.

RegeditEnabled

Administrator should set this parameter to **True** if users have access to read/write registry entries using **regedit.exe**. Architect/Requirements then gets and sets registry entries using **regedit.exe**.

The administrator must set this parameter to **false** if users do not have access to read/write registry entries using **regedit.exe**. Architect/Requirements does not use **regedit.exe** to get and set the registry. Instead it, it uses a third party utility, **ICE_JNIRegistry.dll**.

The default is **True**.

If you are using Internet Explorer as your browser, and if **RegeditEnabled** is set to **false**, and you want to perform multiple installations, you must start a new session of Internet Explorer for each installation. This is because while loading **ICE_JNIRegistry.dll**, a security exception is thrown by Internet Explorer when same browser window is used for second installation.

DB.greadSize

Integer threshold value for when to use a group read while reading a list of objects from the database. If a list is larger than the **greadSize** a group read is used. When reading lists of objects, such as the members of a folder, group reads can improve performance by reducing the number database operations (chattiness). However, group reads hurt performance when reading small lists. The threshold list size where group reads begin to improve performance depends on factors such as network latency between the database and application server. This value does not normally need to be adjusted.

WOLF.Remote.Trace.Deep

A boolean value that indicates if trace reports for integrations show a single level or all levels of linked objects. Setting **WOLF.Remote.Trace.Deep** to **True** will display all levels of trace links for a trace report request from a remote application.

Configuring Security Services

You can customize Architect/Requirements to allow Security Services by modifying the following parameters using the Configuration Web page. Security Services allows users of multiple Teamcenter applications to use a single login window to enable access to all Teamcenter modules.

Security Services includes a logging feature that may be useful for support and troubleshooting. For information about the feature, see Appendix C, Configuring Security Services Logging.

SSO.AppID

The ID that the Security Services server uses to identify Architect/Requirements.

SSO.Enabled

If this value is **True**, Architect/Requirements redirects login requests to the Security Services login URL. Users must have a Security Services user id and password.

SSO.LoginURL

The URL of the Security Services logon service.

SSO.ServiceURL

The URL of the Security Services identity service.

Configuring for Performance Improvement in WAN Environments

The Architect/Requirements system administrator can use certain configuration parameters to significantly improve performance for deployments in WAN environments. The http response from the Architect/Requirements server to the client is compressed using Java's API for zip.

Click the **Web Application Configuration** link on the Administrative Tools page to display the configuration parameters.

Filter.compression Threshold

Number of bytes in response above which the compression of response is enabled. The default value is **128**.

Filter.debug

Debug level for the compression filter. Zero is the minimum. Zero (**0**) is the minimum and is recommended for production.

Filter.doCompression

Indicates whether the http responses are compressed using GZIP compression. If all users are LAN-connected, there may not be any net gain to using compression, but it is a big benefit for WAN users. If compression is provided by application servers, Siemens PLM Software recommends that you set the **Filter.doCompression** parameter to **false** to avoid redundant layers of compression. The default value is **True**.

Configuring Password and Login Settings

You can customize user password settings to determine password length, password character requirements, and password expiration. The default value for each password parameter is 0, meaning the parameter is not enforced. If a user violates a password parameter, an error message appears. You can also set the number of failed logins before a user is suspended.

The password expiration and grace period rules do not apply to enterprise administrators, including the default **tcradm** account. If they did apply to these users, there would be a risk that all administrators could be denied access.

Password.Minimum Alpha Chars

The minimum number of required alpha characters in a password.

Password.Minimum Length

The minimum length for a password.

Password.Minimum Numeric Chars

The minimum number of required numeric characters in a password.

Password.Expiration Days

The number of days before a password expires. When this number is reached, users are given a warning message, instructing them to change the password.

Password.Grace Days

When a password is set or reset by the system administrator, the user has this number of days to create a new password. If the user fails to create a new password within the time allowed, he or she loses access to the system.

Password.Grace Expired Logins

Once a password expires, the user has this number of *logins* before losing access to the system. Each time a user logs in after password expiration, a warning message appears telling the user that he or she is about to lose access to the system.

Password.Account Lockout Threshold

The number of login attempts that are allowed before a user is suspended. Zero (0) indicates that the rule is not configured. Once the system administrator resets the password, the user must change it with the grace period defined by **Password.Grace Days**.

Setting Other Configuration Options

You can customize other Architect/Requirements options by modifying parameters using the Configuration Web page (**http://.../tcr/ugs/tc/req/configtcr.jsp**).

JRE.Version

This option allows the Administrator to supply a list of client side Java Run Time Environments (JRE). The list contains all "supported" JREs for the current release of Architect/Requirements.

The option list is configurable since new JRE versions released after Architect/Requirements/Requirements Management may be found to be compatible, and can be added to this list to enable their use. But, do not add JRE versions to this list unless instructed to do so by UGS Customer Support. Adding JREs that are not known to be compatible can lead to unpredictable client errors.

When you log in, Architect/Requirements looks at this list (contained in the database) and makes sure you have one of the JRE's that are on the list installed on your computer. If not, you are asked if you would like to install a compatible JRE.

The **JRE.Version** configuration option's default value includes only the 1.6 version. However, sites upgrading from previous Architect/Requirements releases should edit their **JRE.Version** option to have reference to the correct JRE version. Select the **Reset to Default** check box and click **Update**. Add the required JRE version to the list.

For information about version of Java Runtime Environment (JRE) and Java Plug-in supported, see the Siemens PLM Software Certification Database:

http://support.ugs.com/online_library/certification/

Package.Location

The path on the client machine to the optional **.jar** files containing methods that can be run via **createAction RunJava** to be included in the client's **classpath**. This parameter value must include the complete path.

If there is more than one jar file, the paths must be delimited by semicolons. For example:

```
C:\apps\sample1.jar;C:\apps\sample2.jar
```

The path name can accept environment variables in the **%ENVVAR%** notation.

For more information about **createAction RunJava**, see the *Systems Architect/Requirements Management API Reference* manual.

Configuring Log Server Parameters

You can customize the Architect/Requirements log server parameters by modifying the following parameters:

Log.Server.Level

Logging level for Architect/Requirements application server log file. This determines the types of messages written to the log file. Following types of messages can be written to the log file:

- `INFO`: It is the default mode. Setting **Log.Server.Level** to `INFO` helps record informational and warning messages.
- `WARN`: Setting **Log.Server.Level** to `WARN` helps record warning messages only.
- `DEBUG`: Setting **Log.Server.Level** to `DEBUG` helps record all messages including the debug messages.

  The log file size increases rapidly when `DEBUG` mode is set, hence, the `DEBUG` option must be used for short durations to diagnose specific problems.

Log.Server.Location

The directory where Architect/Requirements application server log file is saved.

You must restart the application server after updating **Log.Server.Location**.

Log.Server.MaxBackupIndex

The maximum number of backups of the Architect/Requirements application server log files that are maintained. When a new log file is created, a copy of the old log file is made. This setting determines the number of old log files to be preserved. After the maximum is reached, the oldest log file is deleted when a new log file is created. Log files are created when the application server is restarted or when the current log file reaches the maximum size as specified in **Log.Server.MaxFileSize**. Backup log files have a numerical extension added to give them a unique name.

You must restart the application server after updating **Log.Server.MaxBackupIndex**.

Log.Server.MaxFileSize

Maximum size (in MB) of the Architect/Requirements application server log file. When a log file reaches the maximum size, a backup of the log file is made and a new file is created.

You must restart the application server after updating **Log.Server.MaxFileSize**.

Log.Server.MemoryMonitorInterval

Time interval, in seconds, that the application server memory usage is logged to Architect/Requirements log file. The information is useful for investigating problems with excessive memory use.

Log.Server.Tcl.Level

Logging level for Tcl script logging in the Architect/Requirements application server log file. The setting behaves the same as Log.Server.Level except that it applies when Tcl scripts are running. This can help with debugging Tcl problems by recording only Tcl related debug messages rather than all debug messages.

Naming Convention

A **TcrServerLog_**<*port*>**.html** file is created in the location specified in the **Log.Server.Location** parameter. Replace `<port>` with the port number on which the application server instance is running. The naming convention is to support multiple instances/profiles of application server running on the same machine and saving the logs in the same location as specified in the **Log.Server.Location** parameter. The **TcrServerLog_**<*port*>**.html** file has the following columns:

- **Time**: Time when the event being logged occurred. It is the time when the log statement in the code is executed.
- **Thread**: Name of the thread that executed the log statement in the code.
- **UserName**: Name of the user logged on when the log was created. The log event created for the logged on user.
- **SessionName**: Name of the Versant session used when the log event occurred.
- **className**: Name of the class from where the event being logged occurred. This information is helpful for the debugging process.
- **Level**: Level at which the event was logged.
- **Message**: Message that was logged.

Naming Process

When the log file reaches the size specified in the **Log.Server.MaxFileSize** parameter then it is renamed to **TcrServerLog_**<*port*>**.html.1** and a new **TcrServerLog_**<*port*>**.html** is created. All subsequent messages are logged to the new file. The rollover behavior happens as follows:

1. **TcrServerLog_**<*port*>**.html.**<*N*> gets deleted or rolled off the list.

   Here **N = Log.Server.MaxBackupIndex**.
2. **TcrServerLog_**<*port*>**.html.**<*N* **–1**> gets renamed to **TcrServerLog_**<*port*>**.html.**<*N*>.
3. **TcrServerLog_**<*port*>**.html.1** gets renamed to **TcrServerLog_**<*port*>**.html.2**.
4. **TcrServerLog_**<*port*>**.html** gets renamed to **TcrServerLog_**<*port*>**.html.1**.
5. A new **TcrServerLog_**<*port*>**.html** is created.

Managing Message Queues

Architect/Requirements can perform some operations in the background. These operations include emptying the recycle bin, adding or removing properties from a type definition, and synchronizing information for Teamcenter Engineering and Teamcenter Enterprise integrations. When an operation is done in the background, a task object is created in the database and is added to the message queue. Tasks in the queue are processed in the order in which they are received. When a task operation completes successfully, the task is removed from the queue. If an error occurs, the task can be re-queued and run again later.

Viewing items in the message queue

You can see the pending tasks in the message queue from the **Message Store Administration** page. To view the **Message Store Administration** page, click **Administrative Tools →Configure Message Queueing** on the main Architect/Requirements page. A list of tasks waiting to be run is displayed.

Stopping the message queue

You can stop the message queue temporarily from the **Configure Message Queueing** page by clicking the **STOP** button. You can restart the queue processing by clicking the **START** button.

Removing tasks

An error condition can prevent a task from completing successfully. These tasks can remain in the queue. You can remove such tasks from the message queue.

You can delete tasks only when the queue is stopped.

To delete tasks from message queue:

1. Stop the queue.
2. Select the check box for the tasks that you want to delete and click **Delete Selection**.
3. Restart the queue.

Using Log Files for Troubleshooting

You can use Systems Architect/Requirements Management log files for debugging purposes.

To view the location of the Client log file:

1. Select **Tools→System Information**.
2. Click **Client Log**.

   The client log tab displays the path for the log file. You can navigate to the log path from Windows Explorer. The following log files are present in the log path location:

   - **tcr_log.txt**

     This is the only file with up to date client log information that you must use for debugging.
   - **tcr_log_old.txt**

     This is a backup of the log file. You can use this file for debugging a problem that happened earlier than the first item shown in **tcr_log.txt**.
   - **temp_tcr_log.txt**

     This is a snapshot of the log file created when you click the **Open In Notepad** button. This file is not kept up to date and you must not use it for collecting client log information.

Chapter 3: Tuning the Database

This chapter discusses how the Systems Architect/Requirements Management system administrator tunes the Versant database.

Tuning the Systems Architect/Requirements Management database involves the following:

- Setting Versant database parameters
- Running the Versant reorgdb Utility

Setting Versant Database Parameters

The Versant database can be tuned by changing the database parameters. All databases on a machine are created under the database root directory, which is created when Versant is installed.

Each Versant database is a collection of files located in a directory with the same name as the name of the database. For example, if your Systems Architect/Requirements Management database, **TCR_db**, is on a machine whose database root directory is **/files/versant/db**, that root directory contains a directory named **TCR_db**.

To find the exact location of the database root directory, enter the following command in a command prompt:

```
oscp -d
```

The following sections describe the files contained in each database directory.

System File

The database system volume, **system**, stores class descriptions and object instances for each database. There is one system volume for each database. However, additional volumes can be added through the **addvol** utility.

This file is never edited or manipulated directly. It is documented here only as general information.

Log Files

The physical log volume, **physical.log**, and the logical log volume, **logical.log**, record transaction activities and provide information for rollback and recovery.

These files are never edited or manipulated directly. They are documented here only as general information.

profile.be File

This is the Versant server process profile file. When a database starts, the database server process reads the **profile.be** file to determine the location of the database volumes and to set the database operating environment. If this file does not exist, you cannot start the database.

Changes to **profile.be** do not take effect until the database is restarted. For more information, see Stopping the Database in chapter 4, *Maintaining the System*.

These are some of the **profile.be** parameters that you can modify:

transaction

Specifies the number of concurrent transactions for this database. The default value is **100**. For example:

```
transaction 100
```

You must increase the **transaction** parameter value if:

- Several hundred users use the Systems Architect/Requirements Management database concurrently.
- You see the following error message:

  ```
  DB_TOO_MANY_USERS: (Too many users have logged into the database.)
  ```

For more information, contact Systems Architect/Requirements Management Customer Support.

lock_wait_timeout

Specifies the number of seconds the server waits for an object lock to be released before giving up. The default value is **5**. For example:

```
lock_wait_timeout 5
```

max_page_buffs

Specifies the maximum number of 16 KB buffers for caching disk pages from data volumes. This parameter strongly influences database performance. The default value is **2048**. For example:

```
max_page_buffs 2048
```

The default value should work well for most applications. For a large database, however, you may be able to improve performance by increasing the value. To conserve memory for small databases, you may want to decrease the value.

Running the Versant reorgdb Utility

The purpose of the **reorgdb** utility is to pack data so that space is used more efficiently. It reorganizes all data volumes for the database dbname by rearranging objects to remove gaps in physical space. The sequence of steps involved in reorganizing a database is:

1. Set the database to single-connection mode. For example:

   ```
   dbinfo -1 TCR_db
   ```

   Available options are:

   - `Dbinfo -m TCR_db`: This sets the database in multi user mode.
   - `Dbinfo -p TCR_db`: This prints a message that the database is in multi-user mode. It is just for visual confirmation that the database is in multi-user mode.

2. Stop the Versant database using stopdb. For example:

   ```
   stopdb TCR_db
   ```

3. Run the **reorgdb** *[options] dbname* utility. For example:

   ```
   reorgdb TCR_db
   ```

   The available option is noprint, which suppress display messages while running.

4. Place the database back into Multi-User mode:

   ```
   dbinfo -m TCR_db
   ```

5. Restart the Versant database using `startdb`.

   ```
   Startdb TCR_db
   ```

Chapter 4: Maintaining the System

This chapter describes the Systems Architect/Requirements Management system administrator's responsibilities in regular database and system maintenance, with instructions for specific tasks.

Cleaning up a Project

In previous releases, project deletion was performed in a single, long running operation, thus blocking the use of the Architect/Requirements client during the entire transaction. Now, the project deletion has been broken down into two phases.

In the first phase, a project is selected in the Architect/Requirements client and deleted. At this time, all the users are removed from the project, all the external references to the project are removed, and all the project's objects are removed from the Recycle Bins of all users. The cleanup operation can still take some time; however, after the cleanup, the project is completely inaccessible to the users. Control is now returned to the client. Users can continue working in Architect/Requirements.

All the objects in the project are now orphaned and take up space in the database. These objects do not interfere with any other Architect/Requirements transactions.

In the second phase, the project is cleaned up in the database.

For more information on cleaning up a project in the database, see Cleaning Up a Project in the Database.

Maintaining the Database

The Systems Architect/Requirements Management system administrator is responsible for regular database maintenance. This maintenance includes the following operations:

- Backing up the database
- Restoring the database
- Backing up and restoring the **osc-dbid** file
- Increasing the size of the database
- Running the **Database Maintenance** utility
- Stopping the database
- Starting the database
- Deleting the database

Backing Up the Database

You should have a backup strategy in place to protect the mission-critical information assets in your Systems Architect/Requirements Management database. Backups should already be a part of your organization’s standard disaster recovery plans. Systems Architect/Requirements Management must be added to these plans to ensure the capability to recover from a failure.

Generic operating system utilities for backing up file systems or disk images are not adequate for protecting the Systems Architect/Requirements Management database. When the database is running, some of its information is in memory. Therefore, that information is not captured by disk backup utilities, whether Systems Architect/Requirements Management clients are connected to the database or not.

The only way to ensure a completely recoverable backup is to use the Versant **vbackup** utility. **vbackup** can save a database reliably while users are connected to the database, and provides a binary backup. The binary file can be used to restore the database. **vbackup** is intended primarily for routine disaster recovery. It is included with the Systems Architect/Requirements Management installation package.

Siemens PLM Software strongly recommends that you use **vbackup**. General purpose file backup utilities do not capture all the information for databases that are running.

vbackup can perform incremental backups. A level 0 backup writes the entire database to the backup device. A level 1 backup writes all of the changes since the last level 0 backup. A level 2 backup writes all of the changes since the last level 1 backup.

To back up the database to a file, enter the following command:

```
vbackup -level level -device fileName -backup dbname
```

Enter variable values as follows:

Replace *level* with the type of backup you want to perform (0,1,2). If *level* is not specified, **vbackup** assumes a level 0 backup.

Replace *fileName* with the fully qualified path name of the backup file.

Replace *dbname* with the name of the database. For example, **TCR_db**.

To back up the database to a tape, enter the following command:

```
vbackup -level level -device tape-device -position position -backup dbname
```

Enter variable values as follows:

- Replace *level* with the type of backup you want to perform. If *level* is not specified, **vbackup** assumes a level 0 backup.
- Replace *tape-device* with the name of the tape device.
- Replace *position* with the tape position to which you want to back up the database. If *position* is not specified, **vbackup** assumes the current position.
- Replace *dbname* with the name of the database. For example, **TCR_db**.

Restoring the Database

If you used the Versant **vbackup** utility to back up the database, you can use **vbackup** to restore the database from a backup file or tape. You must first restore backup level 0. If there are more backup levels, restore level 1 and then level 2.

To determine the number of levels necessary to restore the database to its last backup state, enter the following command:

```
vbackup -info dbName
```

Replace *dbName* with the name of the database. For example, **TCR_db**.

To restore the database from a backup file, enter the following command:

```
vbackup -device fileName -restore dbName
```

Enter variable values as follows:

Replace *fileName* with the fully qualified path name of the file from which you want to restore the database.

Remember to restore the latest good level 0 backup and then, the latest good level 1 backup. Then, restore the latest good level 2 backup that goes with the level 1 backup.

Replace *dbName* with the name of the database. For example, **TCR_db**.

vbackup responds with a question about saving or applying log file contents. Although the default reply is **Yes**, Siemens PLM Software recommends that you reply **No**.

Backing Up and Restoring the osc-dbid File

A Versant file named **osc-dbid** tracks all databases on the network. Every time a database is created or deleted, the **osc-dbid** file is updated. Versant consults this file before it assigns a unique number to each database.

Versant uses this unique database number to assign a unique ID (called LOID) to each object on the network. Therefore, the **osc-dbid** file must be backed up regularly.

Ask your system administrator to include the **osc-dbid** file in whatever backups are performed on your system. The **osc-dbid** file is usually located on the Versant server.

To determine the file's complete path name, run the following command:

```
oscp -o
```

To determine the system where the **osc-dbid** file is located, run the following command:

```
oscp -n
```

The **osc-dbid** file can be restored only if no Versant database was created since the previous backup of the **osc-dbid** file. If a new database was created after the last backup of the **osc-dbid** file, do not restore the **osc-dbid** file. Contact Systems Architect/Requirements Management Customer Support for assistance.

Increasing the Size of the Database

The initial size of the Systems Architect/Requirements Management database is 2 GB by default. This initial allocation can be exceeded as a project progresses.

You must increase the size of the database if you see the following exception message:

```
SM_E_OUT_OF_VOL_SPACE: all volumes exhausted.
```

To increase the size of the database:

1. Log in using the account designated as the Systems Architect/Requirements Management administrator.
2.

   Add a volume by entering the following command:

   ```
   addvol [-i] -n volName -p path -s size dbName
   ```

   -i is optional. It causes the specified disk space (*size*) to be allocated and initialized immediately. That disk space is withheld from other applications, ensuring that the space is available when the Systems Architect/Requirements Management database grows enough to require it.

   Enter variable values as follows:

   - Replace *volName* with the name of the new volume, for example, **volume2**.
   - Replace *path* with the same value as *volName* if you are adding the volume on the same disk as the original database. If you are adding the volume on another disk, replace *path* with the full path name of the new volume.

     Using an absolute path name presents difficulties if the database is moved later. Siemens PLM Software recommends that you keep all volumes on the same disk. In either case, verify that sufficient space exists on the disk.

- Replace *size* with the quantity of disk space (in megabytes using **M** after the number, or in gigabytes using **G** after the number) that you want to add to the database.
- Replace *dbName* with the name of the database. For example, **TCR_db**.

For example, if you want to increase the size of the **TCR_db** database by 300 MB, and the name of the new volume is **vol2**, enter the following command on the machine where the database is located:

```
addvol -i -n vol2 -p vol2 -s 300M TCR_db
```

Running the Database Maintenance Utility

The database maintenance utility (**tcradmin**) performs diagnostics on the Systems Architect/Requirements Management database. This utility's **analyzeDB** option can check for inconsistencies that may arise, and can correct many of them when the **maintainDB** option is in effect. The utility makes two passes through the database. The physical pass looks for and reports low level database exceptions, if any. The logical pass detects and reports inconsistencies in Systems Architect/Requirements Management objects. The database maintenance utility (**tcradmin**) can examine the entire database, a specific project, all objects of a class, or a specific object.

Siemens PLM Software recommends that you schedule a regular job or task to run the database maintenance utility, perhaps weekly. If you have questions about scheduling, consult your system administrator.

Database maintenance can be run while the database is in use. However, it is preferable to run it when no users (or few users) are connected to the database, because inconsistencies could be mistakenly reported for objects involved in active transactions.

To use the database maintenance utilities (**maintainDB** and **analyzeDB**), you must be logged in to the system with the user ID that has a matching enterprise administrator user object in the Architect/Requirements database. You can use the `-user` command line option to identify the enterprise administrator account that you can use.

The **maintainDB** utility does the following:

- Creates a new **TcrAdminLog.html** file every time the **tcradmin** script is run.

  If you do not want a new file to be created, run **maintainDB** with the **-startNewLogFile false** option.

- When verbose mode is set to **false**, **maintainDB** prints out error versus all thread activities such as:

  - Begin first pass physical check
  - 10000 objects

  If you do not want the error versus all thread activities to be printed, run **maintainDB** with the **-verbose true** option.

The database maintenance utility (**tcradmin**) is enhanced with each release to detect and repair additional inconsistencies. So, a number of new messages may appear the first time you run it after upgrading. They are likely inconstancies that have been present for a long time, but were never reported before. If they were not evident in using Architect/Requirements before installing this release, there is no need to be concerned about them now that they are detected and corrected.

The **tcradmin** command accepts a maximum of 18 command line arguments. Also, the operating system limits the amount of data that can be entered on the command line. If the command line exceeds these limits, the **tcradmin** utility does not run.

To avoid these limitations, **tcradmin** also accepts arguments in a file.

- The `-argFile` argument specifies the file name.
- The file must have only one argument per line.

To run **maintainDB** using an argument file:

```
Tcradmin -argFile arguments.txt
```

Replace *arguments.txt* with the name of the file that contains the arguments.

For example, the argument file can contain:

```
-action
maintaindb

-mode
database

-logToStdOut
```

Maintenance utility logging and summary

Information about the progress of the utility and problems detected in the database is captured in a log. Output from **analyzedb** and **maintainDB** is written to the Architect/Requirements administrative log file **TcrAdminLog.html**, by default. The **TcrAdminLog.html** file is created in the current directory. You can change the location of the **TcrAdminLog.html** file using the `-logdir <directoryname>` option.

Use the `–logToStdOut` option to redirect the output to **stdout**. To capture the information as plain text in a file, use the operating system mechanism for redirecting **stdout** to a file.

You can mail the summary file to the Architect/Requirements administrator as part of a maintenance script.

Running analyzeDB from a Command Prompt

analyzeDB examines database content for consistency, but does not modify anything. It cannot be run from a client. It is run from a command prompt on the Systems Architect/Requirements Management server where the database is located.

analyzeDB examines the content of each object and all relationships between objects. It can take a long time to check a large database, perhaps several hours to check millions of objects.

Running maintainDB From a Command Prompt

maintainDB does the same checks as analyzeDB, but then takes appropriate corrective action on most anomalies that it finds. It cannot be run from a client. It is run from a command prompt on the Systems Architect/Requirements Management server where the database is located.

Database Maintenance Options

The following table show options for the database maintenance utility (**tcradmin**).

Table 4-1. Database Maintenance Options

Command Line Option	**Possible Value**	**Comments**
`-action` (Required)	`analyzeDB` `maintainDB`	**analyzeDB** reports only errors, does not repair. **maintainDB** reports errors, repairs errors, and reports repair failure.
`-mode` (Optional) If no **-mode** is passed, the default is database. If **-mode** option is provided, you need to provide correct **-name** option.	*project name*	Checks only a specified project. If the project is not found, an error is reported. Enter the project name next to the *-name* option. `-mode project -name myProject`
	file name	Checks only those LOIDs provided in the LOIDs file. If a problem occurs, an error is reported. The LOIDs file should be a **.txt** file with one loid per line. The pathname needs to be provided with path name in quotes. `-mode  file -name “C:\Temp\ TcR_LOIDs.txt”` Note that the LOID file can be created by running maintainDB with the –**generateLoidsOnly** option**.**
	loid	Check a specified LOID. You need to provide that LOID in quotes next to

Table 4-1. Database Maintenance Options

Command Line Option	Possible Value	Comments
		-*name* option. If the LOID is invalid, an error is reported. `-mode loid -name "403.0.12530"`
`-name` (Depends on mode option) If **-mode** option is database and **-name** is not passed, default database is used. If default database not found, error message is reported. For all other **-mode** options, you need to provide correct **-name** option of either **Project/file/LOID**. Otherwise, error message is reported.	*project name* *LOID file name*	If **-mode** is selected as project, Project name. If **-mode** is selected as loid, LOID. If **-mode** is selected as file, fully qualified path name of the file.
-generateLoidsOnly (Optional)		The **generateLoidsOnly** option gathers the objects to check, as specified by other options, and writes the LOIDs to a temporary file. The objects are not checked, just saved for later use. To check these objects run maintaindb again with the –**file** option.
`-logToStdOut` (Optional)		Redirects the output of the **tcradmin** command to **stdout**. By default, most of the output of **tcradmin** is written to the Architect/Requirements administrative log file, **TcrAdminLog.html**. The Architect/Requirements server log is written in a tabular HTML format that is convenient to view. When redirected to **stdout**, the Architect/Requirements log information is written as plain text. To capture the information as plain text in a file, use the operating system mechanism for redirecting **stdout** to a file.

Table 4-1. Database Maintenance Options

Command Line Option	Possible Value	Comments
`-class` (Optional) If any wrong class name is passed, check stops and reports an error message.		Checks objects with the specified class type. All other class are ignored. `-class RequirementDB` Checks **RequirementDB** objects only. If no **-class** detail provided, the default class used is **AbstractKernelDB**. For a list of all class names that can be used with the **-class** option, see appendix *A*, List of Class Names for Checking Database Objects.
`-verbose` (Optional)	`true` `false`	**true:** detailed report of the result. **false:** short summary of the result.
`-blocksize` (Optional) If no blocksize is passed, or if nothing is passed next to blocksize, or given as empty string, the default value is 10,000.		When there is multiple thread checking the database, the **-blocksize** option tells how many LOIDs per thread. `-blocksize 1000`
`-threads` (Optional) If no threads is passed, or if nothing is passed next to threads or given as empty string, the default value is 1		Indicates the number of threads to use for checking the database. `-threads 5`
`-user` (Optional)	*enterprise administrator name* For example, **tcradm**	Required only if the operating system account name running **tcradmin** does not a match an Architect/Requirements enterprise administrator name.

The **-blocksize** and **-threads** settings help manage how the analyzeDB and maintainDB utilities take advantage of the memory and processors in the server.

Objects are processed in batches. The number of objects in each batch is determined by the **-blocksize** value. Each batch is processed by a thread. Each thread starts up, connects to the database, processes its batch of objects, disconnects from the database, and shuts down. A new thread is started and repeats the process for the next batch. This clean start for each thread insures that memory does not creep up as processing continues. Smaller batch sizes use less memory, but the increased thread startup and shutdown can make the total process take longer. The default value is a reasonable starting place, but won't be optimum for all servers. Larger values can improve total processing time on servers with adequate RAM, and especially UNIX servers with a full 4 GB of address space in 32-bit processing mode. Smaller values may be necessary in special situations, but increase processing time.

The memory address space available for object checking is limited by Java runtime parameters within the tcradmin script. To allow the checking to use more (or less) memory, edit the script to change the **-Xmx** value, or Java maximum heap size. For example, set `-Xmx 1024M` for 1 GB of address space. The tcradmin script is in the location specified by the **ImportExportScript** configuration parameter.

Multiple batches can be processed concurrently by setting **-threads** to a value greater than 1. Multiple threads running in parallel can shorten the total processing time, up to the point where some system resource (CPU, RAM, database I/O) becomes saturated. Running multiple threads uses more CPU cycles and require more memory. The optimum number of threads will depend on the number of processors on the host, the total memory available, and consideration for any concurrent workload from TcSE users or other processing. Setting **-threads** to the number of physical processors in the server is a reasonable starting place. But, if other loading allows it, try 1.5 to 2 times the number of processors. Even servers with only a single processor show some benefit of using `-threads 2`.

Working from the defaults and these guidelines, the best values to use depends on local conditions; number of processors, physical RAM, database size, set of objects to be processed at different times, and the target run time window.

analyzeDB Examples

The following is a list of examples using the analyzeDB utility:

- The following example checks the complete database with only **RequirementDB** objects, and it performs only a logical check. With **-verbose** set to true, a detail report is generated. The report is created as **TcrAdminLog.html** file in the current directory. You can change the location of the**TcrAdminLog.html** file using the `-logdir <directoryname>` option.

  There are maximum five live threads to examine and each thread takes 100 LOIDS to examine.

  ```
  TcrServerDir\schema\tcradmin -action analyzeDB
  -mode database -class RequirementDB -type logical -verbose true
  -blocksize 100 -threads 5
  ```

- In this example, you provide the LOIDS in a text file and only those LOIDs are examined. In this example, the **-logToStdOut** option is used so the output is seen as plain text in the console. The check is performed on those LOIDs you provide in the **-loid_file** option. Both a physical and logical check are performed because the **-type** option is not provided.

  ```
  TcrServerDir\schema\tcradmin -action analyzeDB
  -mode file -name "C:\Documents and Settings\user\
  Local Settings\Temp\TcR_LOIDs.txt" -verbose true
  -blocksize 100 -threads 5
  ```

- To examine the entire database, enter the following command:

  ```
  TcrServerDir\schema\tcradmin -action analyzeDB
  ```

  Where *TcrServerDir* is the directory where the Systems Architect/Requirements Management server is installed. In Windows, it is the value of **USER_INSTALL_DIR** of the **HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\**

Teamcenter\SystemsEngineering\Server registry key. In UNIX database, while logged in as the Systems Architect/Requirements Management owner, run the command:

```
cat $HOME/TcSE_Server.properties
```

- To examine a specific project, enter the following command:

  ```
  TcrServerDir\schema\tcradmin -action analyzeDB -mode project
              -name projectName
  ```

  Replace *projectName* with the name of the project.

  On Solaris platforms, an error results if the *projectName* variable contains spaces, even if you place quotation marks around the project name. Instead of entering the project name directly, do the following:

  . Set the project name as an environment variable. For example:

  ```
  setenv PROJECT_NAME "Project Name"
  ```

  . Enter the following command:

  ```
  tcradmin -action analyzeDB -mode project -name $PROJECT_NAME
  ```

- To check a specified database class:

  ```
  TcrServerDir\schema\tcradmin -action analyzeDB -class classname
  ```

  The default is **all**, analyze all database classes

- To specify the type of test to perform:

  ```
  TcrServerDir\schema\tcradmin -action analyzeDB -type
  [physical | logical | both]
  ```

  Select an option as follows

  . **physical:** performs physical analysis only

  . **logical:** performs logical analysis only

  . **both:** performs both physical and logical checks.

maintainDB Examples

The following is a list of examples using the **maintainDB** utility:

- To examine and correct the entire database, enter the following command:

  ```
  TcrServerDir\schema\tcradmin -action
  maintainDB –logToStdOut > "logFile"
  ```

  Replace *logFile* with the name of the file in which you want to record the **maintainDB** results. The result is saved as a plain text file.

- To examine and correct a specific project, enter the following command:

  ```
  TcrServerDir\schema\tcradmin -action maintainDB -mode project
              -name projectName
  ```

  Replace *projectName* with the name of the project.

  On Solaris platforms, an error results if the *projectName* variable contains spaces, even if you place quotation marks around the project name. Instead of entering the project name directly, do the following:

  . Set the project name as an environment variable. For example:

  ```
  setenv PROJECT_NAME "Project Name"
  ```

  . Enter the following command:

  ```
  tcradmin -action maintainDB -mode project -name $PROJECT_NAME
  ```

Null owner errors may be reported when running **maintainDB**. To correct the errors, **maintainDB** moves the objects with no owners to the user's Recycle Bin. The Recycle Bin used will be for the Architect/Requirements user whose name matches the account name of the user running **maintainDB**.

For example, if you are running **maintainDB** logged into the system with the **tcradm** user ID, the objects that report null owners are moved to the Recycle Bin of the corresponding **tcradm** user in Architect/Requirements.

Objects moved to the Recycle Bin can then be manually examined and either destroyed, by emptying the Recycle Bin, or restored, by moving them to a valid owner.

Running the database maintenance utility in delta mode

With large databases, running a complete **maintainDB** as part of regular database maintenance takes a lot of time. Delta **maintainDB** solves this problem by checking only the portion of the database that is likely to have issues. Inconsistencies are normally introduced when objects are modified. In delta mode, **maintainDB** checks only the recently modified database objects. Running the maintenance in delta mode detects almost all issues and takes much less time than a full **maintainDB**.

Recently modified LOID capture

For delta **maintainDB** to function, the **LOIDs** of modified database objects must be captured. To capture the **LOIDs** of the modified database objects, the delta capture mode must be enabled by setting the web application parameter **DB.DeltaCapture** to **true**. In the delta capture mode, the **LOIDs** of database objects modified in each transaction are written to the database.

Enabling the delta capture mode can cause a slight degradation in performance and a small increase in database size. You must enable the delta mode only if you intend to use delta **maintainDB**.

Running delta maintainDB from a command prompt

To run the delta **maintainDB** from the command prompt, execute the `tcradmin` command with the `-action maintainDB -mode delta` option. In the delta mode, **maintainDB** collects the **LOIDs** of recently modified database objects that are stored in the database. These objects are then checked as regular **maintainDB**. After delta **maintainDB** runs, the list of recently modified **LOIDs** is automatically removed from the database.

Delta maintainDB usages

You can use delta **maintainDB** in two ways. You can run delta **maintainDB** on a daily basis to check all database objects modified on that day. Delta **maintainDB** can be used as a replacement for a full **maintainDB** to improve performance. No special options are required in this mode. You can also run delta **maintainDB** incrementally during the day. This mode requires setting up a cron job or other mechanism to execute delta **maintainDB** on a regular basis. Running delta **maintainDB** more frequently spreads the maintenance load throughout the day and detects issues sooner. In this mode, the checks are done while the database is in use. Delta **maintainDB** provides options to reduce conflicts with other users. To reduce the conflicts, you must run delta **maintainDB** with the options mentioned in the following section.

Delta maintainDB options

There are **tcradmin** command line options that work only with delta **maintainDB**. The following table shows the options for delta **maintainDB**.

Command Line Option	Possible Value	Comments
-action (Required)	**maintainDB**	
-mode (Required)	**delta**	To check the recently modified objects only.

Command Line Option	Possible Value	Comments
-delaytime (Optional)	**Time**	To skip checking an object if it has been modified within the specified time in minutes. This helps prevent checking an object that is actively being modified by another user.
-objectlimit (Optional)	**Count**	To limit the **maintainDB** run to check only a specified number of objects. This helps prevent **maintainDB** from over consuming resources such as CPU and memory. Objects not checked remain in the recently modified **LOID** list and are checked in a future delta **maintainDB** run.

All options of **maintainDB** work for delta **maintainDB** also . However, **-class** and options specific to a particular mode do not work for delta **maintainDB**.

Delta maintainDB examples

- To examine and correct all recently modified database objects, enter the following command:

  TcrServerDir`\schema\tcradmin –action maintainDB –mode delta –logToStdOut >` "*logFile*"

- To use delta **maintainDB** while lessening the impact on other database users, enter the following command:

  TcrServerDir`\schema\tcradmin –action maintainDB –mode delta –delaytime 30 –objectlimit 10000  –logToStdOut >` "*logFile*"

  A *30* minute delay avoids checking objects that are actively getting modified by other users. A *10000* object limit prevents **maintainDB** from overusing system resources.

Clearing the recently modified LOID list

When the modified object capture mode is on (the web application parameter **DB.DeltaCapture** is set to **true**), the **LOIDs** of the modified database objects are written to the database. This increases the database size if the list is not cleared regularly. The list is cleared automatically when delta **maintainDB** is run. The list can also be manually cleared with the following command:

TcrServerDir`\schema\tcradmin –action dropDelta`

After a full **maintainDB** is run, the modified object list is not required. The list can be cleared using the **-dropDelta** option. For example:

TcrServerDir`\schema\tcradmin -action maintainDB -mode database -dropDelta`

Temporarily disabling modified LOID capture

For operations such as a large project import, capturing modified **LOIDs** is not required. Capturing **LOID** is disabled by setting the **DB.DeltaCapture** parameter to **false**. **LOID** capture mode is temporarily enabled or disabled in a transaction with the **setEnvironment** command. For example:

```
setEnvironment deltaMode false
```

Stopping the Database

You must stop the Systems Architect/Requirements Management database before you reboot the machine on which the database resides. In addition, you must stop the database to perform some maintenance and tuning operations.

- There cannot be any client connections to the database when it is stopped.
- The Systems Architect/Requirements Management Web server application must be shut down.

To stop the database named *DBname*, run the following command:

```
stopdb DBname
```

For example, `stopdb TCR_db`.

Starting the Database

The Systems Architect/Requirements Management database need not be started explicitly. It is started automatically when a request for connection comes up. Sometimes, however, it may be necessary to start the database for diagnostic purposes.

To start the database named *DBname*, enter the following command:

```
startdb DBname
```

For example, `startdb TCR_db`.

Deleting the Database

To delete the Systems Architect/Requirements Management database named *DBname* completely from your system, enter the following command:

```
removedb -rmdir DBname
```

For example, `removedb -rmdir TCR_db`.

This command does not ask for confirmation.

Cleaning Up a Project in the Database

To recover the database space upon deleting a project in the Architect/Requirements client, a database cleanup operation is required. The cleanup operation can be performed in either of the following methods:

- **Default (Project.Cleanup.Now=false):**

Recover the database space by the **tcradmin** script on the database server machine, using the **projectCleanup** option. Run the following command to use this option:

```
tcradmin –action projectCleanup
```

The **projectCleanup** option searches the database for projects that are deleted. Then, it runs **maintainDB** with specific options that destroy all the objects in the deleted projects, therefore, recovering the database space.

The **tcradmin -action maintainDB** command (with no other options, therefore, running complete maintenance on the entire database) also performs the **projectCleanup** operation as the first step of **maintainDB**.

- **Optional (Project.Cleanup.Now=true):**

  Set the Architect/Requirements Web application parameter **Project.Cleanup.Now** to **true**, which cleans up the database space immediately by spawning an external process on the application server. Then, run the **tcradmin –action projectCleanup** command.

  Project.Cleanup.Now uses the **schema** directory on the application server machine; therefore, the required setup must be in place. **Project.Cleanup.Now** runs on the application server, which adds significantly to the network load during the cleanup operation. Therefore, the default cleanup method is to use **maintainDB** on the database server.

For more information on configuring project cleanup, see Project.Cleanup.Now.

The **tcradmin -action maintainDB** command (with no other options, therefore, running complete maintenance on the entire database) also performs the above **projectCleanup** operation, as the first step of the **maintainDB**.

Running System Utilities

Several utilities display information that can be helpful in diagnosing network and web server installation problems. They are detailed in this section.

In Internet Explorer, open the Systems Architect/Requirements Management home page, and then click the **Administrative Tools** link. From the Administrative Tools page, click the **Diagnostic Tools** link.

From the Diagnostic Tools page, you can access the following options:

- **TcSE Home:** Return to the Systems Architect/Requirements Management home page.
- **Server Properties:** Monitor Java environment properties.
- **Server Information:** View Web server information including server name and IP address, protocol, server port, and application server.
- **Web Application Configuration:** Access the Web Application Configuration page where you can configure Systems Architect/Requirements Management.
- **Office Live Diagnosis:** Access the Office Live Diagnosis page. This page tells users if their local machine is properly configured to run Office Live, which lets you create, edit, view, and manipulate objects in Systems Architect/Requirements Management through the Microsoft Office suite of products.
- **Server Installation Diagnosis:** View information regarding the server installation in the event of an installation problem.
- **Server Integration:** View information regarding Application Registry installation in the event of an Application Registry installation problem.
- **List Current Users:** View a list of all users logged in to Systems Architect/Requirements Management. Shows each user's full name, email, the time he or she logged in, and the status of the session. The status is based on the duration that the session is left idle. If session is idle longer than the applications **session-timeout** value as set in **web.xml** file, it is flagged as **Dormant**.

Systems Architect/Requirements Management allows for enterprise deployments of more than one Application server. This utility works by detecting sessions at the Application server level. So, if you have more than one Application server deployed, you need to connect to each Application server individually to get the list of *all* users logged into Systems Architect/Requirements Management.

Chapter 5: Managing Licenses

This chapter describes Architect/Requirements license key files and contains instructions for working with license keys. A table of possible server errors is included for troubleshooting.

Architect/Requirements License Key Files

To use the Architect/Requirements server, a license key file (**tcr.lic**) must be obtained from Siemens PLM Software. You receive this file when you register your Architect/Requirements application. With the license key file, you also receive a customer number for your organization.

When you obtain the license key file and customer number, you use the Web Application Configuration page to configure license information. For more information, see Configuring License Information in chapter 2, *Configuring Architect/Requirements*.

The Architect/Requirements server must be installed before you can configure license information. For more information, see the *Systems Architect/Requirements Management Server Installation Manual*.

A license key file is an encrypted binary file and consists of the following:

- **Customer Number**

  Specifies the number assigned to your organization by Siemens PLM Software Customer Service. The number should be entered into the Configuration Web page (**http://.../tcr/ugs/tc/req/configtcr.jsp**).

- **Expiration Date**

  Specifies the date on which the license becomes invalid. After this date, the Architect/Requirements server will not start.

- **IP Address**

  Specifies the IP address at which the Architect/Requirements server is licensed to run.

- **Number of Seats**

A **Read/Write** license is consumed if a userâ€™s maximum privilege property is **Read-Write**, **Project Administrator**, **Enterprise Administrator**, or **Database Administrator**.

A **Read/Only** license is consumed if a userâ€™s maximum privilege property is **Read Only**. However, if a **Read/Only** license is not available, then a **Read/Write** license is consumed.

A **Scripting** license is consumed when a user is given **Script Authoring** privilege for at least one project.

Users need the **Script Authoring** privilege to create, edit, and run activators in each project. To give this privilege to a user for a particular project, change the user object in the projectâ€™s **Users** folder by setting the **Additional Privilege** property to include **Script Authoring**.

Because enterprise administrators have access to all projects in the system, their user objects appear only in the **Users** folder of the **TcSE Administration** project. However, a user does not need **Script Authoring** privilege if an activator is triggered as a result of another userâ€™s actions.

- **Version Number**

  Specifies the license version that corresponds to the version of the Architect/Requirements application.

On startup of the Architect/Requirements server, each data item in the license key is checked for validity.

Viewing License Information

For each type of Architect/Requirements license, you can view the following information:

- The total number of licenses for your customer number.
- The number of licenses that are in use.

To view license information:

1. On the Teamcenter systems engineering and requirements management home page, click the **Administrative Tools** link.
2. On the Administration Tools page, click the **TcSE Licensing** link.

   The License Information page displays the license count from all current license keys.

   Also on this page, you can click the **Manage Licenses** link to access the license management utility. For more information, see Accessing the License Management Utility, later in this chapter.

Accessing the License Management Utility

You access the license management utility through the License Management Page. On this page, you manage licenses by adding and removing license keys. For more information, see Adding a License Key and Removing License Keys, later in this chapter.

A license key is an encrypted text string that controls a certain number of licenses. Each license key contains information such as your customer number, an expiration date, and the number of seats for each license type.

- Your valid customer number must be entered before you start this procedure. For more information, see Configuring License Information in chapter 2, *Configuring Architect/Requirements*.
- **Enterprise Administrator** privilege is required for this procedure.

To access the license management utility:

1. On the Teamcenter systems engineering and requirements management home page, click the **Administrative Tools** link.
2. On the Administrative Tools page, click the **TcSE Licensing** link.
3. On the License Information page, click the **Manage Licenses** link.
4. On the Teamcenter systems engineering and requirements management login page, enter your enterprise administrator user name and password, select a language, and click **Log In**.

   The License Management Page is displayed.

   An alternate page is displayed if your database is new and if your customer number is not entered. This page contains a link to the Web Application Configuration page. Click this link, enter your customer number in the **LIC.CustomerNumber** parameter, and click the **Update** button. Then repeat this procedure.

Adding a License Key

You can add a new license key without replacing existing licenses. The licenses in the new key are automatically included in the total license count for the Architect/Requirements server.

You can view the server's total license count on the License Information page. For more information, see Viewing License Information, earlier in this chapter.

Enterprise Administrator privilege is required for this procedure.

To add a license key:

1. Open the license key file in a text editor (for example, Microsoft Notepad), and copy the encrypted string to the clipboard.
2. In the text field at the bottom of the License Management Page, delete the words “Enter Key Here,” and then paste the encrypted string into the field.

   For more information, see Accessing the License Management Utility, earlier in this chapter.

   You can reverse this action by clicking **Clear**.

3. Click **Add Key**.

   The license management utility checks the license key. When the license key is validated, a confirmation page displays the license key information in unencrypted format.

   If the license key is invalid, expired, or a duplicate, an error message is displayed. Click **Back** to return to the License Management Page, where you can enter the key again or enter another key.

4. Click **Add**.

   The License Management Page displays the new license key in the table of encrypted strings.

   If you have two or more Web application servers that point to the same database, and if you manage each server separately through the **Installation Key** parameter in the web.xml file, you must add this license key on each server.

Removing License Keys

When a license key expires, its licenses are automatically subtracted from the total license count for the Architect/Requirements server. However, expired license keys remain in the database until you remove them.

You can view the server's total license count on the License Information page. For more information, see Viewing License Information, earlier in this chapter.

Enterprise Administrator privilege is required for this procedure.

To remove license keys:

1. In the table of encrypted strings on the License Management Page, check the check box for each license key that you want to remove.

   For more information, see Accessing the License Management Utility, earlier in this chapter.

2. Click **Remove Key**.

   A confirmation page displays the key number and encrypted string for each selected license key.

   To return to the License Management Page without removing any licenses, click **Back**. You can click **Cancel** to abort this operation and exit the license management utility.

3. Click **Remove** to remove all licenses in each displayed license key.

   The License Management Page is displayed with the license keys removed from the table of encrypted strings.

   If you have two or more Web application servers that point to the same database, and if you manage the servers separately through the **Installation Key** parameter in the web.xml file, you must remove these same license keys from each server.

Troubleshooting Server Errors

Table 5-1 describes the types of server errors that can occur because of incorrect license configuration.

Table 5-1. Possible Server Errors in Systems Architect/Requirements Management License Configuration

Type of Error	Reason for Error
`InvalidLicenseException: Invalid License`	Either of the following: • The expiration date has passed. • The version number in the Configuration Web page does not match the Systems Architect/Requirements Management version number.
`InvalidLicenseException: Given final block not properly padded.`	Any of the following: • The Configuration Web page does not contain the customer number. • The customer number in the database is invalid. • The version number on the Configuration Web page does not match the Systems Architect/Requirements Management version number.
`License Management not configured.`	Security libraries are not located in the appropriate directories.

Appendix A: List of Class Names for Checking Database Objects

This appendix lists the class names that can be used with the **-class** option of the database maintenance utility (**tcradmin**). The **tcradmin** options are described in table Database Maintenance Options of chapter *Maintaining the System*.

The **-class** <*className*> option of the **tcradmin** command checks objects in the database with the specified class name. All other classes are ignored.

Table A-1. Class Names for -class Option

Class Name	Class Name	Class Name
AbstractApplicationDB	ConnectionDB	ParamDB
AbstractAttributeDB	CounterDB	PortDB
AbstractAttributeDefinitionDB	DateAttributeDB	ProjectDB
AbstractContainerDB	DateDefinitionDB	ProxyDB
AbstractDefinedPropDB	DiagramDB	RequeueProcessInfoDB
AbstractDesignDB	DiagramLinkDB	RequirementDB
AbstractGroupDB	DiagramOwnerLinkDB	SchemaUpdateTaskDB
AbstractHandleDB	DiagramStencilLinkDB	SearchDB
AbstractKernelDB	DocumentTemplateDB	SecurityProfileDB
AbstractLinkDB	EmptyTrashTaskDB	ServerRunningCheckRuleDB
AbstractNamedDefinitionDB	EventLinkDB	SessionDB
AbstractOutputTemplateDB	ExcelTemplateDB	ShortCutDB
AbstractOwnedObjectDB	ExportedProxyDB	ShortCutLinkDB

Table A-1. Class Names for -class Option

Class Name	Class Name	Class Name
AbstractProcessInfoDB	ExportedProxyLinkDB	SpreadsheetDB
AbstractQueryDB	FillinAttributeDB	StencilDB
AbstractQueuedTaskDB	FillinDefinitionDB	StringDB
AbstractRuleDB	FolderDB	StyleSheetDB
AbstractShortCutDB	GenericLinkDB	SymbolicLinkDB
AbstractUsesLinkDB	GroupDB	TcIntegrationSOAPTaskDB
AccessLinkDB	GroupLinkDB	TemplateDB
AccessProfileDB	HierShortCutDB	TestTaskDB
ActivatorDB	LinkDB	TimerRuleDB
AdminFolderDB	MapDB	TraceLinkDB
AttachmentDB	MarkerDB	TransactionDB
BuildingBlockDB	MatlabDB	TrashCanDB
ChangeApprovalDB	MessageQueueDB	UserDB
ChangeLogDB	MessageStoreDB	UserGroupDB
ChoiceAttributeDB	NoteDB	UserTypeDefinitionDB
ChoiceDefinitionDB	NumericAttributeDB	WolfHandleDB
ChoiceStringDB	NumericDefinitionDB	

Appendix B: Modifying the Client Memory Allocation

Architect/Requirements is configured for optimum memory allocation for machines with **4 GB** RAM. However, depending on your computers, you may need to increase or decrease the client memory allocation.

To modify the client memory alocation, you need to update the **login.jsp** file in the **tcr.war** file.

To modify the client memory allocation

1. Create a temporary folder and extract the **launch.jsp** file from the **tcr.war** file.

   Open the command prompt, change to the directory location of the **tcr.war** file, and run the following command:

   ```
   jar xvf tcr.war ugs/tc/req/launch.jsp
   ```

   You must have the JDK installed to run the **jar** command.

   You can download JDK from http://www.oracle.com/technetwork/java/javase/downloads/index.html.

2. Open the **ugs/tc/req/launch.jsp** file in a text editor like **Textpad** or **Notepad++**.
3. Search for the following string:

   ```
   <PARAM name="vm_args" value="-Xms64m -Xmx1024m"> <!-- vm args for launching
   client -->
   ```

   Xmx1024m in the string denotes the maximum heap space. In this case, the heap space is set to **1024 MB** or **1 GB**.

   Change it to the desired value.

   For information about the optimum heap space, see the Java documentation at docs.oracle.com.

4. To prevent adding the existing **tcr.war** file to the new WAR file, delete or move it (the existing **tcr.war** file) from the temporary folder.
5. Add the updated **launch.jsp** file to the **tcr.war** file. Run the following command at the command prompt:

   ```
   jar uvf tcr.war ugs/tc/req/launch.jsp
   ```

6. Deploy the war file on your Web server.

Appendix C: Configuring Security Services Logging

Security services has a logging system separate from the TcSE server log. Security services log files are located in the directory **./logs/TcSS**. The logs will contain any errors or warnings detected.

By default the log file is created relative to the current directory for the application server. This is commonly the application servers root directory. But you may need to determine what the current directory is to locate the logs.

If you need to change the log file location or get more detailed information for debugging there is an XML file used for configuration. The Configuration file is in the war file at this at this location, **WEB-INF\classes\log4j2.xml**.

To modify the configuration:

1. Create a temporary folder and extract the **log4j2.xml** file from the **tcr.war** file.

   Open the command prompt, change to the directory location of the **tcr.war** file, and run the following command:

   ```
   jar xvf tcr.war WEB-INF/classes/log4j2.xml
   ```

   You must have the JDK installed to run the **jar** command.

   You can download JDK from http://www.oracle.com/technetwork/java/javase/downloads/index.html.

2. Edit WEB-INF/classes/log4j2.xml and locate the security services logger entries:

   ```
   <Logger name="com.teamcenter.ss" level="WARN" additivity="true">
     <AppenderRef ref="javaClientFileWriter"/>
   </Logger>
   <Logger name="com.teamcenter._ss" level="WARN" additivity="true">
     <AppenderRef ref="javaClientFileWriter"/>
   </Logger>
   ```

3. Change the log level by replacing **WARN** with **INFO** or **DEBUG**.
4. Change the log file location by changing the AppenderRef entry. If you want the messages in the application servers console change **javaClientFileWriter** to **STDOUT**.

5. Add the updated **launch.jsp** file to the **tcr.war** file. Run the following command at the command prompt:

   `jar uvf tcr.war` WEB-INF/classes/log4j2.xml

6. Deploy the war file on your Web server.

System Administration Index