Oracle® Communications ASAP

System Administrator's Guide

Release 8.0
G48711-01
February 2026

ORACLE

Oracle Communications ASAP System Administrator's Guide, Release 8.0

G48711-01

Contents

2 Monitoring and Managing ASAP

3 Improving ASAP Performance

About This Content

This document describes how to start and stop Oracle Communications ASAP, configure security for ASAP, and manage and monitor ASAP. It also includes information about improving ASAP performance, managing the database and file system, backing up and restoring ASAP, and troubleshooting ASAP.

Audience

This document is intended for system administrators, system integrators, and other individuals who need to maintain and work with ASAP.

1

Setting Up and Managing ASAP Security

This chapter describes the security features for Oracle Communications ASAP.

Overview of Setting Up ASAP Security Features

ASAP security is designed to provide maximum confidentiality and data integrity, while ensuring on-demand access to services for authorized users. ASAP security is designed for three essential functions: managing ASAP users in Oracle WebLogic Server, securing data, and protecting diagnostics files.

ASAP provides these security functions in the following locations:

- WebLogic Server security

  WebLogic Server contains default users, groups, and roles that support the various WebLogic-based ASAP functions, like the Order Control Application (OCA) clients, the Service Activation Configuration Tool (SACT), the Service Activation Deployment Tool (SADT). Java Service Request Processor (JSRP) clients also need WebLogic Server credentials to connect to the JSRP interfaces.

- ASAP server and database credential security

  The ASAP environment contains a credential store factory (CSF) wallet that stores the ASAP schema user names and passwords, and the WebLogic Server user name and password. These credentials are called class A secure data. Each ASAP server can use this wallet to obtain these credentials. The CSF wallet provides both secure storage and encryption for these credentials.

- Network Element (NE) credential security for Network Element Processor (NEP) to NE communication

  The ASAP Control database stores credentials required for NEPs to access NEs. These credentials are called class B secure data. You can use a security tool to add, change, or delete credential information, and also to enable encryption.

- ASAP system configuration parameters

  Some ASAP system configuration parameters can have a significant impact on security. Parameters settings, such as diagnostic levels and server security attributes should be configured correctly to ensure data security.

Configuring WebLogic Server Security

ASAP uses the embedded Lightweight Directory Access Protocol (LDAP) server included with the WebLogic Server software to manage default ASAP users, groups, roles, and methods. For more information on this embedded LDAP server, see the WebLogic Server documentation.

About ASAP WebLogic Users, Groups, Roles, and Methods

User security is managed with the WebLogic Server. For more information, see Oracle WebLogic Server Documentation.

WebLogic Server provides the following features:

- ASAP OCA client users can change their passwords. For more information, refer to the *ASAP Order Control Application User's Guide*.
- ASAP Administrators can use the WebLogic Remote Console to add, delete, or edit users or user groups.
- ASAP Administrators can use the WebLogic Remote Console to assign or unassign permissions to users or user groups.

  The administrator uses security roles, method names, and principal names to configure permissions using the WebLogic Remote Console.

  For information on migrating security from a previous version of ASAP, refer to the *ASAP Installation Guide*.

You can create, delete, and modify ASAP users and groups in WebLogic Server. For more information, see Oracle WebLogic Server Documentation.

> **Note**
>
> ASAP requires that WebLogic user passwords have eight characters or more.

Table 1-1 lists and describes the default ASAP users.

Table 1-1 Default ASAP WebLogic Server Users

Default Users	Description
WebLogic_Admin	This is the WebLogic Server administrator account, where *WebLogic_admin* is the user name you selected when you created your WebLogic Server.
ASAP_admin	This is the default OCA client user. This user can create OCA orders, manage fallout orders, and view audit log reports in the OCA client. For more information about OCA permissions for this user, see "Understanding OCA Client Group Permissions." This user is a member of the **ASAP_VNOS** and **ASAP_Operators** groups.
ASAP_monitor	This is a default OCA client user that can be used to view OCA audit logs and reports. For more information about OCA permissions for this user, see "Understanding OCA Client Group Permissions." This user is a member of **ASAP_Guests** group.
ASAP_operator	This is a default OCA client user. This user can create OCA orders, manage fallout orders, and view audit log reports in the OCA client. For more information about OCA permissions for this user, see "Understanding OCA Client Group Permissions." This user is a member of the **ASAP_Operators** group.
asap_ws_user	User who has access to **ASAP_WS_USERS_GROUP**.
Cartridge_Management_WebService_MDB	User who has access to **Cartridge_Management_WebService** group.
cmws_studio	User who has access to **Cartridge_Management_WebService** group.
OracleSystemUser	Oracle application software system user.

Table 1-2 lists and describes the default ASAP groups.

Table 1-2 Default ASAP WebLogic Server Groups

Default Groups	Description
AdminChannelUsers	AdminChannelUsers can access the admin channel.
Administrators	Administrators can view and modify all resource attributes and start and stop servers.
AppTesters	AppTesters group.
ASAP_Guests	Allows an OCA user to login to OCA and access various methods.
ASAP_Monitors	Allows an OCA user to view OCA audit logs and reports in the OCA client.
ASAP_Operators	Allows an OCA user to create orders, manage fallout orders, and view audit logs and reports in the OCA client.
ASAP_VNOS	The virtual network operator (VNO) group ID must be included as a member of ASAP_VNOS group. The group will be the parent of all VNO groups. This group enables the OCA client to identify and recognize a group ID as VNO group ID and therefore authorize a user as VNO user.
ASAP_WS_USERS_GROUP	Group to use role ASAP_WS_USERS.
Cartridge_Management_WebService	Group to use role CARTRIDGE_MANAGEMENT_WEBSERVICE.
CrossDomainConnectors	CrossDomainConnectors can make inter-domain calls from foreign domains.
Deployers	Deployers can view all resource attributes and deploy applications. DefaultAuthenticator.
Monitors	Monitors can view and modify all resource attributes and perform operations not restricted by roles. DefaultAuthenticator.
Operators	Operators can view and modify all resource attributes and perform server lifecycle operations. DefaultAuthenticator.
OracleSystemGroup	Oracle application software system group. DefaultAuthenticator.

> **Note**
>
> You must add the newly created users as a member of one of the ASAP groups (**ASAP_Guests**, **ASAP_Monitors**, or **ASAP_Operators**).

Understanding OCA Client Group Permissions

Table 1-3 lists and describes the ASAP permissions for OCA client groups.

Table 1-3 WebLogic Server Permissions for the OCA Client Groups

Method	ASAP_Operator	ASAP_Monitor	ASAP_VNOS	Enables Users to…
SA_Order_Control_Application	yes	no	yes	Access the OCA client.
SA_View_Audit_Log	yes	yes	yes	Use the audit log query function.
SA_Order_Entry	yes	no	yes	Use the order entry function.
SA_Order_Management	yes	yes	yes	Use the order management features. This includes the Work Order Query window.
SA_Abort_CSDLs	yes	no	yes	Abort Common Service Description Layers (CSDLs) in the Work Order Details window.

Table 1-3 (Cont.) WebLogic Server Permissions for the OCA Client Groups

Method	ASAP_Operator	ASAP_Monitor	ASAP_VNOS	Enables Users to…
SA_Add_CSDL_Parameters	yes	no	yes	Add CSDL parameters in the Work Order Details window and access the CSDL Parameter dialog box.
SA_Delete_CSDL_Parameters	yes	no	yes	Delete an optional CSDL parameter from the Work Order Details window.
SA_Edit_Global_Macros	yes	no	yes	Override a global parameter to make it a local CSDL parameter and change the value in the Global Parameter dialog box.
SA_Edit_CSDL_Parameters	yes	no	yes	Modify CSDL parameter values in the Input dialog box.
SA_Insert_CSDLs	yes	no	yes	Add CSDLs to work orders (WOs) using the Select CSDLs dialog box.
SA_Release_Work_Order_Change_Due_Date	yes	no	yes	Release a WO to the Provisioning queue with a changed due date in the Release dialog box.
SA_Resequence_CSDLs	yes	no	yes	Correct the CSDL sequence using the Resequence – CSDL dialog box.
SA_Retry_CSDLs	yes	no	yes	Retry individual or failed CSDLs in the Work Order Details window.
SA_Abort_Work_Orders	yes	no	yes	Abort WOs in the Work Order Details window.
SA_Cancel_Work_Orders	yes	no	yes	Cancel WOs from the Work Order Query window.
SA_Delete_Work_Orders	yes	no	yes	Delete WOs from the Work Order Query window.
SA_View_Work_Order_Details	yes	yes	yes	Query WOs and view their details in the Work Order Details window.
SA_Release_Work_Orders	yes	no	yes	Release WOs for provisioning from the Work Order Details window.
SA_Stop_Work_Orders	yes	no	yes	Stop single or multiple WOs from the Work Order Query window.
SA_Change_Password	yes	yes	N/A	Change their user passwords.
SA_All	yes	no	yes	Access all OCA functionality (see note below) and to unlock WOs that have been locked by other users. **Note:** SA_All excludes SA_Change_Password. You must include the SA_Change_Password method in addition to SA_All to grant a user complete access to all OCA functionality. If you do not include SA_Change_Password, the Change Password option in the Security menu will be disabled.

Table 1-3 (Cont.) WebLogic Server Permissions for the OCA Client Groups

Method	ASAP_Operator	ASAP_Monitor	ASAP_VNOS	Enables Users to…
SA_Copy_And_Update_Work_Orders	no	no	no	Use the order entry function. This is similar to the **SA_Order_Entry** permission with limitations. With this permission, you can select the **Copy to Submit...** menu option, and update the existing CSDLs in the **New Work Order** window. However, you will not be able to add, delete CSDLs and access the **Work Order Details** tab. **Note**: • To use this method, update **ssam.jar** and add **SA_Copy_And_Update_Work_Orders** to at least one group. For information on how to update the method, see "Updating Methods Role Assigned to a Group or User in WebLogic Server". • **SA_Order_Entry** permission takes precedence over **SA_Copy_And_Update_Work_Orders** if a user has both permissions.

Configuring Virtual Network Operator Authorization for OCA Users

The ASAP administrator should create WebLogic Server user accounts for VNO fallout managers and include their user login names under one of VNO groups. The login name of any VNO user must be a member of one or more VNO groups and one ASAP group. Below is an example of "User_A" created as a VNO operator for the "Telco 1" group and "User_B" created as a VNO monitor for "Telco 1" and "Telco 2" groups.

Table 1-4 lists and describes possible VNO user and group settings.

Table 1-4 Sample VNO Authorization

Login Name	Is a Member of	Group
User_A	is a member of...	ASAP_Operators Telco 1
User_B	is a member of...	ASAP_Monitors Telco 1 Telco 2

Do the following:

- Create VNO groups with unique names and add VNO groups as members to ASAP_VNOS group.
- Add OCA user login names as members to one or more VNO groups. Ordinarily, an OCA user is added to one VNO group.

- Add each OCA user's login name as a member to one ASAP group (ASAP_Operators or ASAP_Monitors).

VNO functionality is available to divide OCA user groups into geographic regions using the VNO_ID_DEFAULT and VNO_ID_STRIP parameters of the **ASAP.cfg** file.

Refer to the discussion about the Service Request Processor (SRP) Emulator Server Configuration Parameters in the *ASAP Server Configuration Guide*.

Configuring Authentication Providers for ASAP

During the ASAP installation process, the ASAP installer creates default ASAP users, groups, roles, and methods in the embedded LDAP authentication provider included with WebLogic Server. You can use this authentication provider to configure the default ASAP users, groups, roles, and methods, or add, delete, or modify your own users, groups, roles, and methods. You can also integrate an external LDAP server for ASAP users, groups, roles, and methods to your WebLogic Server.

> **Note**
>
> You must ensure that the ASAP_VNO group name is defined in the external LDAP server.

You must perform the following steps to integrate an external LDAP server with ASAP:

1. Configuring an External Lightweight Directory Access Protocol Server
2. Configuring a Primary Authentication Provider in WebLogic Server

Configuring an External Lightweight Directory Access Protocol Server

To configure an external LDAP server for use with ASAP, use the following procedure:

1. Create an Administrators group in the LDAP server.
2. Create a user in LDAP that can be used as an administrator login for Oracle WebLogic Server.
3. Add the newly created user to the Administrators group.
4. Create the following user groups for ASAP:
   - **ASAP_Guests**
   - **ASAP_Monitors**
   - **ASAP_Operators**
   - **ASAP_VNOS**
   - **ASAP_WS_USERS_GROUP**
   - **Cartridge_Management_WebService**
   - **everyone**

> **Note**
>
> You may provide custom group names in the LDAP server corresponding to the seven groups mentioned in step 4.

5. Create users in the LDAP server and add them to the groups created in step 4.

For detailed instructions on configuring LDAP, see the LDAP documentation specific to your LDAP Authentication provider.

Configuring a Primary Authentication Provider in WebLogic Server

To configure the external LDAP server in WebLogic Server:

1. Configure the following authentication providers on the WebLogic Remote Console:
   - Default Authentication provider
   - LDAP Authentication provider

   See the WebLogic Server documentation for information on configuring authentication providers.
2. Set the control flag as follows:
   - For Default Authenticator provider, set **SUFFICIENT**.
   - For LDAP Authentication provider, set **REQUIRED**.

   See the WebLogic Server documentation for information on configuring authentication providers.
3. Reorder the authentication providers.

   > **Note**
   >
   > Ensure that the Default Authentication provider is above the LDAP Authentication provider.

   See the WebLogic Server documentation for information on reordering authentication providers.

Managing WebLogic Server ASAP User Security

These procedures apply to user security that is maintained using WebLogic Server and ASAP, and assume that **myrealm** is the only active security realm. These procedures do not support other realms supported by WebLogic Server, such as the LDAP realm. When an administrator configures WebLogic Server with security realms other than **myrealm**, all features described in this section are disabled, including the change password menu in the OCA client.

ASAP administrators can configure user password policies through the WebLogic Remote Console and the Password Policy Utility page. Password policies defined in the WebLogic Remote Console include:

- Minimum password length
- Lockout enabled

- Lockout duration
- Reset duration
- Lockout cache

Password policies defined in the Password Policy Utility page include:

- Password aging
- Password expiration warning period
- Enabling/disabling password policies

If your ASAP installation includes the OCA client, observe the restrictions described in Table 1-5.

Table 1-5 ASAP Client Password Restrictions

Feature	OCA Client	Description
Password change	Supported	If the administrator has enabled password policies, users must change passwords in the OCA client.
Password change at first login	Supported	If the administrator has enabled password policies, users must change passwords in the OCA client.
Password length	Supported	Passwords must be between 6 and 21 characters in length.
Password aging	Supported	If the administrator has enabled password policies, users must change passwords in the OCA client.
Password syntax	Supported	Password syntax.
Reuse of previously-used passwords	Supported	Enforced when the user specifies a new password in the OCA client.
Lockout features	Supported	Lockout features.

The procedure for changing end user passwords is contained in the *ASAP Order Control Application User's Guide*.

Configuring the WebLogic Server Change Password Utility Page

Secure Shell (SSH) must be enabled on the WebLogic Server in order for the **Password Policy** and **Change Password Utility** Java server pages (JSPs) to be reachable.

To enable SSH on the WebLogic Server:

1. Log in to the WebLogic Remote Console.
2. Click **Edit Tree** if not already selected. See the WebLogic Server Online Help for more information.
3. Under the **Environment** drop down menu, select **Servers**.
4. In the **Configured Servers**, click the name of the server to be configured.
5. Select **Save**.
6. Provide an appropriate SSL listen port number in the **SSL Listen Port** field (that is *WLS_SSL_PORT* from the Installation Values in the *ASAP Installation Guide*). Ensure that this is not the same as the server listen port (that is *WLS_PORT*).
7. Go to the **Shopping Cart** and select **Commit Changes**.
8. Restart your WebLogic Server.

The **Password Policy** page and **Change Password Utility** page are JSPs accessed through a web browser at the following URLs:

- **https://***hostname:WLS_SSL_PORT***/security/PasswordPolicy.jsp**
- **https://***hostname:WLS_SSL_PORT***/security/ChangePassword.jsp**

Note that the **SSL Listen Port** (configured above) is used, **not** the *WLS_PORT* value.

> **Note**
>
> You must configure the **BEA_WLS_HOST** and **BEA_WLS_PORT** variables of these JSPs in **ASAP.cfg**. The ASAP installer populates these variables automatically during the installation process.

Setting WebLogic Server ASAP Password Policies

Administrators must set password policies using both the WebLogic Remote Console and the **Password Policy** page.

To define lockout conditions:

1. In the WebLogic Remote Console, click **Edit Tree** if not already selected. See the WebLogic Server Online Help for more information.
2. In the **Security** drop down menu select **Realms**.

   The Summary of Security Realms screen appears.
3. In the **Realms** list, click the name of the security realm to be configured.

   The Settings screen for the selected realm appears.
4. Select the **User Lockout** tab to access user lock-out settings.
5. On the **User Lockout** tab, complete the following fields:
   - **Lockout Enabled**: Requests the locking of a user account after invalid attempts to log in to that account exceed the specified Lockout Threshold. By default, this attribute is enabled.
   - **Lockout Threshold**: The number of failed user password entries that can be tried before that user account is locked. Any subsequent attempts to access the account (even if the username/password combination is correct) raise a Security exception. The account remains locked until the System Administrator unlocks it or another login attempt is made after the lockout duration period ends. Invalid login attempts must be made within a span defined by the Lockout Reset Duration attribute.
   - **Lockout Duration**: The number of minutes that a user's account remains inaccessible after being locked in response to several invalid login attempts within the amount of time specified by the Lockout Reset Duration attribute.

- **Lockout Reset Duration**: The number of minutes within which invalid login attempts must occur for the user's account to be locked. An account is locked if the number of invalid login attempts defined in the Lockout Threshold attribute occurs within the amount of time defined by this attribute.
- **Lockout Cache Size**: The number of minutes within which invalid login attempts must occur for the user's account to be locked. An account is locked if the number of invalid login attempts defined in the Lockout Threshold attribute occurs within the amount of time defined by this attribute.

6. Click **Save**.
7. Go to the **Shopping cart** and click **Commit Changes**.

To set password policies:

1. In WebLogic Remote Console, click **Edit Tree** if not already selected. See the WebLogic Server Online Help for more information.
2. In the **Security** drop down menu select **Realms**.

   The Summary of Security Realms screen appears.
3. In the **Realms** list, click the name of the security realm to be configured.

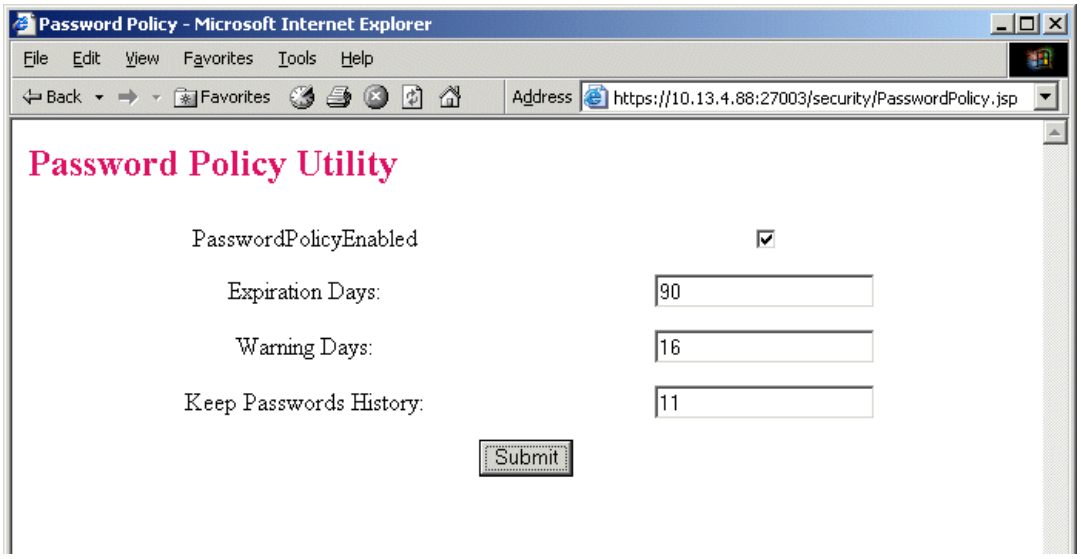
   The Settings screen for the selected realm appears.
4. Select the **Authentication Providers** drop down menu.
5. In the **Authentication Providers** list, select **DefaultAuthenticator**. The WebLogic Authentication provider is configured in the default security realm with the name **DefaultAuthenticator**. If you have configured a different authentication provider, select it instead.

   The Settings for Default Authenticator screen appears.
6. Make any required configuration including the **Minimum Password Length**.
7. Click **Save**.
8. Go to the **Shopping cart** and click **Commit Changes**.
9. Access the Password Policy Utility page by entering the following URL in your web browser:

   (**https://***WebLogic_Host:WLS_SSL_PORT***/security/PasswordPolicy.jsp**).

   The Enter Network Password dialog box appears.
10. Enter the WebLogic administrator username and password and click **OK**.

    The **Password Policy Utility** screen appears.



11. Complete the following fields:
    - **PasswordPolicyEnabled**: Enables or disables the following password security features:
        - Password change, including password change upon first login, passwords being changed to a previously used password
        - Password aging

      > **Note**
      >
      > The following security features are always enabled:
      >
      > - Lockout
      > - Password length of 6 characters or more, but no longer than 20 characters
      > - Inclusion of at least one special character and one number
      >
      > If you disable the password policy, the password history is deleted.

    - **Expiration Days**: Specifies the period of time, in days, for which the password is valid
    - **Warning Days**: The number of days prior to password expiration that the user is notified of impending password expiration. The user is prompted to change the password.
    - **Keep Password History**: The number of previously-used passwords that are tracked in history. If the password policy is enabled, users are unable to reuse passwords tracked in history.
12. Click **Submit**.

Description of the illustration asap_translations.gif

This image displays the work order translations.

Disabling the Change Password Feature in the OCA Client

Table 1-6 lists the permission that applies only to the **Change Password** menu item in the OCA client. This feature lets ASAP Administrators disable the change password feature in the OCA client.

Table 1-6 Change Password Permission ACL

ACL	Permission
ServiceActivation.ASAP.*ENV_ID***. ASAP_Provisioning_Management**	ChangePassword

If the new ACL is not defined in WebLogic Server, the **Change Password** menu item in the OCA client is disabled.

Managing Locked-out User Accounts

WebLogic Server user accounts have settings which automatically lock out the user for a period of time after a number of login attempts which fail due to an incorrect password. These settings are modifiable by the administrator.

To access WebLogic Server user settings:

Under the **Security** drop down menu, select **Realms**. In the **Security Realms** window, click the name of the security realm to be configured.

Select the **User Lockout** tab to access user lock-out settings.

See "Setting WebLogic Server ASAP Password Policies " for more details.

Settings include:

- **Lockout Threshold** – Number of failed password entries before the account is locked. The account remains locked until it is unlocked by the system administrator, or the lockout duration period ends. The default is 5 failed password entries.
- **Lockout Duration** – Number of minutes that a user's account remains inaccessible after being locked after *Lockout Threshold* invalid login attempts. The default is 30 minutes.
- **Lockout Reset Duration** – Number of minutes within which invalid login attempts must occur in order for the user's account to be locked. The default is 5 minutes.

To unlock a locked user account:

1. In the WebLogic Remote Console, click **Edit Tree** if not already clicked. See WebLogic Server Online Help for more information.
2. In the **Security** drop down menu, select **Realms**.
3. Enter the name of the user to unlock and click **Save**. In the **Realms** drop down menu, select the security realm to be configured.
4. Select the **Unlock User** tab.
5. Enter the name of the user to unlock and the click **Save**.
   Click the **Shopping cart** and then select **Commit Changes**.

> **Note**
>
> On a Managed Server, you cannot unlock a locked account through the WebLogic Remote console. The unlock information is propagated through a multicast message which is only configured in a cluster environment. Instead, use the following command:
>
> ```
> java weblogic.Admin -url url -username adminuser -password adminuserpassword
> -type weblogic.management.security.authentication.UserLockoutManager
> -method clearLockout lockedusername
> ```
>
> Alternatively, wait until the Lockout Duration time has passed.

Updating Methods Role Assigned to a Group or User in WebLogic Server

This section provides information on how to update methods in an ASAP role assigned to a group or a user in WebLogic Server. You can use the information defined in deployment descriptors to grant security roles and define security policies.

You can update the deployment descriptors by using WebLogic Workshop (Eclipse component) or by editing the **ejb-jar.xml** and **weblogic-ejb-jar.xml** files manually.

To edit the XML files manually:

1. Navigate to *WebLogic_domain***/servers/***WebLogic_server***/upload/asap***ENV_ID***/app (where the *WebLogic_domain* is the installation directory for your WebLogic Server domain, *WebLogic_server* is the name of your WebLogic Server domain, and *ENV_ID* is the ASAP environment ID.
2. Do the following:

   ```
   jar xvf asapENV_ID.ear ssam.jar
   jar xvf ssam.jar META-INF/ejb-jar.xml
   jar xvf ssam.jar META-INF/weblogic-ejb-jar.xml
   ```
3. Edit **ejb-jar.xml** and **weblogic-ejb-jar.xml** to add, remove, or modify roles to users or groups.

   For example, you may remove the following three methods from the Monitor Role in **ejb-jar.xml** file:

   ```
   <method>
   <ejb-name>ASAPSecurityServices</ejb-name>
   <method-name>SA_Aborted_Work_Order_Report</method-name>
   </method>
   <method>
   <ejb-name>ASAPSecurityServices</ejb-name>
   <method-name>SA_Activity_Report</method-name>
   </method>
   <method>
   <ejb-name>ASAPSecurityServices</ejb-name>
   <method-name>SA_ASDL_Report</method-name>
   </method>
   ```
4. Do the following:

   ```
   jar uvf ssam.jar META-INF/ejb-jar.xml
   jar uvf ssam.jar META-INF/weblogic-ejb-jar.xml
   jar uvf asapENV_ID.ear ssam.jar
   ```
5. Redeploy the **asap***ENV_ID***.ear** file.

Configuring ASAP Server and Database Credential Security

Secure data must be stored in a secure location and distributed to authorized users. The ASAP security system governs how secure data is managed and the ASAP diagnostics files are secured:

- **Secure Data Storage**

  The ASAP security administrator pre-defines the nature and accessibility of secure data for each ASAP server. Class A secure data is stored in the CSF wallet during the initial ASAP installation procedures (see the *ASAP Installation Guide* for more information). The ASAP installer distributes this wallet to each server during the ASAP installation.

- **Secure Data Encryption**

  The CSF wallet encrypts all data contained in it and obtained from it. In addition, the CSF wallet file (**cwallet.sso**) has restricted access permissions.

Many ASAP utilities and scripts use the passwords contained in the CSF wallet. For more information about how these utilities and scripts use the CSF wallet security feature, see "Using the Credential Store Factory Wallet with ASAP Utilities and Scripts."

Your ASAP security administrator creates the class A secure data while installing ASAP. Your administrator can also modify ASAP Server passwords using the ModifyPasswords script (see "Changing Database Passwords in the Credential Store Factory Wallet").

About Credential Store Factory Wallet Secure Data Management

Secure data management in ASAP involves:

- Setting up and maintaining secure data storage
- Encrypting secure data during provisioning

Setting up and Maintaining Secure Data Storage

During the ASAP installation procedure, your ASAP security administrator must enter a predefined user name and password for each ASAP server. The administrator must also enter the WebLogic Server, and Oracle database user names and passwords. ASAP stores these user names and passwords in the CSF wallet (class A secure data).

Data Encryption

The CSF wallet provides transparent encryption features that protect all credentials it stores and transmits.

Using the Credential Store Factory Wallet with ASAP Utilities and Scripts

When the ASAP security feature is configured (security level of the Control server is not 0), every ASAP utility (scripts or programs) that require access to the ASAP server or database are prompted to enter a password based on the target.

The following utilities have arguments -d for ASAP security (see Table 1-7 for more information about these arguments).

- **asap_recompile**
- **asap_utils**

- **start_asap_sys**
- **start_control_sys**
- **startc**
- **starts**
- **stop_asap_sys**
- **stopc**
- **stops**

Table 1-7 ASAP Security Arguments

Argument	Description
-d	The utilities retrieve password information from the CSF wallet. The ASAP installer creates the CSF wallet during the installation of ASAP. You can modify the passwords using the **ModifyPassword** script (see "Changing Database Passwords in the Credential Store Factory Wallet").

```
start_asap_sys -d => Retrieve password the CSF wallet
```

Changing Database Passwords in the Credential Store Factory Wallet

The **cwallet.sso** file is found in the *ASAP_Home*/**install** directory. This file contains security information for installation purposes and includes the ASAP database schema user names and passwords, the Oracle DBA user name and password, and the WebLogic Server domain user name and password. The ASAP installer creates this file during the installation process.

To change the CSF Wallet passwords, use the following procedure:

1. Source the **Environment_Profile** located in the *ASAP_Home/* directory.
2. Set the **TNS_ADMIN** LINUX variable to the location of your **tnsname.ora** file.

   ```
   export TNS_ADMIN=/home/examplelocation/
   ```
3. From *ASAP_Home*/**scripts** directory run the **ModifyPassword** tool.
4. Enter the DBA user name and password.
5. Enter and modify the ASAP database user names and passwords as required.

Configuring Security for Network Elements Communication

NE credentials (also called custom secure class B data) used primarily by NEPs to establish network connections to NEs must be stored in a secure location and distributed to authorized users.

The ASAP security tool supports the following features to protect NE credentials:

- **Secure Data Storage**

  An administrator can use the ASAP security tool to create NE credentials and store these credentials in a central repository on the Control server. The Control server distributes these credentials to SRPs, JSRPs, NEP, or Java-enabled NEPs (JNEPs).

  ASAP stores NE credentials in the Control server in the **tbl_classB_secu** database table.
- **Secure Data Encryption**

The Control server uses a symmetric secret key encryption method to achieve data confidentiality for custom secure data.

- **Key Distribution**

  The Control server acts as a key distribution server, and distributes custom secure data to every ASAP server during provisioning. To acquire the custom secure data, ASAP server has a pre-defined key distribution protocol.

Understanding the Custom Secure Data Structure

Your ASAP security administrator can define class B custom secure data through application programming interfaces (APIs) or action functions used by the customized JSRPs, NEP, and JNEPs. Your ASAP security administrator can also set up your custom secure data using the ASAP security tool.

The class B custom secure data typically includes the user names and password for the NEs and should only be used by custom NEPs, JNEPs or JSRPs.

The Control server contains the secure data that has name, value, creation date, and description fields. The Control server distributes the secure data to each ASAP server during provisioning.

Table 1-8 provides a detailed description of a secure data entry in the secure data storage. The **Security Level**, **Alg (sdu)**, and **Audit Level** fields apply only to ASAP secure data.

Table 1-8 Secure Data Entry

Field	Type	Length	Encryption	Description
Name	Char	80	No	The name field of a secure data entry. This is used as a key to retrieve the secure data entry.
Value	Char	80	Yes	The encrypted value of the secure data entry.
Security Level	Integer		No	This field is only applicable to ASAP secure data (the class field value is 1). If the name is an ASAP server name (for example NEPab12), then this controls the level of ASAP security; otherwise this is ignored. Possible values: • 0 – Turn off security feature (default). • 1 – Turn on security feature. The security level of the Control server controls the level of the entire ASAP security. The security level of each ASAP server is only applicable when the security level of the Control server is not 0.
Cache	Integer	1	No	This field is only applicable to ASAP secure data (the class field value is **1**. If the name is an ASAP server name, then this controls caching ASAP secure data; otherwise this is ignored. Possible values: • 0 – Turn Off cache feature (default). • 1 – Turn On cache feature.
Audit Level	Integer		No	Reserved for future use.
Creation Date	Date		No	The creation date.
Alg	Integer		No	The deployed secret key algorithm. The default value is 1 (AES_GCM algorithm). This applies to the ASAP secure data only.

Table 1-8 (Cont.) Secure Data Entry

Field	Type	Length	Encryption	Description
Class	Integer		No	The secure data classes are: • ASAP secure data • Custom secure data The custom JSRP, NEP, and JNEP only manipulates custom secure data.
Desc	Char	255	No	The description of the secure data entry.

For custom secure data storage, the required fields are **Name**, **Value**, **Creation Date**, and **Desc**.

Managing Custom Secure Data

Secure data management in ASAP involves:

- Setting up and maintaining secure data storage
- Encrypting secure data during provisioning
- Key distribution (for custom secure data)
- Local caching of custom secure data, for improved system performance

Setting up and Maintaining Secure Data Storage

Your administrator can set up the initial custom secure data storage repository and can predefine this custom secure data. ASAP stores this data in a central repository: the Control server.

Encrypting Data During Network Element Provisioning

To protect custom secure data during provisioning, symmetric or secret key encryption is used. You can use the ASAP APIs and action functions to:

- Retrieve ASAP secure data and custom secure data.
- Update or add custom secure data.

Securing Network Element Credentials with the Security Administration Tool

You can use the command line ASAP security administration tool to set up and maintain the secure data. You can use the tool to update the WebLogic passwords in the CSF Wallet.

```
asap_security_tool
[operation_option][-c ctrl_svr_name][-l diag_level]
[-f diag_file_name]
```

Table 1-9 lists and describes the ASAP security tool arguments.

Table 1-9 ASAP Security Administration Tool Arguments

Arguments	Description
operation_option	One of the operations in the Detailed Operation section that follows.
-c	The name of the Control server for the application (can be omitted).
-i	Initializes the secure data storage. With this option, the ASAP security administrator can set up ASAP secure data.
-l	The diagnostic level for the application. The default value is LOW_LEVEL. For more information on diagnostic levels, see "Server Diagnostic Levels."
-f	The name of the diagnostic file for the application. The default is **ASC_SECU.diag**.
-h	Displays a help page.

All arguments are optional. When the **operation_option** is omitted from the command line, a menu is provided, displaying all the available operations.

The ASAP Security Utility Script provides the following options:

```
***** ASAP Security Utility Script *****

   1.  Initialize the secure data storage
   10. Add/Modify secure data entry
   11. Delete secure data entry
   12. Query secure data entry
   20. Import secure data from file to the secure storage
   30. Change current security level
   50. Change the encryption key
   60. Upgrade the database
   100. Display usage message for this Tool

Enter Choice <Q - Quit>:
```

> **Note**
>
> Selecting option 1 to initialize the secure data storage, deletes all the security data in the class A CSF wallet and the class B table. There is no way of restoring the deleted data using the ASAP security administration tool.
>
> You can use option 1 if you want to use password encryption for class B custom secure data. Class A CSF wallet data is always encrypted.

Table 1-10 lists and describes the ASAP security tool operations.

Table 1-10 ASAP Configuration Tool Operations

Operations	Description
-i	Initializes secure data storage. With this option, the ASAP security administrator can set up class B custom secure data. **Note**: This operation is only available when ASAP is not running. **Note**: Selecting option 1 to initialize the secure data storage, deletes all the security data in the class A CSF wallet and class B table.

Table 1-10 (Cont.) ASAP Configuration Tool Operations

Operations	Description
-x	Checks and changes the current security level. **Note**: this operation is only available when ASAP is not running.

-a data	Adds or modifies a secure data entry.
	• ASAP data format:
	`name:value:class:secu_level:cache_mode:`
	`audit_level: alg:description`
	• Custom data format:
	`name:value:description`
	You can add customer secure data while the Control server is running. When the addition of data is complete, the **asap_security_tool** requests the Control server to flush cached secure data.

-d data	Deletes a secure data entry. • Data format: `name:class` You can delete customer secure data while the Control server is running. When the deletion of data is complete, the **asap_security_tool** requests the Control server to flush cached secure data.
-q data	Queries a secure data entry. • Data format `name:class`

-r filename	Imports secure data to the secure data storage. To import a large amount of custom secure data into the ASAP secure storage, compose a flat file containing the essential secure data. The data format is the same as that of adding secure data. For example, a data file may contain the following secure data. Examples of custom secure data entries:
	`TOR_NE:`*`password`*`:1:0:Class B NE login info`
	`ENG_NE:`*`password`*`:1:0:Class B NE login info`
	The value field in the data file should be clear text. The ASAP security tool encrypts the data when necessary. Lines in the file starting with "#" are treated as comments.

Additional Security Considerations

In addition to the other security features, observe the following guidelines described in this section.

Setting Secure Diagnostic Levels

The ASAP diagnostic files contain information logs on provisioning activity and confidential provisioning information such as WO parameters and NE dialog as plain text. The secure ASAP diagnostic feature addresses the following key provisioning data:

- NE dialog to control the content of diagnostic file
- WO parameter

Setting the Network Element Dialog Diagnostic Configuration Parameter

The ASAP NEP diagnostic file contains switch-sensitive information sent and received from the NE. In production environments, Oracle recommends that the audit level is set to SANE so that switch-sensitive information is not included in diagnostics files. Oracle also recommends that you delete old archives and/or store archives in a secure manner.

In addition to setting the audit level to SANE, you can also enable or disable the configurable variable, **NE_DIALOG_OFF**, in **ASAP.cfg**. This variable controls the source code to print out all NE dialog messages from NEP diagnostic file. All the output NE dialog in the NEP application checks against the value of **NE_DIALOG_OFF**, and cuts off the message if the **NE_DIALOG_OFF** is set to 1.

Table 1-11 lists and describes the **NE_DIALOG_OFF** option.

Table 1-11 NE_DIALOG_OFF Configuration Variable

ASAP Configuration Variable	Default	Description
NE_DIALOG_OFF	0	Controls the NE dialog message in the diagnostic file. Possible values are: • 0 – NE dialog messages appear in the diagnostic file. • 1 – Secure the NE dialog. No NE dialog messages appear in the diagnostic file.

Although the NE dialog can be controlled by the action statement, some NE dialogs are not related to any action function, and in this case the **NE_DIALOG_OFF** is used to hide NE information.

Setting Work Order Information Diagnostic Levels

Work orders typically contain business sensitive information. The WO is processed by several components, like Service Activation Request Manager (SARM), OCA; consequently, the WO information is exposed to different diagnostic files. For this reason, Oracle recommends that the diagnostic level be set to SANE in production environments to avoid unnecessarily exposing sensitive information. As well, any archival diagnostics files should be stored in a secure manner. For more information on diagnostic levels, see "Server Diagnostic Levels ."

In addition to setting the audit level to SANE, you can also enable or disable the configurable variable, **WO_SECURITY_PROP**, in the ASAP WO. This variable controls the source code to print out the information messages from all ASAP diagnostic files for a particular WO.

Table 1-12 lists and describes the **WO_SECURITY_PROP** option.

Table 1-12 ASAP Work Order Properties

ASAP Work Order Property	Default	Description
WO_SECURITY_PROP	0	Controls the WO message in the diagnostic file. Possible values are: • 0 – Output WO information in the diagnostic file • 1 – Secure WO information. No work order messages appear in the diagnostic file.

The WO information is output through a generic diagnostic function call, which can also output other information in addition to the WO information. A filter list controls the output. When **WO_SECURITY_PROP = 1** and the message type is contained in the filter list, there is no output to diagnostic file.

All possible output message function calls that are scattered in several ASAP applications must be examined.

> **Note**
>
> This security measure also applies to WO information that is fetched from the SARM database.

Securing JNEP to NE Connection Implementations

Since the NEP and the JNEP are designed to communicate with a variety of NEs, there are various methods used for NEP or JNEP to NE security. You can implement the Java method to retrieve a **user_id** and password from the NEP or JNEP database. The connection to the NE can then be opened with the correct identification.

Advanced Java programmers can also implement a password timeout and automatic password change functionality as requested by the switch vendor and company policies.

Setting JSRP to SARM Security Properties

Each WO within ASAP can be authorized prior to its acceptance in the SARM. The user_id and password properties are compared against the SARM database table **tbl_uid_pwd**.

- **User_id** (optional) – The user_id that the SARM uses for security validation. The user_id is required if the SARM's security validation feature is enabled in the **SECURITY_CHECK** configuration parameter. By default, the security validation feature is turned off.
- **User Password** (optional) – Indicates the password which the SARM uses for security validation. The User Password is required if the SARM's security validation feature is enabled in the **SECURITY_CHECK** configuration parameter.

Setting Security Between Servers

To increase security between ASAP servers, you must turn on SSL or TLS security in gRPC to encrypt communication between ASAP servers. You can turn on SSL in gRPC by setting the parameter **GRPC_SSL_ENABLED** in **sampleASAPConfiguration.properties** found in *ASAP_home***/config** directory. You must specify the SSL identity certificate and SSL trust certificate in pkcs12 format in **sampleASAPConfiguration.properties** file. See "Sample Configuration File" in *Installation Guide* for more information.

Enabling Schema Validation for the JSRP JMS Interface

You can enable the JSRP to validate incoming Java Message Service (JMS) XML messages at the JSRP JMS interface against the ASAP schemas that enforces xml WO message formation. Enabling schema validation helps to secure the JSRP JMS interface against invalid XML messages, but also incurs a performance impact.

To enable schema validation, set the **VALIDATION_ENABLED** parameter to **True** in the **ejb-jar.xml** file from **srp.jar** file of *Domain_home***/servers/***server_name***/upload/asap***envid***.ear**

deployment (where *server_name* is the name of your WebLogic Server instance, and *envid* is the environment ID for your ASAP instance.

2

Monitoring and Managing ASAP

This chapter describes how to monitor and manage Oracle Communications ASAP.

Overview of Monitoring and Managing ASAP

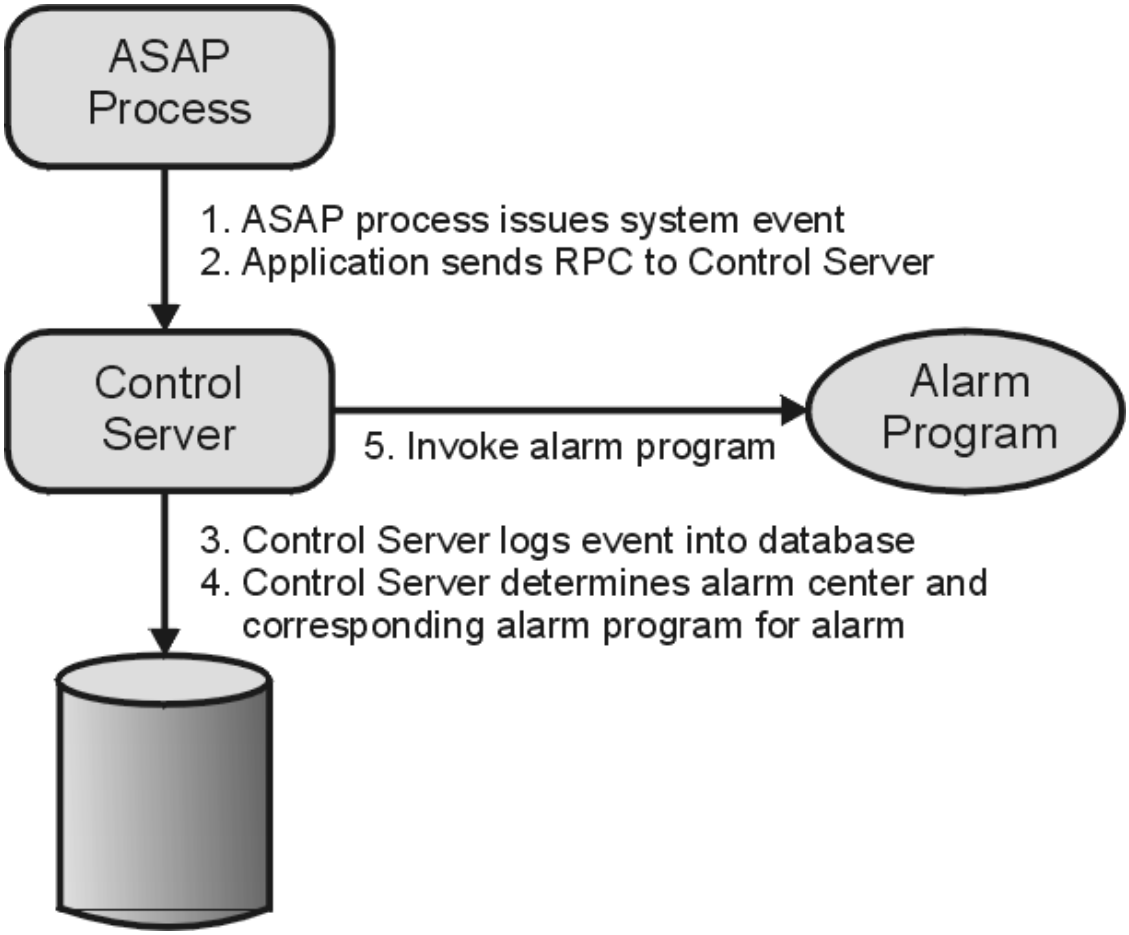
ASAP provides the following management and monitoring capabilities:

- **ASAP process monitoring** – The ASAP Control server starts, stops, and monitors all processes in ASAP. Upon startup, each ASAP application (client or server) establishes a connection to the Control server. The Control server immediately installs a disconnection handler on that connection. When the connection is lost, a callback function is triggered in the Control server, indicating an abnormal termination of the process. A controlled shutdown of ASAP applications does not trigger this callback. When the process termination is detected, the Control server immediately attempts to restart the process (if configured).
- **ASAP database administration process** – Each ASAP server can perform administration tasks within its application database. At a user-configured time, the server administration thread connects to its primary database to perform database archiving, purging, and integrity checks. This thread is also responsible for executing database administration commands to optimize all query plans for the database procedures. This activity ensures that ASAP database performance is not affected by large changes in data volumes. See "Managing the Database and File System" for more information.
- **ASAP diagnostic and system events** – ASAP is equipped with diagnostic call and system activity events that monitor its internal health. ASAP events can be mapped to various alarm categories to generate alarms viewable through a user-defined management console. For more information, see "Configuring System Events and Alarms Using Stored Procedures."

Configuring System Events and Alarms Using Stored Procedures

System events range from informational events to critical error events. When an ASAP client or server generates a system event, the event is transmitted to the Control server which then saves the event. The Control server determines if the event requires an alarm to be triggered. The Control server determines the alarm centers that are mapped to the alarm and the alarm program to call. This alarm program can be a shell script, a Linux executable, or a Linux executable to send the alarm to email or print.

Figure 2-1 Alarm Process



The Control server also monitors each application process within ASAP. If an application process terminates, the Control server issues an Abnormal Process Termination event that is mapped to a user-defined alarm program. Abnormal Process Termination events are written to both the application diagnostic file and the database.

Each event can be configured to enable or disable a system alarm. Every alarm has an associated alarm level (such as MAJOR, MINOR, or CRITICAL) and Auto Clear flag. The Auto Clear flag specifies that the alarm should be automatically cleared upon generation. If the alarm is configured as non-auto clearing, it is generated every five minutes until you reset it using the **asap_utils** administrative remote procedure call (RPC) interface (function 60). If configured as auto-clearing, the alarm is generated only once.

For more information on asap_utils, see *ASAP Server Configuration Guide.*

An alarm can map to up to five alarm centers which trigger alarm programs to alert users of the system alarm. Typically, such alarms can email an administrator, print reports to an alarm center, page a particular individual, or communicate with a network management system.

Configuring Alarm Centers for Alarm Notification and Escalation

Alarm centers can be pagers, email accounts, printers, network management systems, and so forth. Alarm centers are notified of alarms by means of a Linux program, or shell script.

To create an alarm center, specify the Linux program, or shell script that ASAP runs when the alarm is generated. For example, assume there are two alarm centers: a general ADMIN center, which receives messages on a printer, and another center called ADMINPGR, which receives pager notifications. To notify these alarm centers, you must develop two shell scripts or programs.

Alarm center information is contained in **tbl_alarm_center**. Alarm centers are later mapped to system alarms in **tbl_system_alarm**. If you are defining a system event that does not map to an alarm, proceed to "Configuring System Events."

Use the following stored procedures to define, list, or delete alarm centers:

- **CSP_new_center** – Defines an alarm center to which alarm notifications must be sent by the Control server.
- **CSP_list_center** – Lists an alarm center definition from the control database.
- **CSP_del_center** – Deletes an alarm center definition from the control database.

For more information on these stored procedures and **tbl_alarm_center**, refer to the *ASAP Developer's Guide*.

Configuring System Alarms

System alarms identify system events, such as component malfunction. System events can be mapped to operations which enable, disable, or log alarms.

Alarms in ASAP can be routed to multiple alarm centers, based on the time of day.

The types of alarms that can be configured in ASAP are:

- **Continuous alarm** – The continuous alarm is issued periodically until your system administrator manually disables it or ASAP issues a system event to disable the alarm.
- **Single initiation alarm** – The single initiation alarm is generated once each time the system event is issued.

Table 2-1 lists and describes alarms and what action is required in response to the alarm.

Table 2-1 System Alarms

Alarm	Action
Abnormal Process Termination	This alarm indicates that a component of ASAP has shut down. You can immediately save the log file and take the appropriate steps to restart the process.
General System Error	This is a general purpose system alarm. The description of the error from the **event** call should give sufficient details of the error. If ASAP is otherwise running properly, you can simply note this alarm and deal with it at a convenient time.
General System Warning	A general warning event, not an error situation. This event can be used to indicate errors or exceptions in external systems, for example, Port Bind Failure, and Network Element (NE) in Maintenance Mode. You can note this alarm and deal with it at your convenience.
General System Information	System information such as thread spawning, process startup, and graceful shutdown, work order (WO) time-outs, Host CLLI in Busy State. These alarms typically do not require user intervention.

Table 2-1 (Cont.) System Alarms

Alarm	Action
Process Termination Error	This error is generated by an application process whenever an internal condition is encountered. This causes the application process to log the system event in the database and then terminate the application process in an orderly manner. This is a critical error. You must immediately determine the problem and restart the process.
Operating System Disk File Error	This is a critical error. This error is generated when an error is encountered when creating, reading, or writing an operating system file. For example, the Network Element Processor (NEP) is unable to create a file to insert the NE response log, and the Service Request Processor (SRP) is unable to create a file for WO completions. You must deal with this alarm immediately.
Critical Database Resource Error	This can be a serious error. The event is issued by RPC calls if a database or transaction log is full, the database is corrupt. You must attend to this alarm immediately, although you can continue to use the system.
Invocation Application Remote/Registered Procedure Failed	This alarm is issued if the RPC or registered procedure fails. The text in the **event** gives the invocation details. If ASAP is otherwise running properly, you can note this alarm and address it at your convenience.
Linux System Call Error	This is a critical error that is issued if an application encounters a Linux system call error. You must deal with this alarm immediately.
Application Network Error	This is a serious error. (Unable to connect to the SQL/ Application server, connection to server goes down). You should save the log file immediately and deal with this situation.
Open Server Application Object Access Error	This error is issued if ASAP is unable to create, remove, lock, unlock, or access Open Server object (Mutex and message queue). If ASAP is otherwise running properly, you can note this alarm and deal with it at your convenience.
Work Order In Progress Longer Than Specified Threshold	System information event. Indicates that a WO has been in progress longer than a specified threshold time. The description of the error from the **event** call provides information on the error. If ASAP is otherwise running properly, you can note this alarm and deal with it at your convenience.
Work Order Routing Error	This error is generated when the Service Action Request Manager (SARM) is unable to correctly determine the Host NE to which a WO should be routed. The description of the error from the **event** call provides the routing data (Directory Number (DN) or MCLI). Updates to the routing tables are required.
Network Element Has Gone Into Maintenance Mode	An informational alarm. This alarm is generated when an NE enters maintenance mode. If the alarm persists, investigate the reason.

Each system event can be optionally mapped to a system alarm in **tbl_system_alarm**.

You can specify the system alarm level: MINOR, MAJOR, or CRITICAL. Each alarm is either self auto-clearing or non-auto-clearing, and may or may not define one or more alarm centers to which the alarm is to be transmitted.

If the alarm is configured as non auto-clearing, it is generated every five minutes until you reset the alarm. If the alarm is configured as an auto-clearing alarm, it is only generated once.

After you have defined the alarm centers for the system, you can define the alarms. An alarm code is used to identify uniquely each alarm in ASAP. You can configure an alarm level for each alarm and up to five different alarm routes. The alarm routes are used to specify daily time intervals for an alarm to be generated at up to five alarm centers and to specify the time period in minutes between a regeneration of the alarm.

Use the following stored procedures to define, list, or delete system alarms:

- **CSP_new_alarm** – Defines an alarm within ASAP, and optionally, the associated “alarm code" that may be associated with the event.
- **CSP_list_alarm** – Lists system alarm codes and their definitions.
- **CSP_del_alarm** – Deletes a system alarm code.

For more information on these stored procedures and **tbl_system_alarm**, refer to the *ASAP Developer's Guide*.

Configuring System Events

In ASAP, each application can generate system events from within the application code. In addition, you can configure system events to generate an alarm in certain circumstances (for example, Common Service Description Layer (CSDL) failure, SARM to SRP notifications and so on) using certain static tables within ASAP.

Each system event is configured in the control database and can be mapped to a particular system alarm. Each alarm can, in turn, be mapped to one or more alarm centers. Each alarm center can then run a user-supplied alarm program to perform the relevant user-determined alarm notification.

> **Note**
>
> You can define system events in custom code. The system events you define do not have to be mapped to core system events. To generate alarms, ensure that the event you have defined is mapped to an alarm in the Event/Alarm Configuration function.

Defining System Events

Core and customer-specific subsystems can generate system events. System events are defined in the control database table **tbl_event_type**.

Events can be associated with an alarm code. In addition, the event can enable or disable the alarm.

Use the following stored procedures to define, list, or delete system events:

- **CSP_new_event** – Defines an event type within ASAP, and optionally, the associated “alarm code" that may be associated with the event.
- **CSP_list_event** – Lists database threshold definitions.
- **CSP_del_event** – Deletes a database threshold definition.

For more information on these stored procedures and **tbl_event_type**, refer to the *ASAP Developer's Guide*.

Sample Alarm Program - alarm.sh

You can refer the sample alarm Linux shell script in *ASAP_home***/scripts/alarm.sh**. You must copy this script from the *ASAP_home***/scripts** directory to the *ASAP_home***/programs** directory for a control server alarm center to use it.

You can configure custom alarm scripts or programs for each alarm center, where the script's logic determines the actions to be taken based on the alarm severity.

For example, minor alarms can write messages in a log file; whereas major alarms can write messages in a log file and also send email notifications.

The log file **ASAP_Alarm_Log** for alarm event is populated under *$LOGDIR*.

Sample Alarm Output

This section contains sample alarm output.

```
**************************************************************************
ASAP Alarm Issued @ Wed Aug 7 21:00:29 ADT 2002
Alarm Program = /ASAP/PRODUCTION/programs/alarm.sh
**************************************************************************
Arguments:
Event Id  = 71457
Alarm Name = ABNORMAL
Event Code = ABNORMAL
Event Desc = Abnormal Process Termination - Application
Event Text = Warning: Abnormal Termination of Process LU62SEND
Source File = ProcessManager
Source Line = 701
Alarm Level = CRITICAL
Application = CONTROL

Additional Parameters =
** End of Alarm **
```

The output specifies the time and date of the alarm, the script called by the alarm, together with the following information:

Table 2-2 Alarm Output

Alarm Name	Description
Event ID	Unique ID for the event that generated the alarm.
Alarm Name	Alarm code associated with the system event.
Event Code	ASAP event generated by the application.
Event Desc	Brief description of the event.
Event Text	Brief description of the reason for the system event within the source code.
Source File	Line in the source file where the event was generated.
Source Line	Source file name where the event was generated.
Alarm Level	Possible values: MINOR, MAJOR, or CRITICAL.
Application	Logical name of the ASAP application server that generated the system event.

Understanding Default System Events

The static table **tbl_event_type** contains the system events that the ASAP application can generate and, if required, the system alarm code associated with that event. You can modify existing system events in this table or add custom events.

For information on adding alarms and events, see "Configuring System Alarms" and "Configuring System Events."

Each system event must have a record in **tbl_event_type**.

The following tables contain the system events that are contained in **tbl_sys_event** and are generated by the application. You can update the static text, alarm level, and description of these system events and add custom events.

API System Events

Table 2-3 lists and defines the system events included in the core application programming interface (API).

Table 2-3 Core API System Events

Event	Static Text	Alarm Level	Description
ABNORMAL	Abnormal process termination as the application process terminated unexpectedly.	Critical. Non auto-clearing.	Event issued by the Control server if any process (server or client) it monitors has terminated unexpectedly.
DISK_ERR	Operating system disk file error.	Critical. Auto-clearing.	Event issued when an error is encountered creating, reading, or writing an operating system file.
NTWK_ERR	Application network connection error.	Minor. Auto-clearing.	Unable to connect to SQL/Application server, connection to server gone down, etc.
RPC_ERR	Invocation of application remote/ registered procedure failed.	Minor. Auto-clearing.	Event issued if RPC/Reg Proc fails. The text in event gives the invocation details.
RPCSPACE	Critical database resource error.	Critical. Non auto-clearing.	Event issued RPC calls if the database full, transaction logs full, database corrupt, etc.
SRVOBJER	Open Server Application object (Msg Queue, Mutex, etc.) access error.	Minor. Auto-clearing.	Unable to create, remove, lock, unlock, or access Open Server Object (Mutex, message queue, etc).
SYS_ERR	General system error.	Major. Auto-clearing.	General purpose system error. The description of the error from the event call gives sufficient details of the error.
SYS_INFO	General system information notification.	None.	System information such as thread spawning, process startup, and graceful shutdown, WO Timeouts, Host CLLI in Busy State, and administrative flushing of data from memory.
SYS_TERM	Process termination event.	Critical. Auto-clearing.	This event is called by an application process when an internal condition is encountered. The event causes the application process to log the system event in the database and then terminate the application process in an orderly manner.
SYS_WARN	General system warning.	Minor. Auto-clearing.	Warning event, not an error situation. This event can be used to indicate errors or exceptions in external systems, for example, Port Bind Failure, Host LU6.1/LU6.2 Bridge Down, and NE in Maintenance Mode.
LINUX_ERR	LINUX system call error.	Minor. Auto-clearing.	Event issued if application encounters Linux system call error. For example, the NEP is unable to create a file to insert the NE response log or the SRP is unable to create a file for WO completions.

SARM System Events

Table 2-4 lists and describes the events that the SARM can issue.

Table 2-4 SARM System Events

Event	Static Text	Alarm Level	Description
ROUT_ERR	WO routing error.	Major. Auto-clearing.	Notifies when the SARM process is unable to determine the correct NE to which a particular WO should be routed. The routing data (MCLI or DN) is included in the event text.
WOINPROC	WOs in progress longer than specified threshold.	Informational. Determined by individual customer.	Informs when one or more WOs are in progress for more than the specified threshold time.

The Java Request Processor

The customer's service request systems communicate with the SRPs. If the same provisioning request is received from more than one source, it is translated by an SRP into identical sets of Common Service Description Layer (CSDL) also called service action commands and parameters.

Service action commands are independent of the originating system. The communications capability, the standard representation of services, and the multiprocess deployment of the SRP allow the same ASAP system to adapt to new customer service request sources without the need to change the existing service delivery flow.

After the SRP has translated the service and action combinations into service action commands, the SRP extracts service, action, and associated parameters from the native order format. It determines parameters for the work order that constitute the header portion of the ASAP work order (such as due date, order number, and priority). In addition, the SRP finds the global parameters for the work order that are required for provisioning.

When the translation is complete, the SRP sends the ASAP version of the work order to the Service Activation Request Manager.

NEP System Events

Table 2-5 lists and describes the events that the NEP can issue.

Table 2-5 NEP System Events

Event	Static Text	Alarm Level	Description
BIND_ERR	Port binding error event.	Minor. Auto-clearing.	Unable to allocate device to connect to NE if, for instance, the maximum number of connections to the NE is exceeded. Each device in ASAP has a command processor thread that is always running. When there is a connection request for an NE, the session manager tries to obtain an enabled and unbinded (available) device. If the session manager cannot obtain such a device, the session manager will throw a BIND_ERR event.
CONN_ERR	NE connection error event.	Minor. Auto-clearing.	NE connection attempt failed.

Table 2-5 (Cont.) NEP System Events

Event	Static Text	Alarm Level	Description
DIAL_ERR	Dialup error event.	Minor. Auto-clearing.	The dial-up program to connect to NE has failed. After a connection is established to a dialup type device, but before the NE_LOGIN (or LOGIN) is performed. If the DIALUP fails, then DIAL_ERR event is thrown.
LOGN_ERR	Login error event.	Major. Auto-clearing.	The login program to the NE has failed. The LOGIN is run after NE connection is established, or, for dialup type devices, after the DIALUP has run. If the LOGIN fails, this event is thrown.
MAINTNCE	Host NE has gone into Maintenance mode.	Informational. Determined by individual customer.	Informs when NE enters Maintenance mode. When the current Atomic Service Description Layer (ASDL) ends with MAINTENANCE exit type, then this event is thrown.
PORT_DIS	Port disabled upon connection failure event.	Major. Auto-clearing.	Connection to NE failed, port/device disabled. During connection, the NEP tries to connect to the NE. If this connection attempt fails, then the PORT_DIS event is thrown. At the same time, the port or device is disabled. After the PORT_ENABLE_TIMER (a configuration variable defined in **ASAP.cfg**) has concluded, the port is automatically enabled. Consequently, the same port will not be tried for the second connection attempt while it is disabled. Another port is selected. The port is disabled after the currently executing ASDL is completed (for example, in any state like SUCCEED(104), FAIL(253) and so on). After this ASDL completes, the device disconnects from the NE and the port is disabled. Any subsequent ASDLs is provisioned with other available ports and devices.

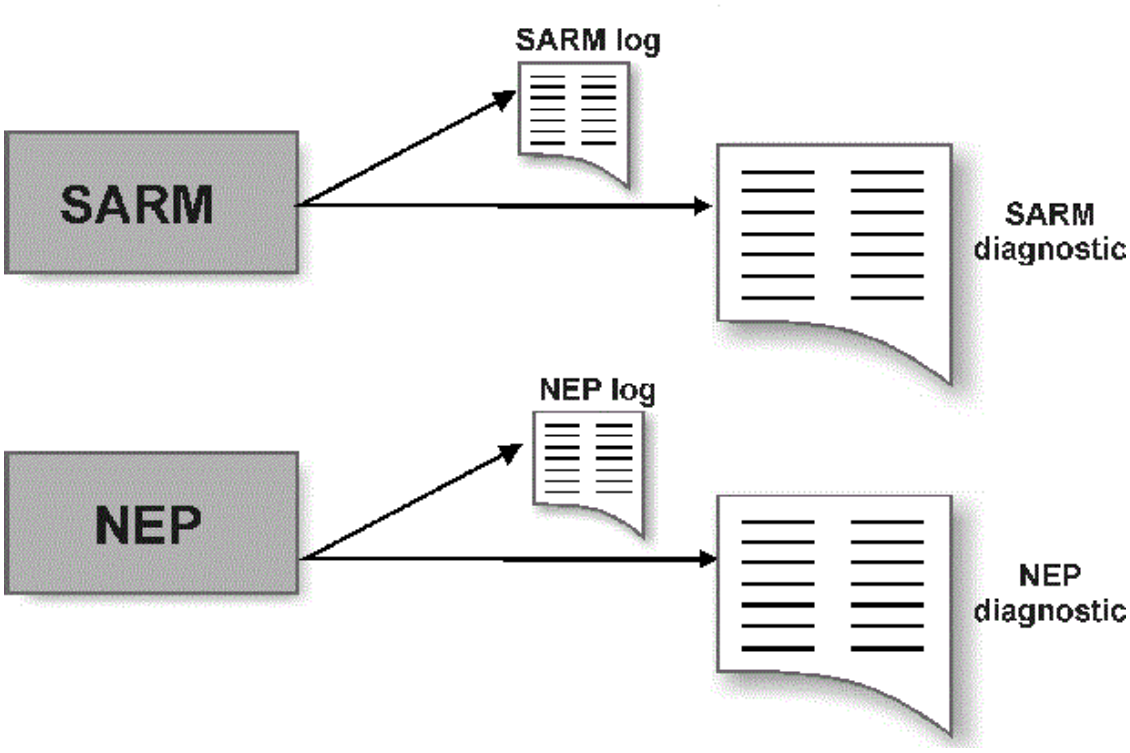
Configuring and Reading Log and Diagnostic Files

There are two sets of files maintained by ASAP applications for monitoring and troubleshooting purposes: log files and diagnostic files. These files are created by ASAP daily or whenever an application starts up.

> **Note**
>
> If you write to diagnostic or log files while ASAP is running, you will cause the associated ASAP application to malfunction.

Figure 2-2 shows the sources of log and diagnostic information.

Figure 2-2 Source of Diagnostic Information



About Log Files

Log files contain high-level error messages. These files are created by ASAP daily or whenever a server starts up. Log files are located in the directory $*LOGDIR* (where $*LOGDIR* is $*ASAP_BASE*/DATA/logs) under *appl_name***.log,** where *appl_name* is the appropriate client or server application.

Every diagnostic call made in ASAP has an associated diagnostic level denoting the importance of the diagnostic message. A specific diagnostic level is also applied to every application process so that only messages with a level greater than, or equal to, the configured level are written to the diagnostic file. This arrangement facilitates the configuration of the diagnostic logging without recompiling the application. For more information on diagnostic levels, see "Server Diagnostic Levels ."

Every application runs with the diagnostic level specified in the Application Process Table. The diagnostic level of an application server can be modified dynamically while the server is running by means of an administrative RPC sent using the **asap_utils** RPC interface.

About Diagnostic Files

Diagnostic files include all low-level activities for each application. These files are created by ASAP daily or whenever a server starts up. Diagnostic files are located in the directory $*LOGDIR/yyyymmdd* under the name *appl_name***.diag**. where *yymmdd* is the date for which you want to view the diagnostics and *appl_name* is the appropriate client or server application.

The diagnostic file rolls over to a new file whenever the maximum diagnostic file size is reached. This file size is configurable using the **MAX_DIAG_FILE** configuration parameter. You

can also configure the number of diagnostic files to be created. This field, together with the maximum file size, limits the amount of disk space that may be used by diagnostic files. When the file limit is reached, the API closes the existing file and opens a new one.

The value of the **MAX_DIAG_FILE** parameter may depend on the level of diagnostic messages written by the application and the available disk space.

Configuring and Reading WebLogic Server Log and Diagnostic Files

ASAP uses log4j to generate and manage the log messages from ASAP's WebLogic Server components. It does not affect any logs generated by ASAP's (SARM, NEP, Control (CTRL) server) components. The generated log messages are stored in the **asap.log** file located in *WebLogic_domain*.

Through the use of log4j, you can develop and maintain a logging strategy that minimizes the overall impact of logging operations on the application's resources. log4j does this by letting you control the volume of log messages generated.

You also have the ability, when necessary, to dynamically change the level and detail of the logged messages. This feature helps you to, for example, increase the level and detail of logged messages to help analyze performance problems within a production environment.

To read more about log4j, refer to the Apache website:

http://logging.apache.org/log4j

Defining Severity Levels

To control the volume of log messages generated and written to the output destinations (the console and the WebLogic Server log file, which are referred to as Appenders by log4j), you assign severity levels to the various areas of the application that generate their own, discreet messages (these areas are referred to as Categories by log4j. If an event occurs inside a given category that triggers a message below the severity level assigned to the category (that is, less severe than the assigned level), log4j does not generate the message.

Example

If you assigned the severity level Warning to a given category, log4j does not generate any messages for that category that are flagged in the ASAP code as Debug or Info level messages. log4j will, however, generate all messages that are flagged as Warning, Error, and Fatal.

You can further control the number of messages written to the output destinations, or Appenders, by also assigning a severity level to them. When you assign a severity level to an Appender, it rejects messages below that severity level, even if log4j passes the message to it.

Example

If you configure the console to the Warning severity level, and one of the Categories is configured with the Info level, the console will not display the Info message, even though log4j generates the message and passes it on to the console, because the message's severity level falls below the threshold for which the Appender is configured. If, however, that same Category later generates an Error level message, the console will accept and display the message, because it carries a severity level equal to or higher than the console's threshold.

By default, the console and the WebLogic Server log file accept all error messages, from the least severe to the most severe.

The Levels

The following is an ascending list of the severity levels, starting with the least severe:

- Debug – (Least severe) Designates fine-grained informational events that are most useful to debug an application
- Info – Designates informational messages that highlight the progress of the application at coarse-grained level
- Warning – Designates potentially harmful situations
- Error – Designates error events that might still allow the application to continue running
- Fatal – (Most severe) Designates very severe error events that will presumably lead the application to terminate

Configuring the Severity Levels

There are two methods by which you can configure the severity levels:

- log4j.xml – The **log4j.xml** file is located in **asap$***ENV_ID***.ear:APP-INF/lib/asaplibcommon.jar**. This method allows you to modify the log4j configuration (like log message format, name of the log file, volume of the log files, how many are saved, and so on) and requires you to stop, then restart the servers before the changes will to take effect. For more information, see "Configuring the log4j.xml File."
- log4jAdmin – The log4jAdmin web page lets you to dynamically change the severity levels while the servers are running. The changes take effect immediately. As the changes are persisted to the database the severity levels remain at the new level when you restart the server. For more information, see "Using the log4jAdmin Web Page."

The logging levels configured in the database take precedence; all other configuration is controlled by the **log4j.xml** configuration file embedded inside **asap$***ENV_ID***.ear:APP-INF/lib/asaplibcommon.jar**.

Configuring the log4j.xml File

Use this method to modify the log4j configuration as described in "Configuring the Severity Levels." To change the log4j configurations later, you must stop the server, modify the **log4j.xml** file, then restart the server. In most cases, therefore, if you need to modify the logging levels, you should use the log4jAdmin web page, as described in "Using the log4jAdmin Web Page."

There are two sections of the **log4j.xml** file that you need to look at when configuring this file:

- Appenders – This section defines the output destination for the messages. At installation, the Appenders section contains two entries:
  - Console – From this entry, you control the level of messages that the WebLogic Remote Console accepts.
  - WebLogic – From this entry, you control the level of messages that the WebLogic Server log file accepts.
- Categories – This section contains references to all of the ASAP categories that generate messages and gives you the ability to control the level of messages they generate.

To configure the **log4j.xml** file:

1. Unpack the **asap.ear** or the **srt.ear** file, depending on the application for which you are configuring logging parameters.
   - The **asap.ear** file is found in the $ASAP_BASE**/lib** directory. After the **.ear** file is unpacked, unpack **APP-INF/lib/asaplibcommon.jar**. The **log4j.xml** file is located in the root directory.
   - The **srt.ear** files is found in the $ASAP_BASE**/SRT/lib** directory. After the **.ear** file is unpacked, unpack **APP-INF/lib/asaplibcommon.jar**. The **log4j.xml** file is located in the root directory.
2. In the Appenders section of the **log4j.xml** file, search for the following string, which appears at the top of the Appenders section:

   ```
   <!-- Append messages to the console -->
   ```

   The Console entry governs what level of messages are written to the console.
3. If necessary, change the threshold level. By default, it is configured to DEBUG, allowing the console to display all messages sent to it. If you want to restrict the number of messages displayed, change the threshold entry to the severity level appropriate for your installation (see "Severity levels" above, for a description of the severity levels).

   In the example below, the level is changed from DEBUG to INFO.

   Before

   ```
   <param name="Threshold" value="DEBUG"/>
   ```

   After

   ```
   <param name="Threshold" value="INFO"/>
   ```
4. In the Appenders section, search for the following string:

   ```
   <!-- Append messages to the weblogic's log file-->
   ```

   The WebLogic Server log file entry governs what level of messages are written to the WebLogic Server log file.
5. If necessary, change the threshold level as described in Step 3.
6. Go to top of the file and search for the following string:

   ```
   category name
   ```

   This takes you to the first category entry in the file.
7. Review each of the categories in this section, changing the severity level where necessary. In the example below, the level is changed from INFO to WARNING.

   Before

   ```
   <category name="com.mslv.system"> <priority value="info"/> </category>
   ```

   After

   ```
   <category name="com.mslv.system"> <priority value="warning"/> </category>
   ```
8. When you finish updating the categories, save and close the **log4j.xml** file.
9. Repack the **.ear** file.
10. Redeploy the **.ear** file:
    - For Java Service Request Processor (JSRP):

      In $ASAP_BASE**/lib**, run the following commands:

```
ModDeployDescriptor -u <WebLogic Admin Id>
java weblogic.Deployer -adminurl http://$WLS_HOST:$WLS_PORT
-user $WLS_USER -password $WLS_PASSWORD -name asap$ENV_ID -remove
java weblogic.Deployer -adminurl http://$WLS_HOST:$WLS_PORT
-user $WLS_USER -password $WLS_PASSWORD -upload
-source $ASAP_BASE/lib/asap$ENV_ID.ear -name asap$ENV_ID
-targets $TARGET_WLS_SERVER -activate
```

- For SRT:

  In $*ASAP_BASE*/**SRT** run the following commands:

  ```
  ant -f install.xml undeploy
  ant -f install.xml deploy
  ```

Using the log4jAdmin Web Page

Use log4jAdmin web page to check the current logging levels or to change the logging levels dynamically.

> **Note**
>
> You can only use this method to change the severity levels of the Categories. To change the Appender's levels, you must reconfigure the **log4j.xml** file. See "Configuring the log4j.xml File" for an explanation of the Categories, the Appenders, and how to configure the XML file.

The changes you make to the logging severity levels using this method are persisted to the database in the table **tbl_code_list** on the Control server.

> **Note**
>
> Because log4jAdmin is bundled with the core, it shares the core session timeout configuration.

Checking the Current Logging Levels

You can use the Filter Loggers feature at the top of the page to check the logging level of specific categories or subcomponents. If you know the name of the category or subcomponent that you want to check, you can use the filter to display only that category, or related, categories.

To check logging levels:

1. Open the log4jAdmin web page by entering the following path in the browser's address line (the URLs are case sensitive):
   - **JSRP:**

     http://*Weblogic_Host:WebLogic_PORT/ASAP_ENVID*/**log4jAdmin.jsp**

     or

     https://*Weblogic_Host:WLS_SSL_PORT/ASAP_ENVID*/**log4jAdmin.jsp**
   - **SRT:**

http://*Weblogic_Host:WebLogic_PORT/ASAP_ENVID***/SRT/log4jAdmin.jsp**

or

https://*Weblogic_Host:WLS_SSL_PORT/ASAP_ENVID***/SRT/log4jAdmin.jsp**

2. In the **Filter Loggers** field, enter the beginning of the name or a part of the name.
3. Do one of the following:
   - Click **Begins With** to filter on the beginning of the name.
   - Click **Contains** to filter on part of the name.

     The list displays the categories and subcategories that match the entry in the Filter Loggers field.
   - (Optional) To change the logging level do the following:
     a. Scan the entries in the left-hand column and find the category or sub-component which you want to change.
     b. Scan across the row to the severity levels. The level that currently is selected is highlighted in a different color from the other levels and appears in the **Effective Level** column.
     c. Click the logging level which you want to change:

        To change an entire category, click the category name.

        To change the subcomponent, click the sub-component name.

        The change takes place immediately.
     d. When you finish making the changes, close the page.

Enabling Stored Procedure Error Messages

To enable stored procedure error messages for a sqlplus session, use the following procedure:

1. From a Linux terminal, source your ASAP *ASAP_home*/**Environment_Profile**.

   ```
   . ./Environment_Profile
   ```
2. Log into your sqlplus session for the database you want to run stored procedures on. For example:

   ```
   sqlplus $CTRL
   Enter password: password
   ```

   Where *password* is the password for your ASAP server database schema.
3. Run the following command to enable stored procedure error messages.

   ```
   set serveroutput on;
   ```

   For an error message example, see the following error message in bold:

   ```
   var retval number;
   exec :retval := SSP_new_comm_param('T',
   'TEL_HOST','COMMON_DEVICE_CFG','HOST_USERID','asapXXX','userid');
   Host TEL_HOST For Device Type T And Parameter HOST_USERID Does Not Exist, No
   Comm Param Inserted.
   .
   PL/SQL procedure successfully completed.
   .
   SQL> print retval;
   ```

```
.
    RETVAL
----------
         0
```

Managing ASAP Metrics

ASAP provides a sample Grafana dashboard that can be used to visualize ASAP metrics available from a Prometheus data source. ASAP relies on Prometheus to scrape and expose these metrics.

See the following topics for further details:

- Configuring Prometheus for ASAP Metrics
- Viewing ASAP Metrics Without Using Prometheus
- Viewing ASAP Metrics in Grafana
- Exposed ASAP Metrics

Configuring Prometheus for ASAP Metrics

Configure the scrape job in Prometheus by updating the **prometheus.yml** file as follows:

```
scrape_configs:
  - job_name: 'asapmetrics'
    scrape_interval: 120s
    scrape_timeout: 60s
    metrics_path: /ENV_ID/OrderMetrics
    scheme: http/https
    basic_auth:
      username: WebLogic user name
      password: WebLogic password
    static_configs:
      - targets: ['hostname:port number']
    tls_config:
      insecure_skip_verify: true
    params:
         query: [all]
```

Where

- *WebLogic user name* is the user name of WebLogic Server
- *WebLogic password* is the password of WebLogic Server
- *hostname* is the name of the machine on which ASAP is installed
- *port number* is the port number or SSL port number on which WebLogic is listening

> **Note**
>
> The filter options are: **all**, **today**, and **total**.
>
> If you use filter, update `query: [filter]` in the above yml file.
>
> If you do not use filter, comment out `params: query: [filter]` in the above yml file.

Viewing ASAP Metrics Without Using Prometheus

The ASAP metrics can be viewed at:

`http://hostname:port/ENVID/OrderMetrics`

This only provides metrics of the server that is serving the request. It does not provide the consolidated metrics for the entire cluster. Only Prometheus Query and Grafana dashboards can provide the consolidated metrics.

Viewing ASAP Metrics in Grafana

ASAP metrics scraped by Prometheus can be made available for further processing and visualization. ASAP comes with sample Grafana dashboards to get you started with visualizations.

Import the dashboard JSON files from **$ASAP_CNTK/samples/grafana** into your Grafana environment. See *ASAP Cloud Native Deployment Guide* for more information.

The sample dashboard displays the following:

- Work order count by order state
- Completed work order count in a configured interval

Exposed ASAP Metrics

ASAP provides the following metrics for monitoring based on the work order status:

- Completed
- Completed in last interval
- Initial Work Orders
- Failed
- Cancelled
- In progress

Work order metrics can be queried with different parameter values. Use the following URL to query the work order metrics:

http://*host*:*port*/*env_id*/OrderMetrics?query=*parameter*

The supported parameter values are:

- **total**: Provides total work order count. This is the default parameter used.
- **today**: Provides todays total work order count.

- **last_interval**: Provides completed orders in the last interval count along with total order metrics.
- **all**: Provides all the three work order counts (total + today + last_interval).

The following ASAP metrics are exposed via ASAP Servlet APIs.

Order Metrics

The following table lists the order metrics exposed.

Table 2-6 Order Metrics Exposed via ASAP Servlet APIs

Name	Notes
asap_wo_complete_total	The total work orders in the completed state.
asap_wo_initial_total	The total work orders in the loading state.
asap_wo_failed_total	The total work orders in the failed state.
asap_wo_cancelled_total	The total work orders in the canceled state.
asap_wo_inprogress_total	The total work orders in the in progress state.
asap_wo_complete_last_interval	The total work orders which are in the completed state in the last interval.
asap_wo_complete_today	The total work orders which are in the completed in the current date.
asap_wo_failed_today	The total work orders which are in the failed state in the current date.
asap_wo_cancelled_today	The total work orders which are in the canceled state in the current date.

3

Improving ASAP Performance

This chapter describes ways to improve Oracle Communications ASAP performance.

About Improving ASAP Performance

This chapter is intended to aid those who have prior knowledge of the ASAP configuration and Linux operating systems. Before starting the tuning exercises described in this chapter, you should be familiar with the following items:

- Location of ASAP diagnostic files and the Linux utilities that are used to view and manipulate them such as grep, tail, pg, top, vmstat, sar, prstat, glance (on HP), and so on.
- Location of the ASAP configuration files (**ASAP.cfg**, **ASAP.properties**, **Environment_Profile**, **NEP.jinterpreter**, **config.xml**, **startWebLogic.sh**), how to use an editor such as **vi**, how to modify the configuration files, the layout of configuration files, for example, server specific versus global variables.
- How to use Linux utilities, such as **top** and **sar** to monitor the resources being used by ASAP.

For more information, consult your system's online documentation about Linux utilities or the ASAP documentation.

Troubleshooting and Monitoring ASAP Performance

The WebLogic Remote Console can be used to monitor the Java Service Request Processor (JSRP).

In case of errors, while running stored procedures by the ASAP servers:

Increase the value of the configuration variable, **APPL_POOL_SIZE** to make more connections available.

If a stored procedure fails, the thread running the procedure goes to sleep for the time determined by the **RPC_RETRY_SLEEP** configuration parameter. ASAP then tries to run the procedure with the number of times determined by the **RPC_RETRY_COUNT** configuration parameter. All of these attempts may fail. Since the thread cannot be released for the entire duration of this error retry process, poor performance is reported, increasing the number of connections in the use and long waiting times.

Manually Tuning ASAP Performance

The performance of an ASAP system is governed by the available hardware, installation, and configuration decisions made during the initial installation phase. Due to the multi-threaded nature of ASAP, fine-tuning the system will help you to obtain the maximum benefits from the allocated resources.

This section provides you with tools and guidelines to tune your ASAP system in a short period of time. It covers the following topics:

- A recommended approach to tuning.

- A list of system limits that must be monitored to ensure that they are not exceeded during tuning.
- Guidelines to tune the JSRP, SARM, and NEP processes.

Tuning Guidelines

If you wish to go beyond the provided pre-tuned configurations, there are many ways to tune an ASAP system. However, the following technique has been verified by the internal Oracle Communications testing team. You can use it to optimize a simple ASAP configuration in less than half a day. A simple configuration consists of all ASAP components residing in the same system with small numbers of individual components, for example, fewer than five NEPs and one or two SRPs.

The following steps are the order in which the tuning process is carried out:

1. Setting a Target

Select a performance target that is based on realistic throughput (work orders (WOs) per second) or resource consumption early in the process. Without a goal, an iterative process, such as tuning, could continue indefinitely.

2. Using Simple Work Orders

To achieve consistent results, use simple familiar WOs during the tuning process. Use a repeatable test and pick a scenario such as batches, or workflows. Once tuning is complete, verify the performance with realistic data.

3. Starting with Minimum Configuration Values

Start with minimum configuration values because it is easier to detect and correct bottlenecks than it is to determine where excess resources are being consumed.

4. Following Work Order Flow

The tuning process follows the same flow that WOs take through the system. Tuning starts at the SRP (that is JSRP depending on your implementation) proceeds through to the SARM and then to the NEP (that is Java-enabled NEP (JNEP) depending on your implementation) and on to the NE, then it returns back through the same steps.

5. Checking for Bottlenecks

Bottlenecks that can develop as resources are shifted among the components which make up an ASAP system. Bottlenecks may occur in areas that were previously optimized. When you move to a new area of the system, you should review the servers that have already been tuned to ensure that their configurations have remained optimal. For example, if you tune the NEP after the SRP and SARM have been optimized, review the SRP and SARM after you have finished tuning the NEP to ensure that they have remained optimized.

Setting System Limits

During the tuning process, you must change configuration variable settings to levels that are higher than their defaults. These increases have two direct effects which you must monitor during the tuning process:

- Increased demands are placed on the hardware allocated to the ASAP system. You must use a utility, such as **top** to continually monitor the ASAP system in order to ensure that it is not consuming more resources than planned.

- If system limits are exceeded, increased ASAP resource consumption may cause errors to be reported in the systems diagnostic files. Monitor the diagnostic files closely during the tuning process so that these limits can be altered to higher values when required.

This section details errors which may appear in the diagnostic files and the configuration variables used to control system limits. Configuration variables are located in the **ASAP.cfg** file. The following are the configuration variables used for tuning.

- **APPL_POOL_SIZE**
- **CONTROL_POOL_SIZE**
- **MAX_CMD_DBPROCS**
- **MAX_CONNECTIONS**
- **MAX_CORE_DBPROCS**
- **MAX_MSGPOOL**
- **MAX_MSGQUEUES**
- **MAX_SERVER_PROCS**
- **MAX_THREADS**
- **MAX_ORDERS_IN_PROGRESS**
- **WO_AUDIT_LEVEL**

For more information on configuration variables, refer to the chapter describing configuration parameters in the *ASAP Server Configuration Guide*.

Audience

This document is intended for business analysts, planners, system administrators, system integrators, and other individuals who must understand ASAP.

Tuning ASAP Server Message Queues

The following sections contains the ASAP servers that can be tuned:

- JSRP/SRT
- SARM
- NEP
- JNEP
- WebLogic Server domain

Tuning JSRP Message Queues

The purpose of tuning the JSRP is to provide WOs at a rate that creates an even flow to the downstream SARM process.

Figure 3-1 illustrates the schematic flow of the JSRP.

Figure 3-1 JSRP Message Queues

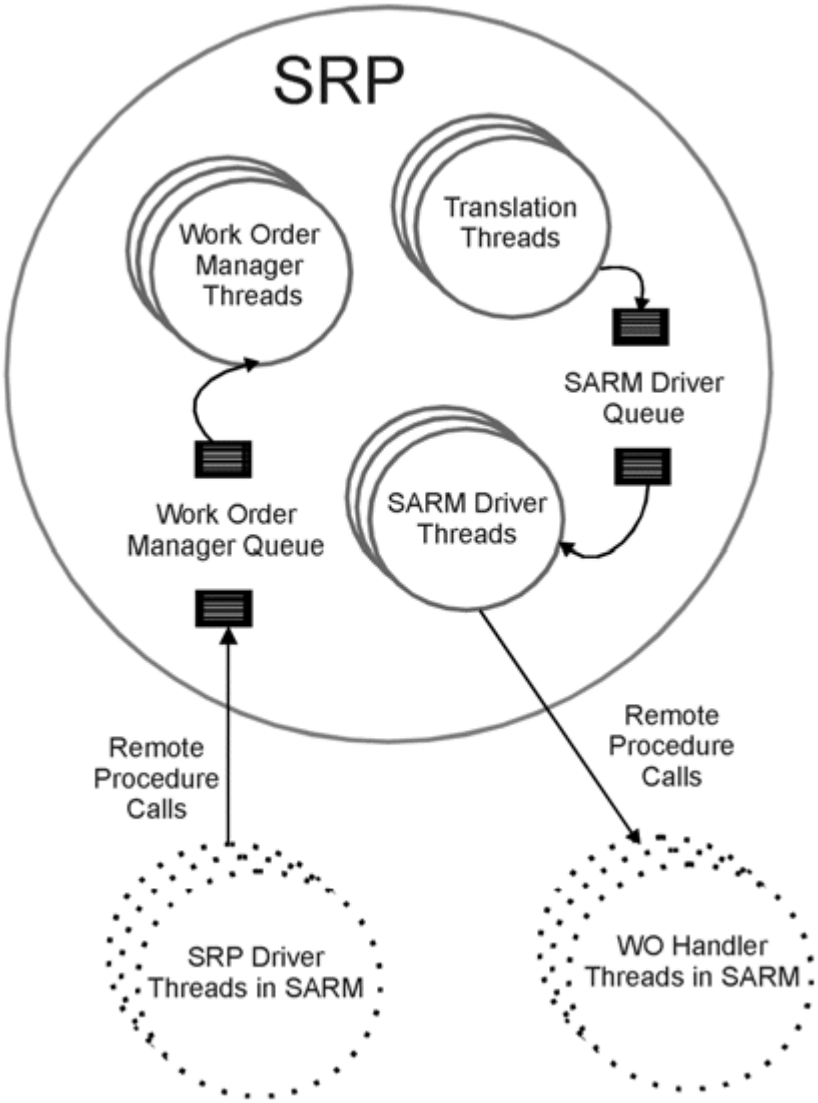


Table 3-1 lists and describes the WO manager queue.

Table 3-1 Work Order Manager Queue

Item	Description
Parameter Controlling Message Addition Rate to Queue	Number of SRP Driver Threads in the SARM. **MAX_SRP_DRIVERS**
Parameter Controlling Message Removal Rate from Queue	Number of WO Manager Threads **MAX_WO_MGRS**

Table 3-2 lists the SARM driver message queues.

Table 3-2 SARM Drive Queue

Item	Description
Parameter Controlling Message Addition Rate to Queue	Number of Translation Threads (implementation dependent).
Parameter Controlling Message Removal Rate from Queue	Number of SARM Driver Threads **MAX_SARM_DRIVER** Variable size: • small: 5 • medium: 10 • large: 25

Tuning SARM Message Queues

The purpose of tuning the SARM is to:

- Provide the Atomic Service Description Layer (ASDL) commands to the NEPs.
- Send event notices back to the SRPs at an even rate to both the upstream and downstream processes.

Since only one SARM process exists in an ASAP system, the performance of the SARM cannot be enhanced by spreading the load across multiple Central Processing Units (CPUs) or systems. Therefore, the SARM must be well tuned to get high performance from ASAP. The **WO_ACCEPT** and **WO_STARTUP** events are turned off by default to improve SARM performance. You can turn the events on by setting **wo_start_evt** and **wo_accept_evt** to **NULL** in database table **tbl_asap_srp**. See "tbl_asap_srp" in *Developer's Guide* for more details.

Figure 3-2 illustrates the flow of messages through the SARM.

Figure 3-2 SARM Message Queues

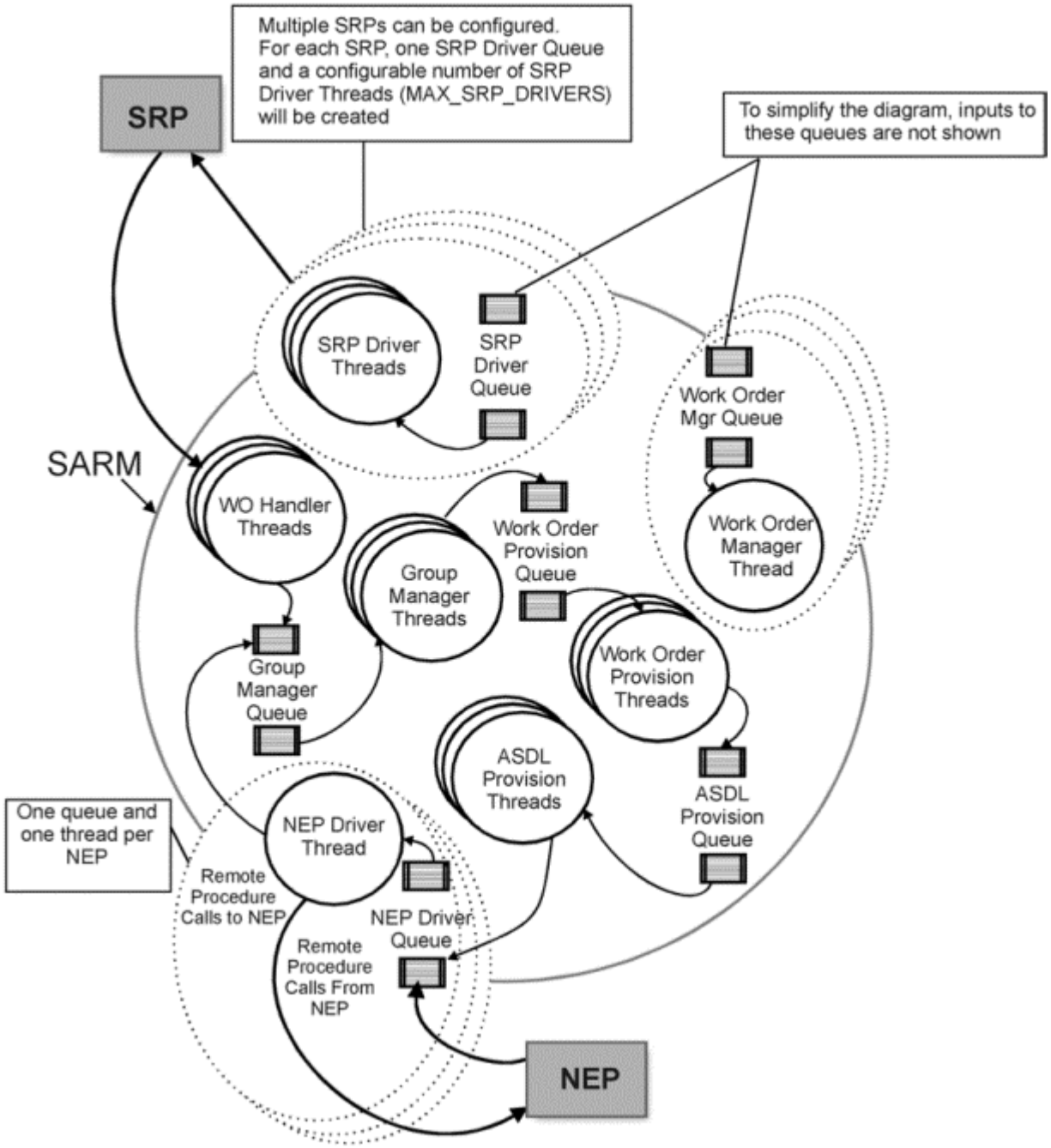


Table 3-3 lists and describes the group Mgr message queue.

Table 3-3 Group Mgr Message Queue

Item	Description
Parameter Controlling Message Addition Rate to Queue	The number of WO Handler threads and number of NEPs in the system are fixed for the purpose of Group Manager message queue tuning. Do not configure them at this time.

Table 3-3 (Cont.) Group Mgr Message Queue

Item	Description
Parameter Controlling Message Removal Rate from Queue	Number of Group Manager Threads. **MAX_GROUP_MGRS**

Table 3-4 lists and describes the WO Mgr message queues.

Table 3-4 Work Order Mgr Message Queues

Item	Description
Parameter Controlling Message Addition Rate to Queue	The number of WO Handler Threads and Number of NEPs in the system are fixed for the purpose of WO Manager message queue tuning. Do not configure them.
Parameter Controlling Message Removal Rate from Queue	Number of WO Manager Threads **MAX_WO_MGRS**

Table 3-5 lists and describes the WO provision queue.

Table 3-5 Work Order Provision Queue

Item	Description
Parameter Controlling Message Addition Rate to Queue	Number of Group Manager threads **MAX_GROUP_MGRS**
Parameter Controlling Message Removal Rate from Queue	Number of WO Provision threads **MAX_WO_HANDLERS**

Table 3-6 lists and describes the ASDL Provision Message Queues.

Table 3-6 ASDL Provision Message Queues

Item	Description
Parameter Controlling Message Addition Rate to Queue	Number of WO Provision Threads **MAX_WO_HANDLERS**
Parameter Controlling Message Removal Rate from Queue	Number of ASDL Provision Threads **MAX_PROVISION_HANDLERS**–(Less) **MAX_WO_HANDLERS**
Example	If you have five (5) WO Handlers and you want ten (10) ASDL Provision Threads, set the **MAX_PROVISION_HANDLERS** to fifteen (15). The difference is the ten (10) that you wanted.

Table 3-7 lists and describes the NEP driver message queues.

Table 3-7 NEP Driver Message Queues

Item	Description
Parameter Controlling Message Addition Rate to Queue	Number of ASDL Provision Threads **MAX_PROVISION_HANDLERS** (less) **MAX_WO_HANDLERS**

Table 3-7 (Cont.) NEP Driver Message Queues

Item	Description
Parameter Controlling Message Removal Rate from Queue	Number of NEPs in the system (dependent on throughput requirements and machine resources).

There is one NEP Driver Queue for each NEP in the system.

Table 3-8 lists and describes the SRP driver message queues.

Table 3-8 SRP Driver Message Queues

Item	Description
Parameter Controlling Message Addition Rate to Queue	Nearly every thread in the SARM can add messages to this queue. Therefore, it is not possible to control the number of messages that are added.
Parameter Controlling Message Removal Rate from Queue	Number of SRP Driver Threads. **MAX_SRP_DRIVERS**

There is one SRP Drive Queue for each SRP in the system.

Tips for Tuning the SARM

To tune the SARM, use the following:

- The number of WO threads (**MAX_WO_HANDLERS**) in the SARM must be equal to the sum of all SARM Driver Threads (**MAX_SARM_DRIVER**) in all of the SRPs.
- The total number of configured handler threads (**MAX_WO_MGRS**) in all SRPs must be equal to the driver threads (**MAX_SRP_DRIVERS**) in the SARM.
- All event notifications not used by any customized SRP implementation must be turned off.
- Sanity level diagnostics during production should be used.
- The configured number of provision handlers (ASDL and WO) must be no less than the sum of the number of handler threads (**MAX_WO_HANDLERS**) and the number of operating NEP servers. Start tuning with a ratio of 1:3 of WO Manager Threads to provision handlers (ASDL and WO).

Tuning NEP Message Queues

The purpose of tuning a NEP is to provide:

- ASDLs to the NEs
- Remote Procedure Call (RPC) responses back to the SARM at a rate that does not cause excessive buildup in any of the SARM Notify, Session Manager, or NE Command queues.

Figure 3-3 illustrates the schematic flow of the NEP.

Figure 3-3 NEP Message Queues

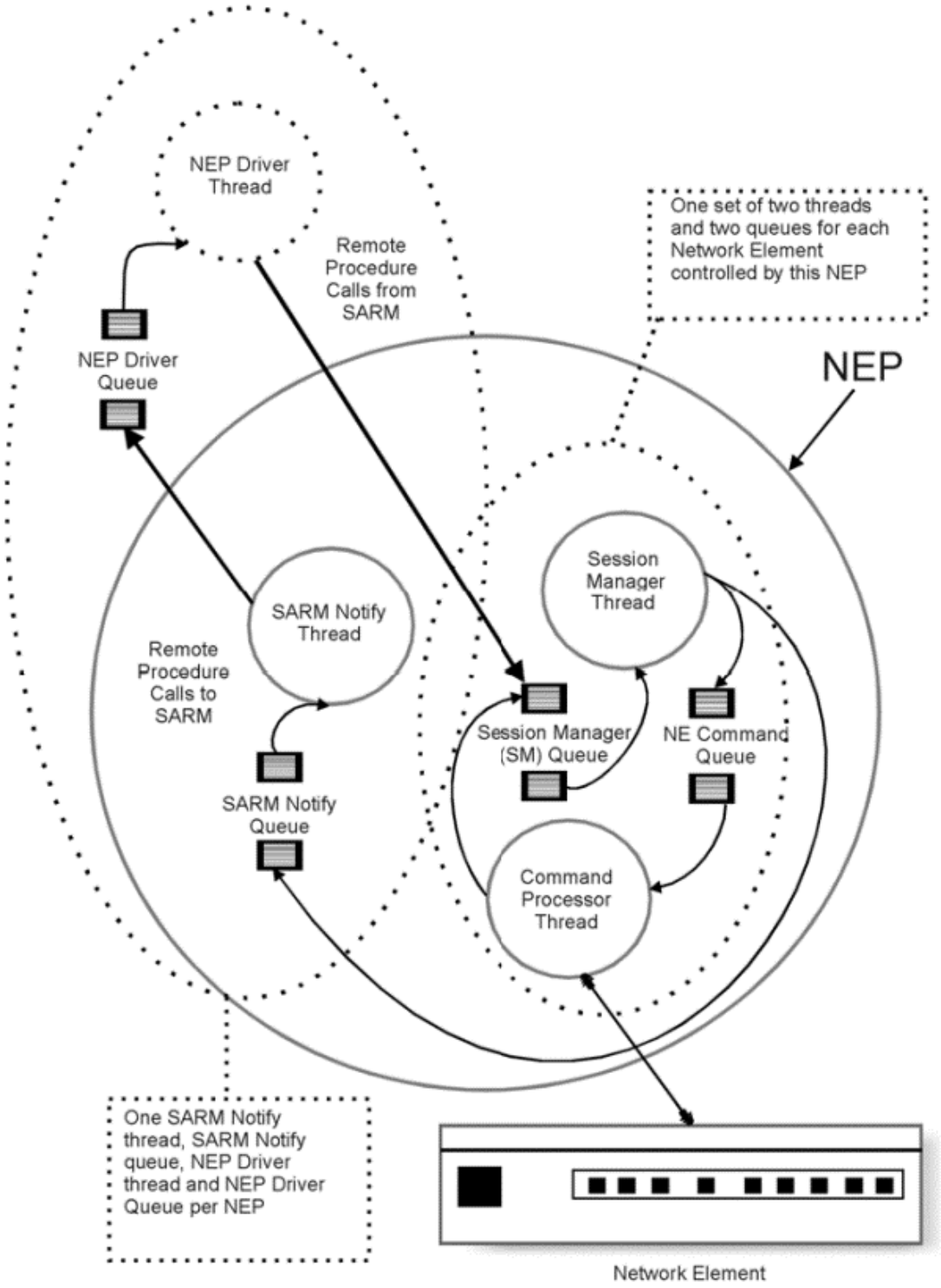


Table 3-9 lists and describes the SARM notify message queue.

Table 3-9 SARM Notify Message Queues

Item	Description
Tool	NEP Server
Parameter Controlling Message Addition Rate to Queue	Session Manager Thread for the NE. Non-configurable
Parameter Controlling Message Removal Rate from Queue	SARM Notify Thread for the NEP. Non-configurable

You can manage this queue indirectly by decreasing the number of NEs supported by the NEP (usually by increasing the absolute number of NEPs, if machine resources permit) or by balancing the load between NEPs (by moving a busy NE from a busy NEP to a less busy NEP).

Table 3-10 lists and describes the NE command queue.

Table 3-10 NE Command Message Queues

Item	Description
Tool	NEP Server
Parameter Controlling Message Addition Rate to Queue	Session Manager Thread for the NE. Non-configurable
Parameter Controlling Message Removal Rate from Queue	Command Processor Thread for the NE. Non-configurable

You can manage this queue indirectly by increasing or decreasing NE response times (faster response times decrease the length of the queue).

Table 3-11 lists and describes the session manager queue for the NE.

Table 3-11 Session Manager Message Queues for the NE

Item	Description
Tool	NEP Server
Parameter Controlling Message Addition Rate to Queue	Command Processor Thread of the NE - NEP Driver thread in the SARM for the NEP. Non-configurable
Parameter Controlling Message Removal Rate from Queue	Session Manager Thread for the NE. Non-configurable

You can manage this queue indirectly using the NE response times (fast responses increase the length of the queue). However, depending on communication and switch technology, this may not be configurable.

Tips for Tuning NEP

To tune the NEP, use the following:

- Unless machine resources are limited, target for a ratio of 50 NEs per NEP.

- If possible, group similar NE technologies and switch software loads on a single NEP. This reduces the number of ASDL programs held in memory because each NEP caches its own copy of every ASDL command that it has been asked to perform.
- Balance the load on each NEP by distributing your busy NEs across several NEPs. You must determine the expected load based on both the number and complexity of the ASDL commands being serviced.

Other Performance Issues

You must take into consideration other factors that can affect performance. This section covers the diagnostic levels and query optimization that you need in the tuning process.

The topics in this section include:

- Local versus NFS-Mounted File Systems for Diagnostic Files
- Server Diagnostic Levels
- Diagnostic Messages Output
- Query Optimization
- Table Partitioning

Local Versus NFS-mounted File Systems for Diagnostic Files

ASAP diagnostic files on NFS-mounted file systems increase network traffic and slow down disk I/O. For production systems, the log directories should be local, not NFS mounted.

Server Diagnostic Levels

Low-level server diagnostic levels increase disk I/O. Oracle recommends that you set the diagnostic level for the production system at either **PROGRAM_LEVEL** or **SANITY_LEVEL**.

In the Control server database table **tbl_appl_proc**, set the diagnostics level (column **diag_level**) to **SANE**.

Diagnostic levels set in the **tbl_appl_proc** are persistent through reboots. For more information on **tbl_appl_proc**, see *ASAP Developer's Guide*.

To set a diagnostic level temporarily, use **asap_utils** parameter **107**. See the *ASAP Server Configuration Guide* for more information.

The Table 3-12 provides the default diagnostic levels

Table 3-12 Server Diagnostic Levels

Diagnostic Level	JAVA Code Value	Description
KERNEL _LEVEL	**KERN**	Used by the kernel to generate diagnostic messages. It is only to be used by the core libraries for very low-level debugging of core code.
LOW_LEVEL	**LOW**	Used by the application to generate low-level diagnostic messages from any of its functions. Such messages should enable the programmer to debug an application. Once debugged, the diagnostic level of the application should be elevated above **LOW_LEVEL**.
FUNCTION_LEVEL	**FUNC**	Used by the application at the beginning and end of each function to track the operation of the application. This is generally not used in the core application.

Table 3-12 (Cont.) Server Diagnostic Levels

Diagnostic Level	JAVA Code Value	Description
RPC_LEVEL	**RPC**	Used by the application to produce remote procedure call (RPC) diagnostic messages.
SANITY_LEVEL	**SANE**	Used by the application for high-level diagnostics. This level of diagnostic messages provides user information about the processing of the system.
PROGRAM_LEVEL	**PROG**	Only error messages will be logged. This is primarily used to generate error messages when the ASAP system is running in a high performance production environment.
FATAL_LEVEL	**SHUT**	Used for fatal error conditions if the process is terminated. You only use this level if an error condition occurs within the application so that if the application were to continue, more errors would occur and compound the problem. For instance, if a stored procedure is missing from the database, then the application terminates and manual intervention is required.

The most commonly used diagnostic levels are **LOW_LEVEL**, **SANITY_LEVEL**, and **PROGRAM_LEVEL**.

Diagnostic Messages Output

Output of diagnostic messages can be written to disk line-by-line or buffered by the LINUX I/O subsystem. Buffered output results in diagnostics being written to disk in pages, which results in optimal performance.

> **Note**
>
> Oracle highly recommends that you do not use diagnostic line flushing for production systems. Flushing diagnostic messages to disk in lines results in high disk I/O frequency.

To disable diagnostic line flushing, set the configuration parameter **DIAG_LINE_FLUSH** (in the common application programming interface (API) section of the **ASAP.cfg** file) to "**0**." The default value is **1**.

Query Optimization

Oracle RDBMSs use a cost-based query optimizer to determine query access paths. The optimizer is primarily influenced by table and index statistics (number or rows, data distribution, etc.) available to it in the system catalogue tables. These statistics can be updated manually (usually when the amount and distribution of data in a table changes significantly) using utilities provided by the RDBMS. Keeping these statistics current is extremely important since the optimizer will make default assumptions in the absence of these statistics. The quality of the database optimizer decisions depends on the accuracy of these statistics, and hence, directly affects the performance of the ASAP application.

Updating the statistics on large tables (over 1 million rows) can take a long time. This can affect your online response time or induce rollover to small error messages.

> **Note**
>
> When the Oracle statistics are collected on the SARM schema, you should also collect the histograms on the **TBL_WKR_ORD**.**WO_STAT** column.

4

Backing Up and Restoring ASAP Files and Data

This chapter describes Oracle Communications ASAP backup and restore strategies for the file system and the database.

About Backing Up and Restoring Files and Data

As an ASAP system administrator, you should observe the following guidelines for backing up and restoring your ASAP system and databases:

- ASAP System Backup and Restore
- Database Backup and Recovery

ASAP System Backup and Restore

Backup the ASAP environment, Oracle WebLogic Server, and Oracle database schemas in the following situations:

- After you successfully install ASAP
- Before you upgrade ASAP in case you need to rollback your upgrade
- After you successfully upgrade ASAP

See the *ASAP Installation Guide* for procedures to backup your ASAP system.

In addition to these ASAP system backup scenarios, you should also follow the database backup schedule and procedures described in "Database Backup and Recovery."

See the section about rolling back your ASAP system in the *ASAP Installation Guide* to restore your ASAP system. These roll back procedures described in the *ASAP Installation Guide* will return your ASAP system to the initial installation state and post or pre upgrade state. To recover lost data, restore your database backups described in "Database Backup and Recovery."

Database Backup and Recovery

ASAP application databases must be protected in the event of corruption or failure.

The Service Activation Request Manager (SARM) and Service Request Processor (SRP) databases should be backed up daily. The Control and Network Element Processor (NEP) databases can be backed up less often since they contain only static and monitoring data. The schedule of transaction dumps is dictated by the amount of space available for the transaction logs.

The Administrative database does not need to be backed up because it only contains performance statistics.

Database Backup Strategy

Backup procedures must be performed periodically and should be fully automated by the System Administrator and Database Administrator. Cron scripts can be written which will dump the ASAP application databases, and associated transaction logs to disk and then to tape. In the case of Oracle, these cron scripts should backup to tape all data files, the control file, and archive logs to tape. For Oracle databases the Database Administrator should first decide whether to employ hot backups of the tablespaces or a full cold backup. It is recommended that a cold backup be done at least once a week in addition to a full database export. The backup strategy you choose will be determined primarily by the amount of data loss that can be tolerated in the event of failure.

The ASAP databases are as follows:

- **Control** – Can be recreated from the data on the switches, but may be backed up if required.
- **NEP** – Can be recreated from the data on the switches, but may be backed up if required.
- **SARM**- Should be backed up daily.

ASAP WebLogic Server Domain Back Up

You can archive domain configurations:

- Automatically to other media (such as tapes or disks)
- Manually by copying files to another directory, and preferably, other machines

You should regularly back up the following:

- Configuration data
- Security data
- Deployed applications

Configuration Data

Back up configuration data, including the **config.xml** file and the **config.xml.booted** file from the *Domain_home***/config/** directory. The **config.xml.booted** file is created when you successfully start the administration server and can serve as backup in the event that the **config.xml** file becomes unusable. You can overwrite the **config.xml** file with the **config.xml.booted** file to restore the domain to the last successful startup configuration.

Security Data

Back up the Administration Server security data for a domain. The security data that you should archive includes:

- Security Configuration Data
- WebLogic LDAP Repository
- **SerializedSystemIni.dat** and Security Certificates

For more information on administering domains, refer to *Oracle WebLogic Server documentation*.

5

Managing the Database and File System

This chapter provides information about managing your Oracle Communications ASAP database and file system.

Overview of Managing the Database and File System

The Oracle database and file system management tasks include:

- Configuring the operating system kernel and Oracle database initialization parameters for optimal system performance.
- Monitoring database segment and file system size by defining maximum thresholds for both.
- Using Oracle database management tools to maintain and tune your database.
- Enabling automated ASAP database administration features.
- Using scripts to purge the Oracle database tables, clear system events, alarms, and process information, and, in a test environment, to periodically purge all data.

Configuring Kernel and Database Initialization Parameters

This section outlines some important issues for implementing ASAP. These issues are listed in point form for easy scanning. For a complete reference on the subjects, refer to the appropriate Oracle documentation.

- Before installing the Oracle instance ensure that the default values for kernel and resource settings for the server are set to provide enough shared memory and semaphores. Specifically, check the **project.max-shm-memory** resource.

  See the *ASAP Installation Guide* for more details.
- You can change the Oracle Server configuration parameters from their default values to suit the requirements of ASAP. To view the Oracle Server parameters, select from the **V$PARAMETER** table or look in the **INIT***sid***.ora** file for the server. Most of the changes to the parameters must be accompanied by shutting down the server and restarting it, in order for the change to take effect.
- Oracle works best with ASAP when given sufficient resources. There are four parameters that are extremely important to configure correctly: **DB_BLOCK_SIZE**, **DB_BLOCK_BUFFERS**, **SHARED_POOL_SIZE**, and **SESSIONS**.

1. DB_BLOCK_SIZE

Set the **DB_BLOCK_SIZE** – 8K is the default. ASAP benefits from a larger block size. This must be done during database creation and cannot be changed.

2. DB_BLOCK_BUFFERS

DB_BLOCK_BUFFERS is the area of the SGA that is used for the storage and processing of data in memory. As users request information, the information is put into memory. If the **DB_BLOCK_BUFFERS** parameter is set too low, the least recently used data will be flushed

from memory. Set this value between 3000 to 5000 depending on the memory limitations of your machine. A value of 5000 with a block size of 8K uses approximately 40MB of memory.

3. SHARED_POOL_SIZE

SHARED_POOL_SIZE is used to process the procedures, packages, and triggers, as well as the library and data dictionary cache. If the **SHARED_POOL_SIZE** is set too low, you will not get the full advantage of your **DB_BLOCK_BUFFERS**. Set this value between 90MB to 150MB. For most installations this should be sufficient. Again this value should be adjusted depending on the memory limitations of your machine. Setting the **SHARED_POOL_SIZE** too high can cause your system to swap excessively.

4. SESSIONS

SESSIONS determines the number of user connections that ASAP can establish. Configure 150-200 sessions for each ASAP environment, more if you are running multiple Service Request Processor (SRP) and Network Element Processors (NEPs). Be sure to allocate enough semaphores at the Linux level to accommodate the amount of sessions.

In all the cases above, the values given are examples only. The appropriate value for your installation will depend on the machine characteristics and limitations.

- Oracle, like all databases, benefits significantly by spreading the I/O among it's data files across multiple controllers and drives. You must place your **SYSTEM**, **TEMP**, and **ROLLBACK** tablespaces on separate drives and controllers from your DATA and INDEX. **SARM_DATA** and **SARM_INDEX** will benefit the most from I/O optimization since a lot of activity occurs on these files. In addition, employing optimal striping and data placement of hot files will greatly enhance your database performance. Since minimizing I/O is the primary goal of most database tuning, this point cannot be emphasized enough.

Database Management and Tuning Recommendations

- Oracle can run in two modes **ARCHIVELOG** or **NOARCHIVELOG** mode. It is very important to run your production database in the **ARCHIVELOG** mode. This ensures that you are able to recover up to the minute in case of failure. Oracle uses logs to record transaction information in order to recover from instance and data file failures. Be sure to multi-plex these redo log data files and place them on separate disks and controllers.
- Oracle offers both online and offline database backup choices. Which one you choose will depend on how much downtime your ASAP installation can tolerate. Online backups involve backing up at the tablespace level and require more planning and consideration. Offline backup involves shutting down the database and backing up the data files, control file, and archive log files to tape. While this is occurring, ASAP is down.
- Monitor your tablespaces for index and block fragmentation on a regular basis. Index fragmentation occurs due to the deletion and updating of table rows. Rebuilding the index using the **ALTER INDEX** *name* **REBUILD** (where *name* is the name of the index) can resolve this problem. Block fragmentation can occur due to **PCTFREE** and **PCTUSED** parameters set inappropriately. This results in migrated and chained rows which affect database I/O performance.
- Update your table and index statistics on a regular basis for those tables whose data distributions change dynamically. As mentioned in a previous section, the query optimizer is primarily influenced by these statistics. If a query or transaction that was running fine appears to suddenly slowdown or stop responding, chances are the table and index statistics need to be updated.

- Tune any custom SQL statements you introduce to your ASAP installation, since they involve both your application and database. To analyze your SQL statements you will need to:
  1. Add the line to your **init***sid***.ora** file: **timed_statistics=true**
  2. Restart your database.
  3. Start the sql tracing for your session using the **ALTER SESSION SET SQL_TRACE=TRUE**.
  4. Run your application
  5. Run **tkprof** against the tracefile created by your application: **tkprof** *tracefile outputfile* **EXPLAIN=***username*/*passwd*
  6. Look at the formatted output of the trace command and make sure that your SQL statements are using indexes correctly. You must run explains on the SQL statements to determine the access path they are using.

Refer to the *Oracle Tuning Guide* for more detailed information on tuning SQL statements.

- Use the **UTLBSTAT** and **UTLESTAT** scripts provided by Oracle to monitor your Oracle instance, and diagnose performance problems. These scripts are located in the *Oracle_Home*/**rdbms/admin** directory in a LINUX environment. These scripts create a report file **report.txt** which shows differences in the statistics from the time **UTLBSTAT** was submitted, until the time that **UTLESTAT** was submitted. To obtain time-based statistics from **UTLBSTAT** and **UTLESTAT**, the **TIMED_STATISTICS** parameter must be set to **TRUE**.

Statspack

Statspack is an alternative to the **UTLBSTAT/UTLESTAT** tuning scripts. **Statspack** collects more information, storing performance statistics data permanently in the database for later use in analysis and reporting. The data collected can be analyzed using the report provided, which includes an instance health and load summary page, high resource SQL statements, as well as the traditional wait events and initialization parameters.

For more information, see *Oracle 11g Database Performance Tuning Guide* and *Reference Guide*.

Enabling Automated ASAP Database Administration Options

The Control application programming interface (API) provides a background thread that performs database administration tasks within ASAP. At a user-configured time of day, every application server process connects to its primary database and calls a user-defined function that performs the following tasks:

- Gathers customer-defined statistics related to ASAP processing.
- Archives data about to be purged from the database, if required.
- Purges data older than a date and time specified by the user.
- Performs an “orphan" purge of orphaned records to confirm the database's integrity.

The ASAP configuration variable, **DB_ADMIN_ON**, enables and disables the database administration procedures. This feature is of value to users who may have many application servers defaulting to the same database and do not want them performing database administration procedures independently on it.

Purging the Database and File System

To keep disk usage at an acceptable level the ASAP administrator should regularly purge the diagnostic and log files generated by ASAP, as well as the work orders (WOs) received from the service order system. Database purging is controlled by database administrator stored procedures that are automatically run daily.

The diagnostic files are located in the directory *ASAP_home***/DATA/logs** (where *ASAP_home* is the ASAP installation directory). The flat files of service orders are usually located in the *ASAP_home***/DATA** directory in a dated file.

Purging the Database

Database purging can be performed for the SRP, Service Activation Request Manager (SARM), and control databases, but is most commonly performed in the SARM as this is where the majority of the WO information is stored. The SRP and SARM databases maintain a history of all WOs received, while the ASAP control database maintains a history of alarms, events, performance, and process information.

The purging of the SRP and SARM is based on WO age. The purge age is usually determined by the amount of available disk space. Usually, only orders that have been completed for a certain amount of time are purged.

Database purging is controlled by database administration functions that automatically run according to the following parameters that you can configure in **ASAP.cfg**:

- **DB_ADMIN_ON** – Boolean flag. If set, it enables the database administration thread operation in the application server. This can be disabled in particular servers in situations where multiple servers share the same application database (for example, multiple NEPs) and then only one server is required to perform this database administration. Default = 1.

  > **Note**
  >
  > If you are using **purgeOrders** script for purging the SARM database, set this parameter to **0**. For more information, see "Purging the SARM Database".

- **DB_ADMIN_TIME** – The number of minutes after midnight when the database administration tasks are to be performed. This is usually performed at a time of low system activity. Default = 300.
- **DB_ADMIN_PROC** – The function the database administration thread calls at a specified time in the day. This function can be configured to perform multiple tasks, including archiving and purging dynamic data. Default = SSP_db_admin.
- **DB_ADMIN_PROC_PARAM** – The integer parameter passed to the database administration function. For example, this can specify a purge interval for a particular database. Default = 30.
- **GATHER_STATS** – Not supported. Enables the gathering of statistics for tables and indexes. Default = 0.
- **GATHER_STATS_PROC** – Not supported. Indicates the procedure to use to gather statistics on the SARM database. Default = SSP_gather_asap_stat.
- **DB_PCT_ANALYZE** – Percentage of table to analyze when gathering stats. Default = 20.

- **DB_PCT_ANALYZE_IDX** – Percentage of index to analyze when gathering stats. Default = 40.
- **GATHER_DEGREE** – Degree of parallelism to use when gathering stats. Default = 1.

> **Note**
>
> You can configure every ASAP server to run similar administration functions, but the SARM is the only server where a default function (**SSP_db_admin**) has been provided. To perform administration on databases other than the SARM database, you must copy the above configuration parameters to the appropriate section of the **ASAP.cfg** configuration file, and load the **SSP_db_admin** to those servers.

Table 5-1 provides recommended data purge frequency and methods.

Table 5-1 Database Purge Frequency and Methods

Data Object Being Purged	Selection Criteria	Frequency of Purge	Method of Purge
ASAP log files	All log files older than X days	Daily	Cron script. A sample cron script is located on "Sample Cron Script for Clearing Alarms, Events, and Process Information."
ASAP control database	All dynamic tables older than X days	Daily	Function Sample functions are described in "Sample Database Purge Script."
SARM database	All WOs older than X days	Daily	Function Sample functions are described in "Sample Database Purge Script."

Sample Database Purge Script

The following sample script can be used to delete successfully provisioned WOs from **tbl_work_ord** that are older than a specified number of days. In addition, this script calls the **SSP_orphan_purge** function, which deletes all orphaned records. When you delete WOs, information related to these WO may remain in other tables. The **SSP_orphan_purge** function purges information that is related to WOs that have been removed from the database. The affected tables include:

- **tbl_asap_stats**
- **tbl_info_parm**
- **tbl_srq**
- **tbl_srq_csdl**
- **tbl_srq_log**
- **tbl_asdl_log**
- **tbl_srq_parm**

- **tbl_srq_asdl_parm**
- **tbl_wo_event_queue**
- **tbl_wo_audit**
- **tbl_usr_wo_prop**
- **tbl_aux_wo_prop**

> **Note**
>
> The **SSP_orphan_purge** function is time-consuming and requires considerable system resources. Therefore, it should not run during peak hours.

```
create proc SSP_db_admin @days int
as
begin
    declare @cutoff_dts datetime
    if (@days is not null and @days > 0)
    begin
        select @cutoff_dts = dateadd(day, -@days, getdate())
        delete tbl_wrk_ord
        where comp_dts < @cutoff_dts and wo_stat = 104
    end
    exec SSP_orphan_purge
end
```

You can customize this script in a variety of ways (for example, you can delete WOs that have successfully completed (104) or orders that have completed and failed due to timeout). When ASAP triggers the **DB_admin** procedure, ASAP also performs other optimization, such as recompiling stored procedures for optimal database access, and so forth.

In the following example, after the function has deleted each set of 1000 orders, the function performs a commit and the rollback segment is flushed. This prevents the Oracle rollback segment from being exceeded. This example also employs the **SSP_orphan_purge** function described for the previous example.

```
CREATE OR REPLACE FUNCTION SSP_db_admin(
days INTEGER )
RETURN INTEGER
AS
StoO_selcnt INTEGER;
StoO_error INTEGER;
StoO_rowcnt INTEGER;
StoO_errmsg VARCHAR2(255);
cutoff_dts DATE;
retval integer;
BEGIN
IF (SSP_db_admin.days IS NOT NULL AND SSP_db_admin.days > 0) THEN
BEGIN
SSP_db_admin.cutoff_dts :=  SYSDATE-SSP_db_admin.days;
BEGIN
StoO_error   := 0;
StoO_rowcnt  := 0;
-- Created a loop to split the deletion of orders in portions of 1000
-- to remove the risk of reaching the rollback segment limit
-- NOTE! The orphans are deleted by a trigger defined on the
-- tbl_wrk_ord table and not by the SSP_orphan_purge function call
-- below
LOOP
```

```
  DELETE  FROM tbl_wrk_ord
WHERE comp_dts < SSP_db_admin.cutoff_dts
AND wo_stat = 104
AND rownum <= 1000;
 EXIT WHEN SQL%ROWCOUNT = 0;
  COMMIT;
END LOOP;
StoO_rowcnt := SQL%ROWCOUNT;
COMMIT WORK;
-- EXCEPTION
-- WHEN OTHERS THEN
-- StoO_error  := SQLCODE;
END;

END;
END IF;
BEGIN
retval := SSP_orphan_purge;
EXCEPTION
WHEN OTHERS THEN
StoO_error := SQLCODE;
StoO_errmsg := SQLERRM;
END;
RETURN 0;
END;
/
```

Sample Cron Script for Clearing Alarms, Events, and Process Information

The following is a sample cron script that clears alarm entries, event logs and process information.

```
#############################################################################
#
# ASAP database and log housekeeping script
#
# Call this script with one parameter, specifying the number of days.
# Alarm entries, event logs and process info
# logs are cleared. Log files and directories are also cleared.
#
# The script will exit if the number of days is less than 5
#
#############################################################################

. $HOME/.profile > /dev/null

PROG_NAME=`basename "$0"`

if [ "$1" == "" ]
then
    echo "Usage $PROG_NAME <admin_days>"
    exit
fi

let ADMIN_DAYS=$1+0
if [ "$?" != "0" ]
then
    echo "Number of days must be numeric"
    exit
fi

if [ $ADMIN_DAYS -lt 5 ]
```

```
then
    echo "Cannot run with less than 5 days lead-time"
    exit
fi


################################################################################
# Clear CTRL entries
################################################################################
echo "Deleting alarm logs, event logs and process info from CTRL more than $ADMIN_DAYS
days old"

sqlplus -s $CTRL_USER/$CTRL_USER << HERE

var admin_days number;
exec :admin_days := $ADMIN_DAYS;

delete from TBL_ALARM_LOG where start_dts < sysdate - :admin_days;
commit;
delete from TBL_EVENT_LOG where event_dts < sysdate - :admin_days;
commit;
delete from TBL_PROCESS_INFO where info_dts < sysdate - :admin_days;
commit;

HERE

################################################################################
# Truncate ASAP.Console and ControlProgramOutput so that old entries can be
# cleared with the diagnostic files
################################################################################
cd $LOGDIR
FILE1=ASAP.Console
FILE2=ControlProgramOutput
CUR_DTS=`date +'%Y_%b_%d-%T'`

if [ -f $FILE1 ]; then
    NEWFILE=$FILE1.diag.$CUR_DTS
    echo Copying current $FILE1 to $NEWFILE
    cp $FILE1 $NEWFILE
    cp /dev/null $FILE1
else
    echo $LOGDIR/$FILE1 not found
fi

if [ -f $FILE2 ]; then
    NEWFILE=$FILE2.diag.$CUR_DTS
    echo Copying current $FILE2 to $NEWFILE
    cp $FILE2 $NEWFILE
    cp /dev/null $FILE2
else
    echo $LOGDIR/$FILE2 not found
fi

################################################################################
# Clear log files
################################################################################
echo "Clearing old ASAP log files..."

find . -type f -name '*diag*' -atime +$ADMIN_DAYS -exec echo Removing file {} \; -exec
rm -f {} \;

################################################################################
```

```
# Clear log directories not accessed within admin days
################################################################################
echo "Clearing old ASAP log directories..."

find . -type d -name '2*' -atime +$ADMIN_DAYS -exec echo Removing directory {} \; -exec
rm -rf {} \;

cd -

echo "$PROG_NAME finished"
```

Purging Test Systems

The cleandata script is designed to clean data from the test system during system testing. It is intended for use in development environments.

> **Note**
>
> You must stop ASAP before running clean data script and WebLogic server needs to be restarted after running clean data script. Otherwise, a JDBCStoreException is thrown, and a WebLogic Server error is logged.

Usage

```
cleandata [-d]
```

The **-d** option instructs the script to retrieve password information from the credential store factory (CSF) wallet, located in *ASAP_Home*/**install/cwallet.sso** file, where *ASAP_Home* is the directory in which ASAP is installed. The **cwallet.sso** file is typically used only in development environments.

This script does the following:

- Deletes events, process information, and alarms from the Control (CTRL) database (specifically, truncates **tbl_event_log**, **tbl_process_info**, **tbl_alarm_log**, **tbl_unid**)
- deletes WOs from the ASAP database and runs an orphan purge (specifically, truncates **tbl_wrk_ord**, truncates **tbl_ne_monitor**, **tbl_ne_event**, **tbl_unid**, **tbl_label_value**, **tbl_srt_correlation**, **tbl_srt_ctx**, **temp_wrk_ord**)

  Note: The **tbl_ne_monitor** table has been deprecated from ASAP 4.6.x onwards.
- Deletes WOs from the SRP database (**tbl_wrk_ord**)
- Deletes performance data from Admin database (truncates **tbl_perf_order**, **tbl_perf_csdl**, **tbl_perf_asdl**, **tbl_perf_ne**, **tbl_perf_ne_asdl**, **tbl_aims_rpc**, **tbl_aims_rpc_param**, **tbl_aims_rpc_dest**, **tbl_aims_audit_log**)
- Deletes Java Message Service (JMS) messages from the Admin database
- Deletes diagnostic and log files

> **Note**
>
> Restart the WebLogic Server for the domain after running the cleandata script.

Purging the SARM Database

To purge the SARM database, you can also use the **purgeOrders** script. This script calls a multithreaded Java application which makes purging faster. This application purges the SARM database only.

Usage

You run the purge application by running the **purgeOrders** script. This script is located in the $ASAP_BASE/scripts directory. You must source the ASAP **Environment_Profile** before running this script.

The command-line syntax of the purgeOrders script is:

```
purgeOrders [-t <number of threads>] [-l <diag_level>] [-f] [-h]
```

where:

-t <*number of threads*> (optional) is the number of threads. If specified, it overrides the value of DB_PURGE_THREADS in ASAP.cfg file.

-l <*diag_level*> (optional) specifies the diagnostic level (KERN, LOW, SANE, or PROG). The default is SANE.

-f forces the script to run in non-interactive mode. Without this option, the script displays an interactive prompt to confirm running the purge application.

-h prints the usage information.

Configuration Parameters

The purge application reads the following configuration parameters from the SARM section of the ASAP.cfg file.

- **DB_PURGE_DAYS** : Number of days to keep the updated orders. The updated orders older than this number of days are purged. Orders are not purged if the value is 0.

  Valid Range: >=0

  Default = 100

- **DB_PURGE_THREADS**: Number of threads to use. The purge application spawns this number of threads and runs the purge stored procedure in each thread. This allows you to run the purge procedure at different times of the day using different levels of parallelism.

  Valid Range: 1 to 50

  Default: 10

  > **Note**
  >
  > The -t parameter in the command line overrides this value specified in ASAP.cfg.

- **DB_PURGE_COMMIT_ROWS**: Number of rows to be deleted before a COMMIT is issued.

  Valid Range: >=1

  Default: 1000

- **DB_PURGE_ORPHANS_ENABLED**: Enable or disable the **SSO_orphan_purge** procedure which removes orphaned data definitions in the SARM database.

  Valid Range: 0 or 1

  Default: 0

- **DB_PURGE_MAX_TIME**: Maximum purge time. The maximum amount of time in minutes that the purge operation is allowed to run. If this parameter is set to **0**, the purge operation is run without a time limit.

  Valid Range: >=0

  Default: 0

- **DB_PURGE_GET_RANGE_PROC**: The stored procedure that the purge application should call to get the SRQ_ID range (MIN, MAX) of the orders to be purged.

  The signature of the stored procedure is as follows:

  ```
  FUNCTION SSP_get_srqid_purge_range(
     days IN INTEGER,
     min_srqid OUT INTEGER,
     max_srqid OUT INTEGER)
  RETURN INTEGER
  ```

  Valid Range: N/A

  Default: SSP_get_srqid_purge_range

- **DB_PURGE_RANGE_PROC**: The stored procedure that each thread should call to purge the work orders in a SRQ_ID range.

  The signature of the procedure is as follows:

  ```
  FUNCTION SSP_purge_wo_in_srqid_range(
     days IN INTEGER,
     commit_rows IN INTEGER,
     start_id IN INTEGER,
     end_id IN INTEGER,
     max_time IN INTEGER,
     rows_deleted OUT INTEGER)
  RETURN INTEGER.
  ```

  Valid Range: N/A

  Default: SSP_purge_wo_in_srqid_range

- **DB_PURGE_LOG_INTERVAL**: The interval, in minutes, to log messages about the progress of the purge operation.

  Valid Range: >=1

  Default: 5

Logging and Diagnostics

The purge application logs diagnostic messages to the $ASAP_BASE/DATA/logs/<date>/SARM<ENV_ID>_PURGE.diag file.

The purge application logs progress messages to the diagnostic file while the purge threads are running, and after all threads are finished. The diagnostic messages show the elapsed time, number of orders deleted, and the purging rate.

Scheduling Purge Jobs

Since the purge application is a Java application that runs outside of the SARM server, you need to schedule the new purge application to be run outside of SARM. This is done using the Linux cron utility. When you run the purge application from cron, you should use the -f command line option so that the **purgeOrders** script runs in non-interactive mode.

Purge Conflict Resolution

If there is another instance of the purge application running against the same SARM database, the purgeOrders script prints the following error message and stops:

```
Error: There is another AsapParallelPurge job running for SARM<ENV_ID>. Exiting ...
```

If the old purging mechanism is enabled, by setting DB_ADMIN_ON to 1 in the SARM section of ASAP.cfg, the new purge application logs the following warning message to the diagnostic file and continues to run.

```
WARNING: The SARM database administration thread is enabled (DB_ADMIN_ON=1 in ASAP.cfg).
This may result in multiple purge jobs being run concurrently.
```

Customization

You can provide your own stored procedures for determining the SRQ ID range of the work orders and purging the work orders. You do this by specifying the names of your stored procedures in the DB_PURGE_GET_RANGE_PROC and DB_PURGE_RANGE_PROC parameters in the ASAP.cfg.

Using the Purge Application

To use the purge application:

1. Disable the existing SARM database purge operation by setting **DB_ADMIN_ON** to **0** in the SARM section of the **ASAP.cfg** file.
2. In the SARM section of the **ASAP.cfg** file, review the default settings of the DB_PURGE* parameters and change them if necessary. For example:
   - Change **DB_PURGE_DAYS** to comply with your company's data-retention policy.
   - Change **DB_PURGE_THREADS** to increase or decrease the level of parallelism.
   - Change **DB_PURGE_MAX_TIME** if you want to specify a time limit for purging.
   - Change **DB_PURGE_GET_RANGE_PROC** and **DB_PURGE_RANGE_PROC** if you need to use your own stored procedures.
3. Schedule the **purgeOrders** script to be run regularly using an external scheduling tool.

6

Troubleshooting ASAP

This chapter provides troubleshooting information for Oracle Communications ASAP.

Overview of Troubleshooting ASAP

As an ASAP system administrator, you can perform the following tasks:

Troubleshooting Checklist

When any problems occur, it is best to do some troubleshooting before you contact Oracle Global Support:

- You know your installation better than Oracle Global Support does. You know if anything in the system has been changed, so you are more likely to know where to look first.
- Troubleshooting skills are important. Relying on Global Support to research and solve all of your problems prevents you from being in full control of your system.

If you have a problem with your Product system, ask yourself these questions first, because Oracle Global Support will ask them of you:

- What exactly is the problem? Can you isolate it? For example, if it is a problem with an application, does it occur on one instance of the application, or all instances?
- Oracle Global Support needs a clear and concise description of the problem, including when it began to occur.
- What do the log files say?
- This is the first thing that Oracle Global Support asks for. Check the error log for the Product component you're having problems with.
- Have you read the documentation?
- Look through the list of common problems and their solutions in Diagnosing some common problems with Product.
- Has anything changed in the system? Did you install any new hardware or new software? Did the network change in any way?
- Have you read the Release Notes?
- The Release Notes include information about known problems and runarounds.
- Does the problem resemble another one you had previously?
- Has your system usage recently jumped significantly?
- Is the system otherwise operating normally?
- Has response time or the level of system resources changed?
- Are users complaining about additional or different problems?
- Can you run clients successfully?
- Are any other processes on the system hardware functioning normally?

If you still can't resolve the problem, contact Oracle Global Support as described in "Getting Help with ASAP Problems."

Using Error Logs to Troubleshoot ASAP

ASAP error log files provide detailed information about system problems. If you're having a problem with ASAP, look in the log files.

Log files include errors that need to be managed, as well as errors that do not need immediate attention (for example, invalid login attempts). To manage log files, you should make a list of the important errors for your system, as opposed to errors that do not need immediate attention.

Understanding Error-Message Syntax

For more information about error message syntax, see "About Log Files."

Collecting Diagnostic Information

For more information about collecting diagnostic information, see "About Diagnostic Files."

Common ASAP Problems and Solutions

This section describes the following problems, and how to resolve them:

- #unique_128
- Problem: ASAP Servers Do Not Start - Wrong ASAP User Owner and Permissions
- Problem: ASAP Servers Processes Do Not Start - Database Server Processes Used
- Problem: ASAP Servers Processes Do Not Start - Database Server Sessions Used
- Problem: ASAP Servers Do Not Start - Insufficient Server User Connections Defined
- Problem: ASAP Servers Do Not Start - Insufficient Number of Threads
- Problem: Control Server Crashes - No Free Messages
- Problem: JNEP Server Does Not Start - Wrong Database Connection Information
- Problem: JNEP Server Does Not Start - Invalid Server Port Numbers
- Problem: NEP Server Does Not Start - Problem with JNEP Java Process Start Script
- Problem: WebLogic Server Fails to Detect Passive RAC Database During Failover
- Problem: Exception During Connection with Network Element Processor

Hold

During a Hold operation, a work order is given the Held status, and the SARM retains the work order and activates it only after receiving a release request or an order update. The following qualifications apply when the states listed below are updated with a Hold order:

- **Non-Existent orders** – The SARM accepts the work order.
- **Initial, Held, Reviewed, or Translation Error states** – The SARM accepts the work order and overwrites the existing copy of the work order with the order update.

- **In Progress, Cancelled, or Completed states** – The SARM rejects the Hold order.
- **Failed states** – The SARM cancels all CSDLs on the work order and updates the existing copy of the order using a newly generated work order with the same work order ID and a different service request ID.

Problem: ASAP Servers Do Not Start - Wrong ASAP User Owner and Permissions

Owner and file permissions might have been changed preventing the ASAP servers record data to the file system. For example, file or folder owner changes or file permission changes for DATA and its subfolders may cause this issue.

The following error message is generated:

```
/oracle/user1/A720_I/scripts/start_control_sys: line 197: /oracle/user1/A720_I/DATA/logs/
ASAP.Console: cannot create [Permission denied]
```

Make sure that all the folders and files under ASAP installation folder are owned by the ASAP user and the user has all the required file permissions. For example, it should have read/write for everything under DATA folder otherwise it cannot create the server diagnostic files.

Problem: ASAP Servers Processes Do Not Start - Database Server Processes Used

The following error message will be seen in the control or other server diagnostic files:

```
ORA-00020: maximum number of processes (%s) exceeded
```

Increase the value of the PROCESSES initialization parameter.

Problem: ASAP Servers Processes Do Not Start - Database Server Sessions Used

The following error message will be seen in the control or other server diagnostic files:

```
Error - ORA-00018: maximum number of sessions exceeded
```

Increase the value of the SESSIONS initialization parameter.

Problem: ASAP Servers Do Not Start - Insufficient Server User Connections Defined

Some ASAP servers do not start and a message like the following is observed in the Control server diagnostic file:

```
"Configuration of 70 connections has been exceeded, connection rejected."
```

Each ASAP server started establishes a defined number of connections to the Control server. There is an open server configuration parameter named **MAX_CONNECTIONS** which can be used by any open server application to limit the maximum number of connections from client applications. This problem is usually encountered when additional Network Element Processor (NEP) servers are configured as each additional NEP server means more connections to the Control server.

The solution is to increase the value of the parameter **MAX_CONNECTIONS** for the Control server in the *ASAP_home***/config/ASAP.cfg** file. You can also set a value in the global section for all ASAP servers. Note that when adjusting this parameter, you may need to adjust additional parameters (see "Improving ASAP Performance ").

Problem: ASAP Servers Do Not Start - Insufficient Number of Threads

ASAP servers cannot start with the reason being indicated as some remote procedure calls (RPCs) are not defined. For example:

```
Error : RPC install_exit_handler Not Defined
Error: Unable to Spawn Service Thread TORONTO - Insufficient Resources
SARMH730 Server: Information: Error 16115 Severity 10 State 0
'Could not start thread
Error: Unable to Spawn Service Thread WO Mgr 2 - Insufficient Resources
```

ASAP Servers are configured to have a maximum number of threads in the *ASAP_home***/config/ASAP.cfg** file. If this number is not big enough, when a thread is tried to be started, the server process will generate an error message and terminate.

Adjust the value of the parameter **MAX_THREADS** in the ASAP.cfg, globally or for a specific server.

Note that when adjusting this parameter, you may need to adjust additional parameters (see "Improving ASAP Performance ").

Problem: Control Server Crashes - No Free Messages

Control server crashes with the following messages in the Control server diagnostic file:

```
153957.532:48:PROG:Fatal Thread Error :1429:main.c
CTRLPRD1 Server: Fatal Thread Error:
Open Server Error # [16016] Severity [15] State [0] Error Text [No free messages.]
>> 153957.928:48:PROG:Fatal Thread Error :1436:main.c
Error: [CTRLPRD1] Fatal Thread Error - Terminating
```

Message resources were used up during high volume WO provisioning because **MAX_MSGPOOL** is not big enough for Control server.

Increase the value of **MAX_MSGPOOL** for the Control server in the *ASAP_home***/config/ASAP.cfg** file. This should be done in the CTRL section as otherwise it will affect all the servers.

The minimum value of this configuration value is **MAX_MSGQUEUES***256. You may need to adjust additional parameters (see "Improving ASAP Performance ")

Problem: JNEP Server Does Not Start - Wrong Database Connection Information

The database connection information (**DB_CONNECT**) in **ASAP.properties** was wrong and as a result JNEP was unable to connect to database while starting.

While NEP server java process (JNEP) was creating a pool of database connections, it failed with the exception message :

```
"The Network Adapter could not establish the connection"
java.sql.SQLException: Io exception: The Network Adapter could not establish the
connection
```

```
at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:125)
at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:162)
at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:274)
at oracle.jdbc.driver.T4CConnection.logon(T4CConnection.java:319)
at oracle.jdbc.driver.PhysicalConnection.(PhysicalConnection.java:344)
at oracle.jdbc.driver.T4CConnection.(T4CConnection.java:148)
at oracle.jdbc.driver.T4CDriverExtension.getConnection(T4CDriverExtension.java:32)
at oracle.jdbc.driver.OracleDriver.connect(OracleDriver.java:545)
at oracle.jdbc.pool.OracleDataSource.get
>>
```

Database connection information in ASAP.properties file is wrong. Correct it as below and restart the NEP server.

```
DB_CONNECT=(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = xx.xx.xx.xx)
(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = orcl)))
```

HOST, PORT or SERVICE_NAME could be wrong in this property.

Problem: JNEP Server Does Not Start - Invalid Server Port Numbers

NEP server does not start with an error message in the Control server diagnostic file. A port number outside the range might have been assigned to the NEP server in **ASAP.properties** file. If the port number is within the range, it might have been bound by another process. Or the issue could be with the listener port that is assigned to the NEP java process. It could be out of range or already bound by another process.

Correct the port number for the NEP server in **ASAP.properties** file file to be inside the range. If the port number is bound by another process already, it cannot be used by the NEP server. A free, unused port number should be assigned for the NEP server.

Make sure that the port number assigned to the NEP java process (table tbl_listeners in Control server database) is also within the range and not bound by any other process.

Problem: NEP Server Does Not Start - Problem with JNEP Java Process Start Script

The NEP server java process is started by a script named $NEP_jinterpreter which is under $ASAP_BASE/programs folder. This script may not have appropriate permissions.

Give the proper execution permission for the script.

Problem: WebLogic Server Fails to Detect Passive RAC Database During Failover

In a development environment, when testing Oracle RAC failover capabilities, it is possible to experience the following WebLogic error:

```
weblogic.jms.common.JMSException: weblogic.messaging.kernel.KernelException:
Error persisting message
```

The starting point for this scenario involves the following configuration:

- A running ASAP server environment and active WebLogic Server instance (ASAP Environment).
- A running active Oracle RAC database (RAC1).

- A passive Oracle RAC database that is shutdown (RAC2).

To achieve the error message listed above, you must complete the following steps:

1. Manually startup RAC2.
2. Manually shutdown RAC1 within four minutes of starting RAC2.

   The ASAP Weblogic Server instance requires up to four minute to detect RAC2. Shutting down RAC1 within this time period will prevent ASAP from failing over to RAC2 because the ASAP WebLogic Server has not had enough time to detect the presence of RAC2.

This error message is possible in a specific scenario that is unlikely to happen in a production environment.

Problem: Exception During Connection with Network Element Processor

When upgrading from 7.2 to 7.4, an exception occurred while connecting to one of the Network Cartridge:

```
Exception occurred: javax.net.ssl.SSLHandshakeException: Received fatal
alert: protocol_version
```

To resolve the issue:

- Change in `java.security` in the **ASAP_BASE/JRE/jre/lib/security** directory:

  ```
  jdk.tls.disabledAlgorithms=SSLv3, TLSv1, RC4, DES, MD5withRSA, \
  DH keySize < 1024, EC keySize < 224, 3DES_EDE_CBC, anon, NULL, \
  include jdk.disabled.namedCurves
  ```

- Restart ASAP.

Order Security

As part of the SRP-to-SARM protocol, the SARM performs a security check on all work orders transmitted for provisioning. The SRP passes a user ID and password to the SARM using the protocol. The SARM checks these values with a list of valid users in a static user-populated database table. If the user ID and password combination is not valid, the SARM rejects the work order. If the combination is valid, the SARM accepts the work order.

The order security logic is centralized in the SARM to avoid having each SRP conduct its own security check. As new SRPs are added to the system, centralization becomes a critical advantage.

WebLogic Administration and Managed Server Failure and Recovery Scenarios

If the administration server shuts down while the managed server is running, the administration server regains control of the domain upon restart without having to restart the managed server.

If the administration server cannot be restarted following a failure, you must restore the administration server on another machine.

Restarting the WebLogic Managed Server

If the administration server is running, the managed server retrieves configuration data from the administration server. If the administration server is unavailable, the managed server retrieves its configuration and security information from the filesystem.

SARM Failure Scenario

If the SARM fails, the SRP can neither send orders to the SARM nor receive notifications from the SARM, although the SRP is running and receiving orders from the upstream. Therefore, the SRP cannot process orders and remains idle until the SARM is restarted.

NEPs complete the processing of Atomic Service Description Layer (ASDL) requests currently in progress, and then it becomes idle.

The SRP and NEPs send periodic “heartbeat" messages to determine when the failed SARM becomes available.

NEP Failure Scenario

In the event of NEP failure, the SRP continues to translate and send orders to the SARM, which continues to provision ASDLs scheduled to be provisioned by any operational NEP. All ASDLs to be provisioned by NEs managed by the failed NEP are added to SARM Provisioning Pending Queue. The SARM sends periodic “heartbeat" messages to determine when the failed NEP becomes available.

Control Server Failure Scenario

If the Control server is shut down, all other ASAP applications controlled by it are also shut down.

Database Failure Scenario

If the SRP database fails, the SRP is shut down if it relies on the SRP database. Similarly, the SARM and NEP servers shut down whenever their respective databases fail. If the Control database fails, the Control server is shut down, taking down all other ASAP applications controlled by it. In a distributed environment, if the primary Control server goes down, the secondary Control servers detect this and shut down.

If the SQL Server is down, all databases fail and the entire system is shut down.

NE Unavailability Scenario

If an NE becomes unavailable, all ASDLs to the unavailable NE queue up in the SARM. The SARM will not take new work orders (WOs).

If the asdl_timeout parameter is set for an ASDL, and the timeout parameter exceeds asdl_retry_number parameter, the WO to which the ASDL belongs fails and rolls back.

If the request_timeout parameter (an NE timeout parameter) is set for the WO to which an ASDL belongs, and the request_retry_number (an NE timeout parameter) exceeds, the WO fails and rolls back.

If neither ASDL nor NE timeout parameters are set, and the WO timeout parameters exceed, all orders with ASDLs going to the unavailable NE fail due to timeout.

SRP and SARM Failure Scenario

If both the SRP and SARM fail, each NEP completes its processing of ASDL requests currently in progress. The NEP then closes all connections to NEs and then remains idle.

SARM and NEP Failure Scenario

If the SARM and NEP fail, the SRP cannot send orders or receive notifications from the SARM. No order provisioning is possible.

Getting Help with ASAP Problems

If you can't resolve your ASAP problem, contact Oracle Global Support.

Before You Contact Global Support

Problems can often be fixed by shutting down ASAP and restarting the computer that the ASAP system runs on. To shut down and restart ASAP, see "Starting and Stopping ASAP."

If that doesn't solve the problem, the first troubleshooting step is to look at the error log for the application or process that reported the problem. See "Using Error Logs to Troubleshoot ASAP." Be sure to observe the checklist for resolving problems with ASAP before reporting the problem to Oracle Global Support. See "Troubleshooting Checklist."

Reporting Problems

If the checklist for resolving problems with ASAP does not help you to resolve the problem, write down the pertinent information:

- A clear and concise description of the problem, including when it began to occur.
- Relevant portions of the relevant log files.
- Relevant configuration files.
- Recent changes in your system, even if you don't think they are relevant.
- List of all ASAP components and patches installed on your system.

When you are ready, report the problem to Oracle Global Support.

A

ASAP Directory Structure

This appendix describes the Oracle Communications ASAP directory structure.

ASAP Directory Structure and Contents

Table A-1 lists and describes the ASAP directories.

Table A-1 ASAP Directories

Directory	Description
ASAP_home/**activationModels**	Contains installed SAR files. All additional SAR files that you upload using Service Activation Deployment Tool (SADT) or the **installCartridge.sh** script are added to this folder.
ASAP_home/**config**	This directory contains important configuration files such as **ASAP.cfg** and **OCA.cfg**.
ASAP_home/**DATA**	Contains the **logs** directory.
ASAP_home/**DATA/logs**	Location of all server logs and the ASAP.console log files. The ASAP.Console file contains standard input, standard output, and errors that are sent to a console screen. The ASAP.Console file records any startup errors.
ASAP_home/**DATA/logs/***yyyymmdd*	These directories are created by ASAP daily or whenever a server starts up. Diagnostic files are located here and contain *appl_name*.**diag** files (where *yyyymmdd* is the date for which you want to view the diagnostics and *appl_name* is the client or server application.)
ASAP_home/**isql**	Contains sql scripts for ASAP database upgrades. The sql scripts with the words procs contain stored procedures, functions, and triggers and should be used when upgrading ASAP. The ASAP installer uses the other scripts when you first install ASAP and should not be run manually.
ASAP_home/**db_migration**	Contains database sql and korn shell scripts for upgrading the ASAP database.
ASAP_home/**install**	Contains files related to the ASAP installation.
ASAP_home/**JLIB**	Contains java archive (jar) files used by the ASAP Java components.
ASAP_home/**JRE**	Contains the Java runtime environment and all related sub folders.
ASAP_home/**lib**	Contains jar file and Enterprise Archive (.ear) files for WebLogic based ASAP tools and servers.
ASAP_home/**patch**	This directory contains various patch files installed automatically when you install or upgrade ASAP.
ASAP_home/**programs**	This directory contains various programs used to run ASAP utilities, scripts, and servers.
ASAP_home/**samples**	This directory contains various sample servers, service models, sar files, and server configurations.
ASAP_home/**samples/ASDL_ROUTE**	This sample directory contains the files and instructions for running sample user defined stored procedures to route WOs.
ASAP_home/**samples/CsolSrp**	This directory contains a sample C-based SRP.

Table A-1 (Cont.) ASAP Directories

Directory	Description
ASAP_home/samples/CsolSrp/csolsrp_sample	This directory contains the files and instructions for implementing the most basic functions of an SRP.
ASAP_home/samples/DIT	This directory contains a sample cartridge, and related sub folders, configured to run with the Java Service Request Processor (JSRP) sample client and SRT. This sample is provided to test the functionality of ASAP, the JSRP client, and the optional SRT component.
ASAP_home/samples/EDD	This directory contains two sample programs and related sub folders that demonstrate how to build ASAP external device driver (EDD) applications. Note the EDD functionality has been deprecated.
ASAP_home/samples/JeNEP	This directory contains Java Network Element Processor (NEP) samples and instructions that demonstrate support for Lightweight Directory Access Protocol (LDAP), Socket, and Telnet connection protocols to NEs.
ASAP_home/samples/JeNEP/async_ne	This directory contains a sample that illustrates how asynchronous messaging to and from an NE can be configured with ASAP.
ASAP_home/samples/JeNEP/jenep_demo	This directory contains the sample CORBA, LDAP, Socket, and Telnet Java-enabled NEP (JNEP) directories.
ASAP_home/samples/JeNEP/jenep_demo/LDAP	This directory contains the files required to configure the LDAP protocol for a JNEP.
ASAP_home/samples/JeNEP/jenep_demo/Socket	This directory contains the files required to configure a Socket based protocol for a JNEP.
ASAP_home/samples/JeNEP/jenep_demo/Telnet	This directory contains the files required to configure the Telnet protocol for a JNEP.
ASAP_home/samples/jsrp	This directory contains instructions for configuring and using the sample JSRP, including the JSRP Java Value Types (JVT) and Java Message Service (JMS) interfaces and sample JVT and JMS clients. When you compile the clients, and additional folder appears that contains the java classes for the clients.
ASAP_home/samples/jsrp/src	This directory contains the source files for the JSRP JVT and JMS clients.
ASAP_home/samples/jsrp/txt	This directory contains sample JVT WOs that you can use with the JVT client.
ASAP_home/samples/jsrp/xml	This directory contains sample JMS WOs that you can use with the JMS client.
ASAP_home/samples/nep	This directory contains sample files and instructions for creating a custom NEP.
ASAP_home/samples/sadt	This directory contains sample cartridges, sar files, and sample configuration files.
ASAP_home/samples/sadt/3GWireless	This directory contains a sample cartridge and subdirectories.
ASAP_home/samples/sadt/ATM_FR	This directory contains a sample cartridge and subdirectories.
ASAP_home/samples/sadt/DemoInstall	This directory contains a sample cartridge and subdirectories.
ASAP_home/samples/sadt/IPVPN	This directory contains a sample cartridge and subdirectories.
ASAP_home/samples/sadt/ NetworkManager	This directory contains a sample cartridge and subdirectories.

Table A-1 (Cont.) ASAP Directories

Directory	Description
ASAP_home/**samples/sadt/SampleCommonConfig**	This directory contains sample server configuration xml files. You can use these sample files to modify, delete, or add ASAP servers with Service Activation Configuration Tool (SACT).
ASAP_home/**samples/sadt/sar**	This directory contains the SAR files for the sample cartridges.
ASAP_home/**scripts**	This directory contains various scripts. Some scripts are available for ASAP users (for example, asap_utils, start_asap_sys, and so on), and some scripts are only used by the ASAP installer.
ASAP_home/**xml**	This directory contains XML based files.
ASAP_home/**xml/xsd**	This directory contains the XML schemas that ASAP uses to validate data structures and formats for ASAP server components, SAR files, service models, and so on.
ASAP_home/**xml/xslt**	This directory contains xslt style sheets used by various scripts to transform xml documents into new xml documents. The export_tool.sh script located in the *ASAP_home*/**scripts** directory uses these files to convert database data into xml content that can be used by SACT.

B

Sticky and Non-Sticky Requests Supported by Order Balancer

This appendix provides the list of sticky and non-sticky requests supported by Order Balancer JMS interface and Web Services.

The following is a list of Non-Sticky requests supported by JMS interface and Web Services:

- createorderbyvaluerequest
- getordertypesrequest
- getquerytypesrequest
- getservicetypesrequest
- getsupportedoptionaloperationsrequest
- trycreateordersbyvaluesrequest

Table B-1 lists the Sticky requests supported by JMS interface and Web Services.

Table B-1 Supported JMS Interface and Web Services Operations

JMS Interface	Web Services
abortorderbykeyrequest	abortOrderByKeyrequest
abortservicerequest	cancelOrderByKeyrequest
addextendedorderpropertyrequest	getOrderByKeyrequest
addorderparameterrequest	lockOrderrequest
addserviceparameterrequest	removeOrderByKeyrequest
addservicerequest	resumeOrderByKeyrequest
cancelorderbykeyrequest	setOrderByValuerequest
deleteservicerequest	startOrderByKeyrequest
getinitorderbykeyrequest	stopOrderByKeyrequest
getorderbykeyrequest	suspendOrderByKeyrequest
lockorderrequest	unlockOrderrequest
removeextendedorderpropertyrequest	
removeorderbykeyrequest	
removeorderparameterrequest	
removeserviceparameterrequest	
resubmitorderbykeyrequest	
resumeorderbykeyrequest	
retryservicerequest	
setextendedorderpropertyrequest	
setorderparameterrequest	
setserviceparameterrequest	
startorderbykeyrequest	
stoporderbykeyrequest	
suspendorderbykeyrequest	
unlockorderrequest	
validateorderoperationrequest	
validateserviceoperationrequest	

Table B-2 is a list of operations that are not supported by Order Balancer Web Services.

Table B-2 Unsupported Web Services Operations

Unsupported Operations by Web Services
tryRemoveOrdersByKeys
tryStartOrdersByKeys
tryAbortOrdersByKeys
queryOrders
getOrdersByKeys
queryManagedEntities