Oracle® Communications ASAP

Concepts

Release 7.2

E18880-01

April 2012

ORACLE®

Oracle Communications ASAP Concepts, Release 7.2

E18880-01

Contents

3 Service Request Processor and Java Service Request Processor

4 The Service Activation Request Manager

5 The Network Element Processor

Preface

This guide provides an overview of Oracle Communications ASAP, explains its functional architecture, and describes the working of various ASAP components.

Audience

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

Related Documents

For more information, see the following documents in the Oracle Communications ASAP documentation set:

- *Oracle Communications ASAP Release Notes*
- *Oracle Communications ASAP Installation Guide*
- *Oracle Communications ASAP System Administrator's Guide*
- *Oracle Communications ASAP Service Request Translator User's Guide*
- *Oracle Communications ASAP Order Control Application User's Guide*
- *Oracle Communications ASAP Cartridge Developer's Guide*
- *Oracle Communications ASAP Server Configuration Guide*
- *Oracle Communications ASAP Security Guide*
- *Oracle Communications ASAP Developer's Guide*

Note: To download the *Oracle Communications ASAP Developer's Guide* from the Oracle software delivery Web site, select **Oracle Communications Service Activation Developer Documentation Pack**. You can visit the Oracle software delivery Web site at:

`http://edelivery.oracle.com`

1

Introduction

Oracle Communications ASAP equips telecommunications service providers with a single platform for automated service activation. ASAP receives service requests from any source and transmits the required service activation information to any destination network device.

ASAP's core architecture isolates business semantics (rules and behavior) from technology semantics (interface implementations and protocols). This architecture allows ASAP to handle multiple, heterogeneous network technologies and supports various interfaces, including:

- SOAP/XML
- HTTP/S
- TCP/IP
- Telnet
- SSH
- FTP/SFTP
- CORBA
- LDAP
- TL1
- SNMP
- OSS through Java

Because ASAP complies with accepted international standards, ASAP supports all convergent technology domains with a flexible and customizable repository of service definitions and business semantics. ASAP can activate services on network elements or element management systems.

ASAP offers the following characteristics:

- **Flow-through activation** – You can enable ASAP's automated flow-through capability directly at the source of the service request. Service providers can eliminate manual hand-offs and provisioning fallout. ASAP allows the configuration of flexible service definitions and business rules. It allows rapid queries of network inventory and access to other operations support systems (OSS) to collect information required for uninterrupted activation. ASAP can handle a range of error conditions to provide flow-through and resilience to service order fallout.
- **Convergent activation** – ASAP activates convergent service offerings with a combination of Wireline, Wireless, IP VPN, xDSL, and other services. ASAP

coordinates bundled service packages with full transaction control. ASAP supports:

- **Heterogeneous requests for service** – Service requests can originate from a variety of sources and customer interfaces, including interactive voice recognition, the Internet, kiosks, and customer service agents. For each service request source (including new customer interfaces), ASAP can use the same service definitions, ensuring consistency across all sources.
- **Heterogeneous network support** – ASAP concurrently supports multiple types of network elements and multiple software loads within each specific network element. This ensures support for evolving network topologies.

- **Transaction control** – ASAP transaction control ensures the coordination of requests that consist of multiple services. Transaction control ensures that the service request is activated in the correct sequence on all required network elements. ASAP provides an audit trail of all transactions related to each service request.

  Transaction control is equipped with a rollback option. In case of a failure, you can roll back the changes to restore the network to a known state.
- **Intelligent horizontal deployment** – ASAP can be deployed to support multiple services in multiple network domains. In a single ASAP system, service providers can expand the number of interfaces to service-originating systems and to network elements. Consequently, ASAP eliminates the need to implement multiple activation systems for different service and network domains.
- **Data-driven architecture** – ASAP enables rapid introduction of new services and technology. New services, business rules, and network elements can be introduced quickly. ASAP provides a set of APIs and interface development toolkits. ASAP also supports numerous transport and application-level protocols and standards. These capabilities simplify the development of OSS interfaces.
- **High availability –** ASAP has been deployed in high-availability environments, where service activation operates 24/7 with minimal downtime. Support for these HA environment are built upon intelligent ASAP processing logic that provides automatic process recycling and failover. These ASAP features, along with real application clustering (RAC) integration, minimize downtime.

ASAP Activation Processing Features

ASAP's activation processing features are:

- **Flexible error processing** – Various return states can be defined for specific network conditions (for example, soft-error processing, retry logic, or delayed errors). A combination of script-driven interfaces and soft-error processing logic allows ASAP to handle different network conditions automatically. This feature reduces fallout that is caused by inconsistent data.
- **Scheduling, order relationships, and prioritization** – Service requests can be scheduled immediately or for a future date or time. ASAP manages and resolves parent/child relationships to control dependency and activation sequencing. ASAP's built-in prioritization allows for efficient scheduling. Prioritization can expedite requests through ASAP when customer service agents are activating services in real time.
- **Network element routing and connectivity** – ASAP routes requests to destination network elements and executes the corresponding script on the destination network element through a connection that is automatically established, or

through a dynamic connection if required. Connectivity to network elements is based on user-defined communications devices and their parameters, including connection thresholds.

- **Audit trails, statistics, and reporting** – ASAP maintains detailed logs, audit trails, and statistics of all processing, both system and user-initiated. A reporting option allows the service provider to report against operational, statistical, and configuration data maintained in ASAP. Service providers can use the reports supplied or custom-developed reports.

ASAP Management Components

ASAP's flow-through activation and activation processing features are managed by the following components:

- **Activation fallout management** – Support for activation fallout management is provided through the Order Control Application (OCA) Client. The OCA Client can be deployed as a standalone application or within a Web browser as an applet.
- **ASAP operational configuration** – ASAP supports a command-line mode using XML configuration files to configure, update, and view the ASAP operational configuration. This configuration includes ASAP process distribution and network topology. The network topology specifies details of network element IDs, activation routing, communication devices and parameters, and thresholds of automated and transparent connections to network elements.
- **ASAP monitoring and control** – ASAP enables telecommunications operational centers to configure operational parameters, including ASAP events and alarms. These centers can monitor and manage ASAP operations including ASAP process run-time status, process startup and shutdown, communications device status, and connectivity to network elements, pending requests, and event/alarm acknowledgment.
- **Localization capability** – ASAP provides the capability to translate ASAP to one or more western languages represented by the 8-bit character set. Items that can be localized include:
  - Activation messages
  - Service command descriptions such as Common Service Description Layer (CSDLs), Atomic Service Description Layer (ASDLs), and so on
  - Operational parameter descriptions (events, alarms)
  - Graphical user interface screens and labels
  - Time, date, and number formats

  The Localization Toolkit is an optional ASAP component.

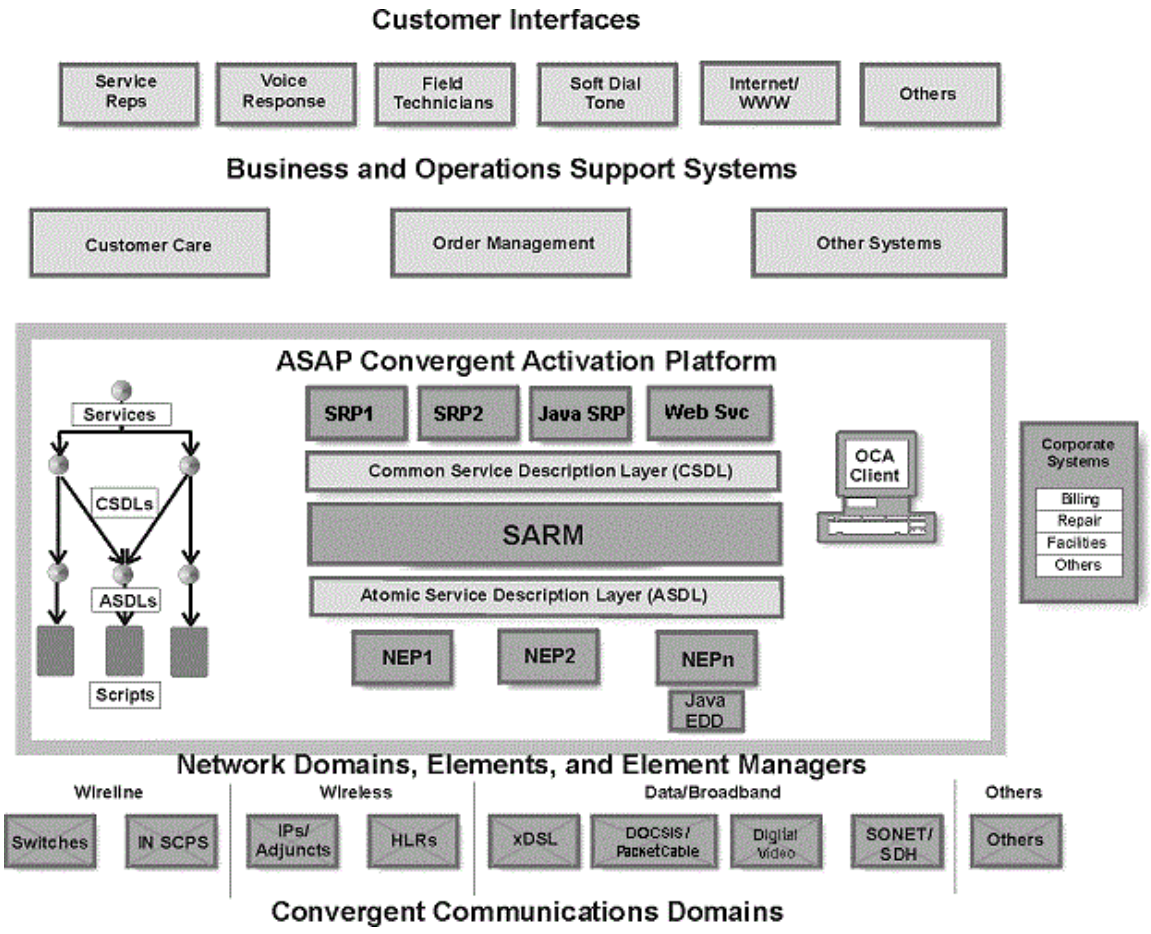
Note: The OCA SDK enables you to localize the OCA client.

2

ASAP Component Overview

Figure 2–1 illustrates the functional architecture of Oracle Communications ASAP.

Figure 2–1 ASAP Functional Architecture



The ASAP multitier architecture operates on an Oracle database engine.

To run ASAP, install the ASAP software and then implement the service order and network element interfaces to support specific services and network elements by using the following customizable modules:

- Service Request Processor (SRP) or Java SRP modules interface with the host system

- Network Element Processor (NEP) and Java NEP modules interface with network elements

> **Note:** These interfaces can be purchased from Oracle in the form of cartridges or they can be developed.

ASAP Architecture

ASAP consists of a set of multithreaded UNIX client/server processes and Java 2 Platform, Enterprise Edition (J2EE) applications that communicate with one another and with the appropriate database server. This arrangement has several advantages:

- **Flexibility** – Client/server applications are easily distributed across several, possibly heterogeneous platforms. The result is a scalable architecture that can expand to meet evolving requirements.
- **Manageability** – Because common services are handled on a server, application size and complexity are reduced. Client applications are simplified, duplicate code is minimized, and application maintenance is easier.
- **Customization** – Client/server architecture enables applications to be developed with distinct components. You can modify or replace these components without affecting other parts of the application.
- **Improved performance** – Multithreaded application architecture improves performance by enabling you to add additional processing functionality as threads within the process, as opposed to additional processes within the network. This capability enables greater throughput for the following reasons:
  - A threaded application makes better use of operating system resources than the conventional, multiprocess application.
  - Interthread communication is much faster than interprocess communication.
- **Improved communication between applications** – Client applications that use different communication protocols cannot communicate directly. A client/server architecture, however, allows communication using a "gateway" server that understands both protocols.

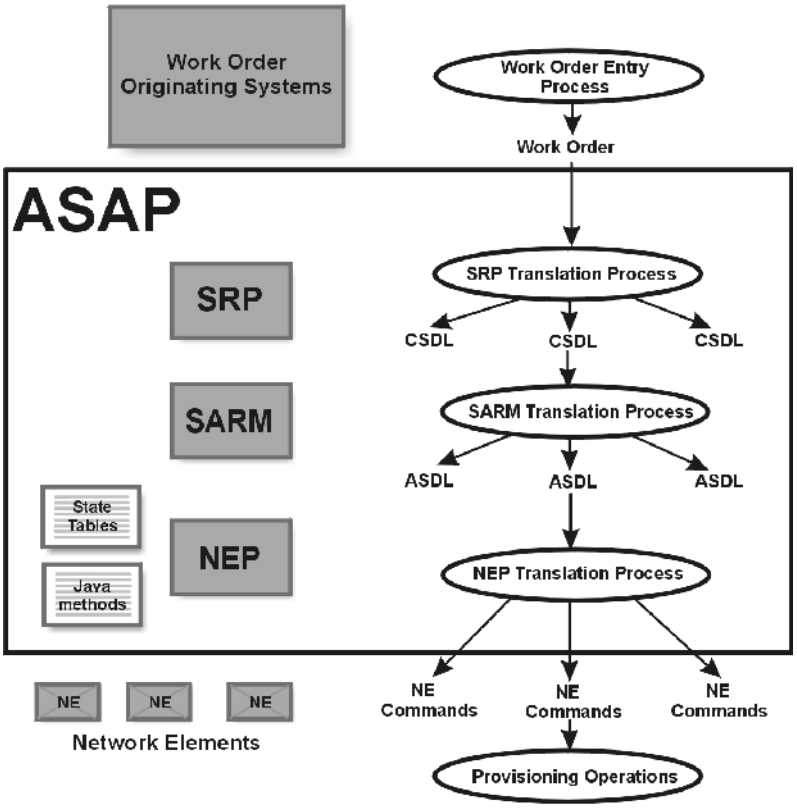
Customer Service Request Sources

The ASAP provisioning process begins when customer service requests are sent from an order management or service order entry system. These requests use either batch or transaction-based interfaces.

The most common format for customer service requests is the conventional work order generated by service representatives or automated systems, such as a web interface or voice response program. Work orders generally require other activities, such as updates to customer, billing, and assignment records, in addition to the provisioning operation.

Figure 2–2 illustrates the work order translations performed as part of the provisioning process.

Figure 2–2 Work Order Translations



The Service Request Translator (SRT) is an optional component that accommodates upstream requests (requests from business and operations support systems) defined within XML documents. Based on the incoming request (XML document), the SRT enables a variety of tasks within a transformation computation, including mapping, data lookup, and message transformation. Mapping tasks allow upstream requests to be broken into one or more downstream requests (request sent to the network element). The SRT output is also an XML document. See *SRT User Guide* for more information.

ASAP Clients

The following client application enables you to define services, create work orders, and configure ASAP.

The Order Control Application Client

The Order Control Application (OCA) Client enables users to query and create work orders, monitor work order-related events, define notifications based on generated events, and manage fallout work orders.

The OCA Client is a Java-based GUI that communicates with the OCA SRP. The OCA Client allows you to interact with ASAP in a flexible manner, providing a single entry point where you can create and manage work orders.

The OCA Client is also available as a Java applet that can be embedded in an HTML Web page and run in a browser.

Access to the Java source code for the OCA Client is provided by the OCA Toolkit option.

Oracle Communications Design Studio

Oracle Communications Design Studio is a GUI-based application that enables users to quickly develop, package, and deploy services in ASAP. Using Design Studio, you can:

- Define new services
- Extend the services that have been defined in an ASAP cartridge
- Reuse existing service definitions
- Deploy services to development or production environments

ASAP Servers

The following server applications contribute to the working of an ASAP system.

The Service Request Processor

The customer's service request systems communicate with one or more 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.

The Service Activation Request Manager

The Service Activation Request Manager (SARM) acts as a request controller that determines and coordinates requests between the various SRPs and NEPs in the ASAP system. The SARM enables you to efficiently manage connections to network elements, including work order priority and load balancing (managing multiple connections to network elements according to configurable thresholds).

The SRP transmits the service actions and parameters to the SARM. The SARM provides centralized coordination and transaction management of provisioning activities across diverse networks in a high-performance environment. Using a data-driven service model, the SARM translates service actions into one or more Atomic Service Description Layer commands (ASDLs) also called atomic actions. Each atomic action represents a single task to be performed on a network element. The SARM then determines the routing for each atomic action and transmits it and its provisioning parameters to the appropriate NEP for provisioning.

The atomic action command parameters are derived from both the service action and global work order parameters.

The Network Element Processor

The SARM routes each atomic action to a NEP. The NEP provides transparent connections to network elements or element management systems. The dialog with the network element is completed by using State Table scripts or Java scripts. State Tables are programs used internally by ASAP to control the dialog with external systems, such as NEs, and perform service order translations.The NEP maps the atomic action to the user-defined script and conducts the activation dialog with the external network devices. To manage a large number of network elements, multiple NEPs can be deployed.

During the NEP translation process, atomic actions become network-element-specific commands. After the NEP has received atomic action commands and their associated parameters from the SARM, the NEP selects either a user-defined State Table (for C-based processes) or the JInterpreter (for Java-based processes) to provision each atomic action command for a network element.

The Admin Server

The Admin server maintains performance data generated by other ASAP components, principally by the SARM. It polls the application servers periodically for their real-time performance data and stores the results in its database. You can reference this performance data, accessible through an API, by using monitoring and reporting tools.

The Control Server

The Control server manages the overall run-time distribution of ASAP components. The Control server has the following management functions:

- System startup and shutdown
- System alarm generation
- System distribution
- Secure data storage
- Encryption key distribution

One Control server must reside on each machine on which ASAP processes run.

ASAP Configuration Management

Configuration management involves two sets of activities:

- Operational configuration – The configuration of ASAP components
- Service configuration – The definition, configuration, and deployment of services

New configuration entries can be added dynamically. Once you add new configuration entries to the database, you can send a request to each relevant ASAP server to load the configuration changes from the database into memory. This enables you to add service actions, atomic actions, service action-to-atomic action mappings, new NEPs, hosts, routings, and network element resources dynamically.

Operational Configuration Management

Using the command-line interface, you can manage the following operational configurations:

- **Operational configuration** – You can configure the operational components of ASAP, including the distribution of ASAP over one or more machines and the definition of ASAP servers.

- **Event/alarm configuration** – You can define system events and specify the circumstances under which system event alarms are issued.
- **Network element configuration** – You can manage mapping relationships, create and maintain NEPs, create network element blackout periods, define a new Host network element, and create new atomic action command routings.
- **Other SARM configurations** – You can configure miscellaneous SARM operations, including atomic action responses for system testing, file system thresholds, modifiable text, ASAP international message conversion setup, SRP-to-SARM security setup, SRP definition to SARM, and management of database monitoring thresholds.

Service Configuration Management

Using the command-line interface, you can manage the following service configurations:

- **Provisioning translation** – You can create and maintain service actions and atomic actions, along with service action-to-atomic action mapping relationships and atomic action-to-State Table or Java script mapping relationships.

Services can be developed according to the Service Model XML Schema. Alternatively, a set of specific services in the form of a cartridge can be purchased or developed. Cartridges provide plug-and-play domain behavior into a core product and include the configuration of the core product. An ASAP cartridge follows the guidelines specified in the *ASAP Cartridge Development Guide*.

Whether services are developed or purchased, the Service Activation Deployment Tool (SADT), Service Activation Configuration Tool (SACT), and the **PostDeploySarFile** script enables you to assemble and deploy generic or cartridge-specific service models. The functionality provided by these scripts can be implemented using the **installCartridge** script.

The **installCartridge** script does the following:

- Facilitates the development of generic service packages, cartridges, and solutions
- Defines a structure for packaging an activation model
- Defines a process for deploying an activation model

You can use this script to bundle and deploy cartridges into an ASAP instance. Packetizing a service bundle also lends itself to version control. The benefits are:

- Reduced deployment time
- Reduced deployment costs
- Ability to layer and version control service models
- Facilitation of cross-product service model deployments

The ASAP Development Toolkit

The ASAP Development Toolkit provides a set of APIs that you can use to customize ASAP client and server applications.

ASAP Client/Server APIs

ASAP provides C/C++, Java, and CML/JMS to create custom SRPs JSRPs, NEPs, and JNEPs or provide programmatic integration with ASAP. These APIs include:

- APIs common to both client and server applications

- Client APIs
- Server APIs
- Java

The ASAP Interpreter and JInterpreter

The ASAP Interpreter and JInterpreter libraries are script engines that allow State Table scripts or Java scripts to be developed, configured dynamically, and run without the need to recompile C/C++ or Java code. This library can be incorporated into application servers to provide internal ASAP State Table scripting or Java scripting capabilities. The interpreter and JInterpreter libraries are provided to build State Table and Java capability in customer NEPS, and JNEPs.

ASAP Utilities

The ASAP utilities are bundled UNIX command-line management tools for developing, monitoring, and controlling ASAP development activities.

ASAP Security Components

This section describes the features and functions of ASAP security components.

User Administration and Security

You can manage ASAP users with Oracle WebLogic Server. Oracle WebLogic Server uses an enterprise-level framework that allows multiple servers to share a common security architecture.

For information on the default permissions that are available to users and user groups, refer to *ASAP System Administrator's Guide*.

Data Security

ASAP provides enhanced security functionality that is designed to respond to the increased security requirement when provisioning over distributed networks.

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 two essential functions: managing secure data and protecting diagnostics files.

ASAP security incorporates the following features:

- **Secure data storage** – Your ASAP security administrator predefines the nature of the secure data and the accessibility by each ASAP server. There are two classes in the secure data: ASAP secure data, such as ASAP database passwords, and custom secure data, such as network element passwords.The ASAP secure data is stored in a Credential Store Factory (CSF) wallet, accessible to each ASAP server. The custom secure data is stored in a central repository called the Master Control server.
- **Secure data encryption** – The CSF wallet that contains ASAP secure data provides transparent encryption functionality. ASAP supports a symmetric secret key encryption method to achieve data confidentiality for custom secure data.
- **Key distribution** – Each ASAP server can locally access the ASAP secure data contained in the CSF wallet. The Master 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, the ASAP server has to follow the predefined key distribution protocol.

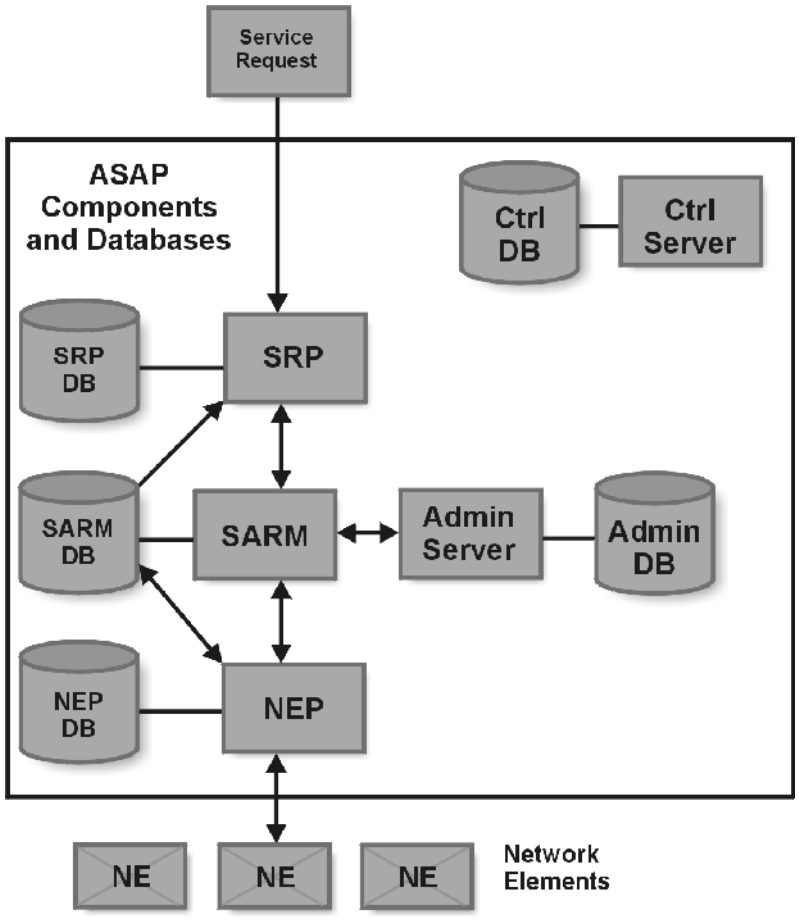
- **Secure NE dialog** – The ASAP NEP diagnostic file contains switch-sensitive information sent to and received from network elements. The NE dialog can be configured to secure the provisioning information.
- **Secure work order information** – Work order information that originates from the upstream system and is sent to network elements contains business-sensitive information. As the work order progresses through several components (SRP, SARM, NEP, and so on), the information is exposed to different diagnostic files. For example, some database queries in the SARM can reveal work order information. By setting a configuration variable or a work order property, you can ensure that work order information is not exposed in any diagnostic files.

ASAP Database Components

Every ASAP component application can have its own database. This close coupling of processes to databases distributes ASAP efficiently in a networked environment.

Figure 2–3 illustrates the process and database components of ASAP.

Figure 2–3 ASAP Process and Database Components



ASAP uses Oracle's relational database management system (RDBMS) as the database engine. A single database can contain one or more ASAP databases. Or every ASAP database in a fully-distributed environment can reside on a separate RDBMS, on

separate machines. This ability to distribute transparently and to scale ASAP accordingly is a feature of ASAP.

The RDBMS Server

The RDBMS employed by ASAP uses an Oracle server. The Oracle RDBMS is a specialized server process that manages ASAP databases. It offers the following features:

- Client/server architecture
- Stored procedures
- Server-enforced integrity and security
- High application availability
- Open, distributed RDBMS
- Scalable high-performance programmable server
- Online backups and maintenance
- Rapid recovery
- Distributed access, data, retrievals, and updates

Sybase Open Client/Open Server

ASAP uses Sybase Open Client and Open Server run-time libraries. The Sybase Open Client provides a library of functions for use when creating Open Client applications that communicate with Sybase Open Server applications. The Open Server offers a set of functions you can use when writing a multithreaded Open Server application to receive requests from Open Client applications. It is a platform independent, multithreading, and remote procedure call (RPC) library.

Services and Network Elements Implemented

You can deploy ASAP across a broad range of services and network element technologies. Tier 1 operators globally are using ASAP for multiple domains including Mobile, Voice (Circuit switched and packet voice), Data (xDSL and HFC) and Video (CATV and Satellite).

Network Technologies Implemented Using ASAP

ASAP has a robust cartridge development program. Oracle offers over 80 certified cartridge adapters for various network elements from different vendors. ASAP customers can license these productized cartridges and receive fully tested and supported solutions for activating services on the various network elements.

Oracle Design Studio allows you to extend or build new cartridge adapters for new network elements or versions that are being introduced into the network.

Interface Protocols/Standards Supported by ASAP

ASAP supports the following interface protocols and standards:

BSS / OSS Interfaces

- Web Services

- OSS through JAVA API (JSR 89)
- SA API
- TCP/IP
- JMS/XML
- C/C++
- CAPI

Network Element and Element Management System Interfaces

- SOAP/XML
- HTTP/S
- SNMP Manager
- TL1
- TCP/IP
  - Socket interface
  - Telnet interface
  - SSH
  - File Transfer Protocol (FTP) interface
  - Secure FTP (SFTP) interface
- Serial asynchronous
- External device driver API for proprietary or custom interfaces as required

> **Note:** Protocols that are not included in the above list can be supported through extensibility.

Cartridge Overview

ASAP cartridges are discrete software components that are developed for the ASAP product. An ASAP cartridge offers specific domain behavior on top of the core ASAP software, and provides the configuration that supports a set of services on a NE.

An ASAP cartridge is not a standalone component, but operates with the ASAP core product. ASAP cartridges offer the following benefits:

- **Reduced Time to Market:** Time to market of new services is reduced through simplified development, implementation, and extension of cartridges on customer sites.
- **Extendable:** Cartridges can be extended to include additional services and components that deliver business value, without requiring changes to the original cartridge.
- **Simplified Effort:** The effort and technical knowledge that is required to perform customizations is reduced.
- **Ease of Installation:** Cartridges can be installed into an ASAP environment without interfering with the existing install base.

An ASAP cartridge allows you to configure ASAP to provision the following:

- NEs from a specific vendor, such as Ericsson or Huawei.
- Technologies, such as GSM or xDSL.
- Services that are supported on the NE, Mobile, Data, or Video.

> **Note:** Cartridges are designed for a specific technology, software load, and service.

An ASAP cartridge supports a particular set of services on an NE. These services are independent of customer-specific service definitions. Professional Services or systems integrators can perform extensions to the cartridge to support customer-specific requirements.

Cartridge Content

An ASAP cartridge contains the following:

- An interface to the NE
- A set of scripts, such as State Tables or Java methods
- A set of atomic actions in the form of ASDL commands
- A set of CSDL commands that form meaningful service actions
- Sample work orders
- Installation scripts

3

Service Request Processor and Java Service Request Processor

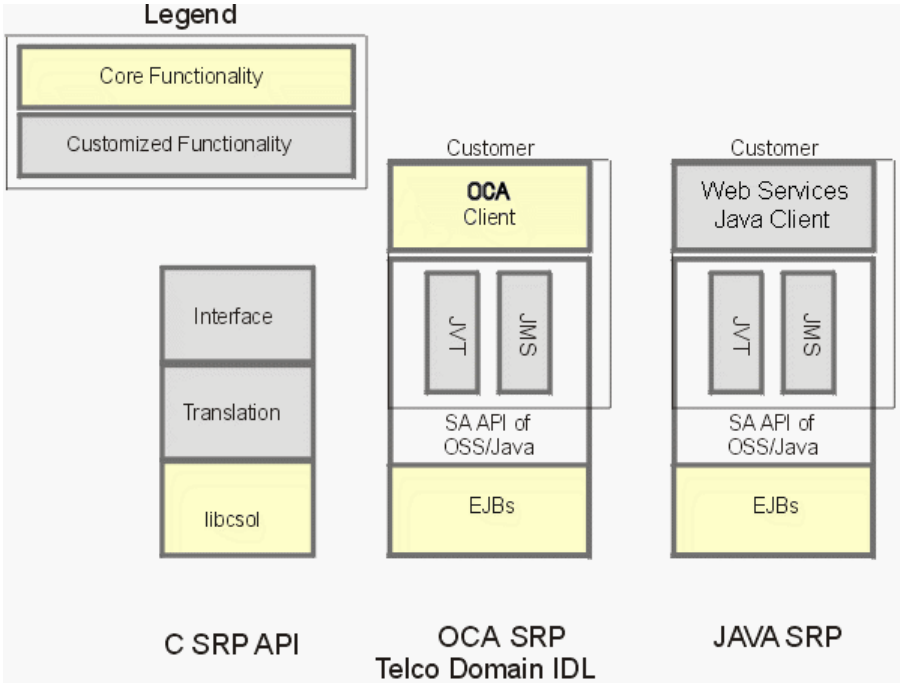
The Service Request Processor (SRP) and Java SRP (JSRP) bridge external systems and the Service Activation Request Manager (SARM). The SRP and JSRP performs the following functions:

- Receives work orders from external sources
- Updates work order status requests to the SARM
- Deletes work order requests to the SARM
- Receives work order event notifications from the SARM
- Submits work order queries to the SARM for additional work order details
- Transmits work order provisioning notifications and details back to the originating system

The SRP and JSRP provide interoperability to interface with one or more Business Support Systems (BSSs) and Operations Support Systems (OSSs) in parallel. Oracle Communications ASAP can interface with one or more of these information systems in parallel using the object-oriented SRP toolkit, the Order Control Application (OCA) SRP, and the Java SRP. These SRPs are customizable to support communication with any upstream system.

Figure 3–1 represents the technology support provided by the SRP and Java SRP.

Figure 3–1 Technology Supported by the SRP



Telecommunications carriers can use ASAP to manage technologies from different vendors and to facilitate the flow-through of information from those devices, using either internal or external networks.

ASAP provides the following types of SRPs that are designed to accommodate a wide variety of upstream technologies:

The SRP provides the following interfaces:

- **Web Services API** – provides a Web Services interface through which external applications can manage service activation activities and operations. The interface is defined in the ASAP Web Service Web Service Definition Language (WSDL) file.
- **Java SRP** – Provides both an XML/JMS, Java Value Type, and Web Service interface into ASAP's provisioning functionality. The Java SRP adheres to OSS/J, an initiative that defines and implements an open, standard set of Java technology-based APIs for OSS.

  The Java SRP can operate with an optional ASAP component, the Service Request Translator (SRT). The SRT enables a variety of tasks within a transformation computation, including mapping, data lookup, and message transformation. The mapping of tasks allows upstream requests to be decomposed into one or more downstream requests, data lookup functions can involve the retrieval data from external systems (such as databases), and transformation actions can involve presentation formatting. The translator is triggered by an XML source document and generates an XML target document as its output. See *SRT User Guide* for more information.
- **OCA SRP** – Co-exists with Java SRP in WebLogic and shares some JSRP interfaces. The OCA Client application uses the OCA SRP.
- **C++ SRP API** – Provides a C++ API to interface with ASAP and is used to support object-oriented technologies. The Object SRP uses native threads to support symmetric multiprocessing and provides maximum scalability of ASAP across

multiple CPUs. Refer to *ASAP Developer Reference* for C++ SRP API design guidelines.

- **C SRP API** – Provides a C API to interface with ASAP.

> **Note:** C SRP API and C++ SRP API enhancements are not supported.

SRP Order Properties

SRP order properties contain ASAP order details that are required for SARM provisioning. Some are optional. You can set and retrieve work order properties using SRP API calls.

The following are some major work order properties:

- **Order types** – Order type properties are used to pass provisioning instructions to the SARM, such as activating or canceling the order.
- **Order scheduling** – Order scheduling properties enable you to specify whether the SARM is to process orders immediately or based on a user-defined due date and time.
- **Timeouts** – The timeout function at the service-order level prevents a work order from staying in ASAP indefinitely. You can configure the timeout period so that a work order fails if it remains in the system longer than required.

> **Note:** The work order timer is set when the SARM starts provisioning the work order or when the work order is in progress. The order is failed after the current ASDL finishes provisioning.

  If the work order fails due to work order or ASDL timeout, ASAP rolls back ASDLs that have completed.

- **Related order properties** – An optional Related Order property is included in the SRP order to identify a prerequisite work order, such as a parent work order upon which a child order is dependent. The SARM activates the child only when the provisioning of its parent is complete. The Related Order property is used for from/to and multiline order processing, revisions, and cancellations.
- **Rollback properties** – ASAP can be configured to roll back ASDLs upon ASDL failure. This field is used to override the default rollback behavior defined in the SARM translation tables.

Batch Orders

Provisioning activities can be batched in the following ways:

- **Single batch orders** – Large orders that contain many instances. Such orders are processed serially within ASAP and are most efficient when all instances in the batch are destined to the same network element. You can specify an error threshold associated with a single batch order that, if exceeded, results in the entire batch being stopped.
- **Multiple orders in a batch** – The SRP provides the capability to aggregate a set of otherwise independent orders into a logical batch group and then activate these orders in parallel. This technique is best suited to a batch consisting of several

instances that span many network elements. You can specify an error threshold associated with such batch groups that, if exceeded, results in the entire batch being stopped.

Once the SRP has completed the translation process, it transmits the SRP version of the work order with the specified properties to the SARM for provisioning. The SARM can either accept or reject the work order.

When the SARM receives the SRP version of the work order, it stores details such as work order properties, CSDLs, and parameters in the SARM database.

SRP Notification Reception

The SRP receives notification events from the SARM while a work order is being provisioned. Specifically, whenever the state of the work order changes (wether the work order is part of a batch or individual), the SARM notifies the SRP. The SRP can then notify the originating system of the change in work order status.

SRP notification events include:

- **Work order timeout** – If the SARM is configured to calculate the work order timeout, it sends a timeout notification to the SRP when the ASDL processing time exceeds the configured time limit. One of the following situations can occur:
  - **Timeout and executing** – The first ASDL timeout that occurs during the provisioning of the work order. The timeout notification is sent to the SRP, and the SARM sets the timer again to provide a grace period for the ASDL and continues provisioning of the ASDL. It does not fail the work order.
  - **Timeout and fail** – Work order timeout or an ASDL timeout for the grace period on a work order. A timeout notification is sent to the SRP and the work order is failed in the SARM.
- **Work order rollback** – This notification type informs the SRP that rollback is being invoked on the work order. In general, Work Order Rollback is produced whenever a work order fails and rollback is configured on one or more ASDLs.
- **Work order failure** – The Work Order Failure notification is triggered by the SARM whenever a work order fails there. The notification includes details on the CSDL that failed.
- **Work order accept** – Transmitted to the SRP when the SARM accepts the work order.
- **Work order estimate** – Transmitted when the SARM receives a work order from the SRP. If the SARM is configured to perform a work order estimated time calculation, it calculates the average time, in seconds, for this work order to be provisioned.
- **Work order completion** – Transmitted to the SRP when the work order completes in the SARM. This notification can report Normal Completion (no errors) or Completion with Exceptions (such as soft errors).
- **Work order start up** – Informs the SRP that the work order has started provisioning in the SARM. When possible, this information is passed back to the originating system to provide feedback on the progress of the work order.
- **Work order soft error** – Generated whenever an ASDL response on the work order is labelled Fail But Continue. If the work order fails or completes, this notification is followed with a Failure or Completion notification.

- **NE unknown** – Identifies a Remote NE on the work order that could not be routed to an appropriate Host NE, causing a SARM Remote NE routing error to occur.

When the SRP receives work order event notifications, it can query the SARM for additional details. Queries are generally made when the order is in a final state such as Failed or Completed, but can be made at any point in the provisioning process.

SRP Information Retrieval

When the SRP receives work order event notifications, it can query the SARM for additional details. Queries are generally made when the order is in a final state such as Failed or Completed, but they can be called at any point in the order provisioning process.

The SRP can query the SARM for the following information.

- **Work order parameters** – Work order parameters are created by the State Tables in the NEP or by the JInterpreter provisioning method used by the NEP, while the ASDL commands are being processed on the ASAP work order. Query results from the NEP such as formatted NE configuration information are transmitted back to the SRP, which can then pass them on to the originating system in their native format.
- **Work order log** – The work order log is the SARM log of all events that took place on a particular ASAP work order. Although you can query any event type, failure is the primary event to be queried. The work order log retrieves the history of all NE commands and responses transmitted to, and received from, all NEs during the processing of a failed work order. This information is useful for analysis.
- **Work order CSDL log** – This function returns SARM CSDL log information for a particular CSDL command on the ASAP work order.
- **Work order CSDL list** – This function returns a list of CSDLs on a particular ASAP work order with a time estimate for each CSDL.
- **Work order revisions** – The Work Order Revisions function records changes made to failed work orders in the SARM so that they can be completed at the NE. This facility provides a “before and after” picture of any modified CSDLs or parameters on the work order, which is an important feature if you update failed work orders from the OCA. This allows the originating system to be notified of all changes to the original work order.

4

The Service Activation Request Manager

The Service Activation Request Manager (SARM) provides much of the functionality for Oracle Communications ASAP. The SARM performs the following functions:

- Maintains connections to all Service Request Processors (SRPs) and Network Element Processors (NEPs) within ASAP
- Controls all ASAP operations in both directions
- Manages host feedback
- Manages connection load balancing

During the provisioning process, the SRP transmits the Common Service Description Layer (CSDL) commands and parameters for a work order to the SARM and receives event notifications from the SARM. The SARM translates these CSDLs into Atomic Service Description Layer (ASDL) commands and then sends the ASDL commands and their associated parameters to the appropriate NEP. The NEP returns the resulting ASDL responses to the SARM, where they are processed.

Figure 4–1 illustrates the various ASAP data components and APIs relating to the SARM process.

Figure 4–1 ASAP Components and APIs

The following sections describe the main order management and connection management features for the SARM.

Order Management

Order management within the SARM refers to the process of scheduling and coordinating the provisioning activities associated with work orders. These activities include releasing immediate orders, managing and releasing future-dated orders, and managing order dependencies (parent/child relationships).

The SARM contains most of the high-level intelligence for network element management, including the following:

- **Rollback Management** – You can configure rollback at the CSDL level or the ASDL level. Should an order be cancelled or encounter an error during provisioning, rollback returns the network element to its original state.

  For more information on rollback, refer to *ASAP System Administrator's Guide*.

- **Managing Dependencies** – When one work order is dependent on another (a parent/child relationship), the SARM manages the dependency and starts provisioning the child order once the parent order is complete.

Order Acceptance/Rejection

After receiving a work order, the SARM either accepts or rejects the work order based on the following:

- Order security
- Order collisions

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.

Order Collisions

The SARM rejects a work order sent by the SRP for provisioning if a version of this work order already exists in the SARM in one of the following states.

- **In-Progress WO** – An existing copy of the work order is currently in progress in the SARM.
- **Completed WO** – An existing copy of the work order has already been completed in the SARM.
- **Timeout WO** – An existing copy of the work order has timed out in the SARM.
- **Configuration error** – The work order has been rejected due to a configuration error, such as an unknown CSDL command, and a copy of the rejected order was saved in the Translation Error state in the SARM database.

Order Processing

The SARM processes a work order by performing one of the following operations.

Update

The Update operation provisions the work order at its due date and time. Once the provisioning begins, the status of the work order is updated to In-Progress.

The activation date and time is not necessarily the same as the due date and time. The SRP can request the SARM to activate the work order at a date and time other than the due date and time, dependent upon business rules in the SRP.

The SARM manages a work order according to its status:

- **New order** – If the order does not already exist in the SARM, the SARM accepts the order.
- **Initial, Held, Reviewed, or Translation Error states** – The SARM overwrites the existing copy of the order.
- **In-Progress, Cancelled, or Completed states** – The SARM rejects the order.
- **Failed state** – The SARM aborts 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 (SRQ_ID).

Cancel

The Cancel operation cancels an existing order in ASAP. The SARM manages a work order according to its status:

- **Non-Existent orders** – If the order does not already exist in the SARM, the SARM accepts the cancellation request and maintains a cancelled record.
- **Initial, Held, Reviewed, or Translation Error states** – The status of the work order changes to Cancelled.
- **In-Progress state** – The order is stopped at the end of the next ASDL command. The ASDL rollback mechanism is used to roll back the provisioning activities performed by completed ASDLs. The status of the work order changes to Cancelled.
- **Completed or Failed states** – The ASDL rollback mechanism is used to roll back all completed ASDLs. The status of the work order changes to Cancelled.
- **Stopped state** – If the work order is stopped without rollback, the completed ASDLs are rolled back and the order is moved into a Cancelled state. If the work order is stopped with rollback, the order is moved directly into a Cancelled state.

Translation Error

The Translation Error operation applies when the SRP transmits a work order that contains translation errors to the SARM. The SARM does not activate the work order; instead, it maintains it in a Translation Error state. The following qualifications apply when the states listed below are updated with a Translation Error 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 translation error work order.
- **Failed states** – the SARM aborts 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.

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 aborts 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.

Stop

The Stop operation halts a work order that is in In-Progress state. It contains an option to specify whether the completed ASDLs must be rolled back. Once the work order is stopped, you can restart it by changing its status to the Initial or Cancelled state.

The Stop operation can only be used on a work order that is in the In-Progress state. This operation is rejected on all orders that are in other states.

Order Types

The SARM accepts the following order types:

- Immediate
- Future
- Batch

Provisioning activities can be batched in the following ways:

- **Single batch orders** – Large orders that contain many instances. Such orders are processed serially within ASAP and are most efficient when all instances in the batch are destined to the same NE. You can specify an error threshold associated with a single batch order that, if exceeded, results in the entire batch being stopped.
- **Multiple orders in a batch** – The SRP provides the capability to aggregate a set of otherwise independent orders into a logical batch group and then activate these orders in parallel with each other. This is best suited to large numbers of instances in the batch that span many NEs for better performance. The user can specify an error threshold associated with such batch groups that, if exceeded, results in the entire batch being stopped.

Connection Management

The SARM contains many features that enable you to efficiently manage connections to network elements.

- **Managing Connect/Disconnect Requests** – When the SARM establishes that a particular ASDL must be routed to a network element, it determines the NEP that manages the network element and the current status of that network element.

When a connection to the network element has been established, the SARM transmits the highest priority ASDL for that network element to the NEP.

- **Primary/Auxiliary Connection Requests** – The first connection opened to a network element is called the primary connection. If the SARM determines that additional connections to a network element are required, it opens auxiliary connections.
- **Spawn and Kill Thresholds** – ASAP enables you to configure the SARM to control the number of connections to a particular network element according to the number of ASDL commands awaiting processing on that network element. If the number of ASDL commands exceeds a configurable limit (the spawn threshold), the SARM automatically transmits a request to spawn another connection to the network element.

  The SARM sends a network element Disconnect request via the NEP according to the number of ASDL commands awaiting processing on the network element.
- **Work Order Priority** – Orders that have higher work order priorities assigned to them are given processing preference at the ASDL level. The work order priority dictates the sequence in which ASDLs are queued for the same network element. This queueing ensures that if there are multiple ASDLs from different work orders in a particular network element pending queue, the higher priority ASDLs are sent to the network element first. The priority of the ASDLs is dictated by the priority of the work order from which they are spawned.

  For more information, refer to *ASAP System Administrator's Guide*.
- **Recent Change Retransmission** – The Recent Change Retransmission (RCR) functionality in the SARM lets you retransmit all provisioning commands in their original provisioning sequence to a particular network element, within a specified time period.

  This functionality is required for some older network elements that may lose their recent change commands due to an unexpected outage, and therefore require retransmission of these commands.

Order Scheduling

The SARM performs the following order scheduling activities:

- **Releases immediate orders** – Batch or individual orders are released immediately upon receipt and are processed by composite priority. CSDLs (and attached ASDLs) within work orders are processed in the order in which they were configured in the work order.
- **Releases future-dated orders** – Batch or individual orders are released at their respective due dates and times. To determine which orders are past due and must be released for provisioning, the SARM polls the database for such orders every user-configured interval. These work orders are processed by composite priority. CSDL (and attached ASDLs) within work orders are processed in the following way:

  a. The SARM checks an OSSJ work order for a **provisioningSequenceNumber** for each CSDL within the work order.

  – If no such number is found, ASAP checks for **originalSequenceNumber**

  – If the value is found, then ASAP applies this value to the ServiceSequenceNumber. If all other CSDLs within the work order also

use the **provisioningSequenceNumber**, then ASAP processes each CSDL based on this value.

b. The SARM checks an OSSJ work order for a **originalSequenceNumber** for each CSDL within the work order.

- If no such number is found, then ASAP processes the CSDLs in the order in which they were received, and gives them a sequence number that start at 5 and increments by 5 for each new CSDL.
- If the value is found, then ASAP applies this value to the **ServiceSequenceNumber**. If all other CSDLs within the work order also use the **originalSequenceNumber**, then ASAP processes each CSDL based on this value.

- **Manages explicit order dependencies** – If the SRP notifies the SARM that a particular work order depends on the completion of another work order, the SARM manages the dependency and starts provisioning the child order once the parent order is complete. This is useful for from-and-to orders, multiline orders, etc.

  The following dependency conflicts can arise:

  - A child work order is not released for processing until the parent work order is processed.
  - A separate cancellation request is required for each child and parent work order.
  - The parent work order is not cancelled until all of its child work orders have been cancelled.

- **Manages implicit order dependencies** – If the SRP does not specify explicit dependencies for future-dated work orders, the SARM manages the dependencies for orders with the same due date and time.

This feature lets you schedule large numbers of orders of equal work order priority for the same due date and time. The SARM prioritizes these orders using the order action, potentially avoiding the Interfering Station or Blocking Service conditions that have implicit ordering of one or more orders. For example, from-and-to orders whose relationship has not been explicitly specified by the SRP.

For each NE, the SARM releases delete action orders first, change action orders next, and add action orders last. It uses the composite work order priority to determine this ordering.

When one work order is dependent on another (a parent/child relationship), the SARM manages the dependency and starts provisioning the child order once the parent order is complete. This feature is useful for from and to and multiline orders.

The following are possible parent/child relationships:

- **Multiple child orders with a parent** – A parent work order can have more than one child.
- **Parent work order transmitted first** – If a parent work order is transmitted before any child work orders, the parent work order is processed first. Child work orders start provisioning once the parent order is complete.
- **Child work order transmitted first** – If a child work order is transmitted before its parent, it is saved in the database in an Initial state until the parent order is complete. Once the parent work order is complete, the Batch Handler picks up the

child order for provisioning. When there are multiple child orders, they are picked up for provisioning in a sequence determined by their composite order priority.

The Batch Handler is a SARM thread that periodically fetches all pending provisioning and cancellation requests for the system (territory in HA mode) and processes them based on the composite priority mechanism.

- **Child order without a parent** – If a child work order is transmitted without a parent, it is saved in the database in Initial state.
- **Child order with in-progress parent** – The child work order is saved in the database in Initial state until the parent work order is complete. The child work order is picked up for provisioning after the parent completes.
- **Child order with completed parent** – If the child work order is an immediate order, it is released at once for provisioning. If the child work order is not an immediate order, it is saved in the database in Initial state, and the Batch Handler picks it up and releases it for provisioning at the scheduled due date and time.
- **Child order with failed parent** – The child work order is saved in the database in Initial state.

Network Element Routing Management

Routing information is defined either in the SARM database tables (static routing) or in the work order itself (dynamic routing).

The static routing feature reads network element information, such as network element technology and software load, from static, user-controlled configuration tables in the SARM database.

ASAP also provides extended dynamic routing functionality. Using dynamic routing, ASAP routes work orders according to routing information that is already contained in the work order (such as IP addresses and user IDs), rather than information configured in the SARM database tables. Based on specific information contained in the work order, ASAP routes the translated ASDLs to the appropriate network element.

Dynamic routing provisioning supports the many smaller network elements that can be found in IP-based networks, but the feature can be used by all of the downstream communication protocols supported by the NEP.

ASAP contains the facilities to analyze work orders and to provision them accordingly. For example, when ASAP recognizes that dependencies exist between service actions, it determines what actions can be performed in parallel. The SARM processes work orders in one of two ways:

- **Serial processing** is performed when the SARM processes ASDLs on a particular work order as there are often dependencies between ASDL commands. The SARM processes ASDLs serially when there are dependencies between different ASDL commands. For example, in a POTS activation, an option can be added only after a line is created. At any given time, there is a maximum of one ASDL being provisioned on a particular work order.
- **Parallel processing** occurs when the SARM processes the ASDLs from different work orders at the same time and can interleave them with each other at the network element interface. In addition, ASDLs on different work orders can use different connections to the same network element, if multiple connections are available for that network element.

Network Element Response Management

In addition to transmitting ASDL commands to the NEP, the SARM is also responsible for interpreting the resulting ASDL responses.

ASDL responses include ASDL success, failure, retry and so forth.

In ASAP, you can define variables that apply to ASDL responses. For example, you can configure a retry threshold should the ASDL command fail to be completed in a prescribed time period. The retry threshold defines the number of times that an ASDL command is retried before being failed by the SARM.

The SARM performs error threshold management to control the release of ASDL commands to a network element and prevent an excessive number of errors from occurring.

The error threshold specifies the number of delayed failures that can occur before the SARM stops the order. This error threshold can be set on the individual work order or, if not specified, assumes a default specified by a SARM configuration variable. This functionality can accommodate batch orders containing many instances of the same provisioning activity within one SARM work order. If a specified number of such instances fail, the SARM stops the work order.

SRP Notifications

In addition to managing responses from the network element, the SARM generates notification events and sends them to the SRP while provisioning a work order.

ASAP system events can be configured to trigger system alarms. For example, if a routing error occurs, or if the provisioning of a work order exceeds a configurable period of time, SARM can generate an event that triggers an alarm in an upstream system.

You can configure system events for each CSDL completion and failure. For example, you can configure the CSDL commands associated with cellular work order provisioning to issue a particular system event, which triggers a system alarm to page a user about the cellular work order failure.

In addition to notifying the SRP of events, the SARM server maintains statistical data on work order provisioning, including the number of orders processed, successfully completed, and requiring user intervention.

This information is available to real-time performance monitoring tools. ASAP provides the ability to audit all transactions involved in work order provisioning. All work order auditing messages are stored in an audit table in the SARM database. The user can query work order audit trail records through an ASAP client or third-party monitoring tools.

5

The Network Element Processor

The Network Element Processor (NEP) is the Oracle Communications ASAP component that manages all interactions with network elements. An NEP sends work orders in the form of Atomic Service Description Layer (ASDL) commands to a network element and receives responses from the network element about network activity.

Through the NEP, ASAP can manage multiple connections to network elements. ASAP supports proprietary State Tables and Java.

State Tables

State Tables are programs that act as an interface between specific network elements and ASDL command execution. They also translate a service request within a customer-based Service Request Processor (SRP) into ASAP work order format.

State Tables interface with network elements through the NEP.

The advantages of using State Tables to interface with network elements are:

- The State Table language is relatively simple and easy to learn, and it does not require the developer to have previous experience with advanced programming languages.This makes State Tables appropriate for switch engineers, who are knowledgeable about network interfaces and the switch commands that must be supported.
- The State Tables provide out-of-the-box action functions for manipulating data, such as switch response handling.
- You can develop custom action functions to manage more advanced logic, such as performing complex mathematical calculations, interacting with custom databases, or supporting custom protocols.
- A variety of communication protocols are supported, including TCP/IP, Telnet, SNMP, LDAP, and FTP.

Java-Enabled NEP

The Java-enabled NEP is fully compatible with traditional State Tables and can still run the State Table Interpreter when Java is not the defined interpreter. A single NEP can communicate with network elements through both the State Tables and the Java interface.

In addition to State Tables ASAP supports a Java-enabled NEP that supports common next-generation protocols such as CORBA, HTTP, SFTP, and XML.

The advantages of using Java to communicate with network elements are:

- Java is a known and accepted programming language.
- You can modify the implementation of Java classes without recompiling the core Java framework.
- Java provides access to a large repository of third-party libraries and allows for easy incorporation of external libraries.
- Java provides the ability to design or structure provisioning classes in an object-oriented fashion, rather than the procedural framework of State Tables.

NEP Features

The NEP contains the following additional features:

- **Customizable NEP database** – You can include network element technology-specific and customer-specific database tables within the NEP database to generate MML commands that are transmitted to the network elements. Such tables can be accessed directly from the State Table or Java method during the provisioning process or can be cached in memory within the NEP for better performance.
- **Program network element response analysis** – Network element responses can be analyzed by the program (State Table or Java Interpreter). After analysis, the ASDL status is returned to the SARM. Switch history information and parameters generated during the provisioning process are passed back to the SARM as specified by the program.
- **Network element blackout support** – Blackouts identify periods when the network element is unavailable to ASAP for provisioning. The NEP provides support for both static (day of the week and time) and dynamic (specific date and time) user-defined network element blackout periods.
- **Automatic NEP port re-enabling** – Whenever an NEP connection attempt to a network element fails, the port within the NEP is disabled. If a network element login attempt fails, the port is also disabled if it has been configured to do so. This allows the NEP to use other ports to connect to the network element, if so configured. The port is disabled to allow manual intervention to determine and address the cause.

  The NEP also provides configurable port re-enable logic that automatically re-enables disabled ports after a user configured time period. This negates the need for manual re-enabling of such disabled ports.

ASAP Cartridges

ASAP cartridges are optional, discrete software components that are developed for ASAP. An ASAP cartridge provides specific domain behavior on top of the core ASAP software and productizes the configuration that supports a set of services on an NE. An ASAP cartridge is not a standalone component, but it operates with the ASAP core product.

ASAP Documentation

For an overview of ASAP documentation, refer to the publication information in *ASAP Release Notes*.