OMA Service Provider Environment Architecture

Approved Version 1.0 – 22 Oct 2009

Open Mobile Alliance
OMA-AD-OSPE-V1_0-20091022-A

Contents

Figures

1. Scope (Informative)

This document provides the architecture and functional components description for OMA Service Provider Environment (OSPE). The relationship between each of the OSPE components and the interfaces to fulfill the requirements from OSPE Requirements Document (RD) are described.

The OSPE focuses on Life Cycle Management (LCM), Service Level Tracing (SLT), and Service Model Management (SMM).

2. References

2.1 Normative References

[OSE-AD] "OMA Service Environment" Open Mobile Alliance, OMA-AD-Service-Environment-V1_0
URL: http://www.openmobilealliance.org/release_program/ose_ad_v1_0_2.html

[OSPE-RD] "OMA Service Provider Environment Requirements", Open Mobile Alliance,
OMA-RD_OSPE-V1_0, URL:http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997,
URL:http://www.ietf.org/rfc/rfc2119.txt

2.2 Informative References

[IMSinOMA-AD] IP Multimedia Subsystem (IMS in OMA) V1.0
URL: http://www.openmobilealliance.org/release_program/ims_v1_0.html

[OMA-DICT] "Dictionary for OMA specifications", Open Mobile Alliance™, OMA-Dictionary-V2_3-20051220-A, URL:http://www.openmobilealliance.org/

[PoC-AD] Push to talk over Cellular (PoC) – Architecture V1_0
URL: http://www.openmobilealliance.org/release_program/poc-v1_0.html

[TMF] TeleManagement Forum™,
URL:http://www.tmforum.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purposes of the present document, the terms and definitions given in the following apply:

Attribute Qualifier	An optional annotation, which is understood by specific applications. The annotation indicates specific usage or functional roles.
Catalogue	A logical collection of information associated with Service Packages, Service Models, Service Deployment Instances, and Service Instances.
Component	A replaceable/reusable unit that is responsible for a particular set of functionality and associated information. A component forms part or all of an enabler. [OMA-DICT].
Component Life Cycle	The set of steps a component go through from idea, to creation, to introduction in the service, to retirement (when a component is removed from the Service).
Inherent Attribute	An attribute independent of any specific service used in the management of Service Models.
Life Cycle Management	The sequence of steps of deployment (including download, installation, configuration, activation, and publishing), update (including download, installation, and activation), and removal/withdrawal of services and components.
OMA Service Provider Environment	A system that consists of Resources that are used to construct end-user services. Examples of an OSPE may include an Enterprise network, Mobile network, or Third Party Service Provider.
Provisioning	The process of setting data (e.g. user data, service subscription data, installation and configuration data, etc.).
Resource	A resource is any l element, component, enabler, function or application that can send, receive, or process requests. A Resource is an abstract concept that represents a capability, e.g. a network element, in a Service Provider’s domain.
Service Attribute	Represents an atomic or compound data element associated with the service. A Service Attribute can have zero or more Attribute Qualifiers.
Service Deployment Instance	Representation of the aspects associated with the lifecycle (e.g. from the service installation to the final tests and service activation, and component upgrades) of a service in an OMA Service Provider Environment, according to its Service Model.
Service Level Tracing	Service Level Tracing is the ability to capture and log all relevant information at each Component particpating in the execution of a specic service, that is initiated either by an end user or a Component.
Service Life Cycle	The set of steps a service goes through from idea, to creation, to introduction in the Service Provider environment, to retirement (when a service is removed from the Service Provider environment). [OSPE-RD]
Service Model	A structured representation of services, , Resources, and their interrelations, Inherent Attributes and Service Attributes for a given service.
Service Package	A Service Package consists of services grouped under one commercial package or bundle and that is offered to customers (e.g. end-users). [OSPE-RD]
Service Subscription Provisioning	The process that includes all the steps needed in order to fulfill a service subscription request from a user/subscriber. The process may include the following steps: provisioning of information in the end user profile (e.g. traditional settings in the HLR), provisioning of some user information and preferences into a service system, settings in the charging rules for a user, setting for policies to allow request for the user/service, etc. [OSPE-RD]

Tracing The mechanism to supportcapturing and recording od runtime information. This is achieved by specifying standard Component interfaces to handle aspects such as start tracing, stop tracing, and tracing result retrieval.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in the following apply:

LCM Life Cycle Management

OAM Operation and Maintenance

OMA Open Mobile Alliance

OSE OMA Service Environment

OSPE OMA Service Provider Environment

SLT Service Level Tracing

SMAC Service Model & Catalogue

SP Service Provider

4. Introduction (Informative)

This Architecture Document (AD) describes the features and architecture of the OSPE enabler. It gives a high level overview of how OSPE is useful to other OMA enablers and non-OMA resources. The OSPE addresses the need for standards on deploying services and components, consistent identification of entities collaborating to offer the service, and assistance in localizing faults associated with services that are offered by a Service Provider (SP).

4.1 Version 1.0

4.2 Planned Phases

This architecture document covers all requirements of OSPE. The OSPE Reference Release will not include any interface specifications.

5. Architectural Model

According to OSPE requirement document [OSPE-RD], OSPE will focus on:

1) Life Cycle Management
   OSPE will support Life Cycle Management of components and services.
2) Service Level Tracing
   Service Level Tracing is the ability to capture and log all relevant information at each Component participating in the service execution of a specific service that is initiated either by an end user or a component.
3) Service Model Management
   The ability to manage information about services, and resources, as well as their relations and versions.

Because one or more enablers may be involved in the execution of a service, OSPE has to be able to interact with OMA enablers and some non-OMA resources to make the activities of Life Cycle Mangement (LCM) and Service Level Tracing (SLT) take effect.

5.1 Dependencies

No dependencies have been identified.

5.2 Architectural Diagram

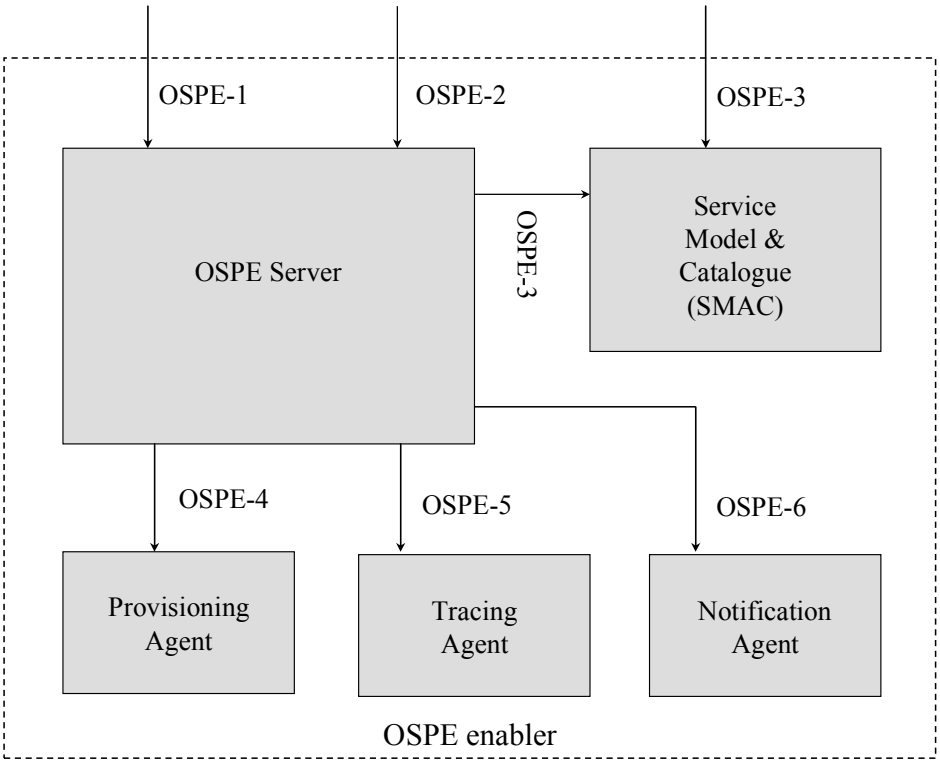

Figure 1. OSPE Enabler Architecture

Figure 1 illustrates the OSPE enabler architecture. The OSPE Server receives LCM and SLT requests through OSPE-1 and OSPE-2 respectively, maps the request into a sequence of actions that are specified by the SP and executes them by invoking functions in Provisioning Agent, Tracing Agent, and Notification Agent (through OSPE-4, 5, and 6). The OSPE Server uses service, component, and resource meta-data stored in Service Model & Catalogue (SMAC) through OSPE-3.

5.3 Description of Components and Interfaces

This section describes the components and interfaces identified in Figure 1. OSPE components include:

- **OSPE Server**
  The OSPE Server uses resource and service meta-data dealing with dependencies and configuration/provisioning information to execute LCM and SLT functions. Detailed functions include:
    - Receiving request and mapping the request into a sequence of individual actions that have been specified by the SP.
    - Issuing LCM commands (e.g., start/stop/pause) and/or data configuration messages to Provisioning Agent.
    - Issuing tracing commands (e.g., initiation/termination tracing, marking a service) to Tracing Agent for initiating tracing operations and retrieving/receiving tracing information.

  In addition, the OSPE Server may send notification messages to Notification Agents (e.g. interfacing with back office systems). Changes in OSPE-managed data may be made available through notifications to other resources in the SP environment, that are responsible for managing and/or storing such data (e.g. to perform changes to their data, when they are notified). For example, when withdrawing a service, OSPE will send out a notification to related enablers or backend systems, announcing the change of service data (service life cycle status), so that the notified enablers or backend systems can take actions for ensuring the post-condition of withdrawing the service is satisfied (e.g., all subscriptions to this service are cancelled).

- **OSPE Agents**
  OSPE Agents (including Provisioning Agent, Tracing Agent, and Notification Agent) are components that receive (e.g. tracing, provisioning, or notifications) messages and either execute them or delegate execution through I2 interfaces. Enabler or other resource implementations can be notified, provisioned and/or traced by OSPE by implementing Notification Agents, Provisioning Agents, and/or Tracing Agents. .
- **Service Model & Catalogue (SMAC)**
  SMAC is a component that manages the Service Models, the Service Deployment Instances and catalogue data. OSPE SMAC manages data relating to: Service Models and the Service Deployment Instances, Service Packages, and partner Service Provider data. This data supports OSPE in order to execute at least LCM and SLT.

The following is a list of the OSPE interfaces:

- OSPE-1: interface that exposes OSPE's LCM functions.
- OSPE-2: interface that exposes OSPE's SLT functions.
- OSPE-3: interface that provides get and/or set operations for information associated with Service Packages, Service Models, and Service Deployment Instances (including their Service/Inherent Attributes).
  Note that OSPE Server uses OSPE-3 to get or set SMAC data.
- OSPE-4: interface for the Provisioning Agent to receive LCM commands and/or data configuration commands.
- OSPE-5: interface for the Tracing Agent to receive a SLT initiation commands and report tracing logs.
- OSPE-6: interface for the Notification Agent to receive notifications.

5.3.1 Components

5.3.1.1 OSPE Server

The OSPE Server provides the basic mechanism for Service Providers to create, execute and manage the service LCM and SLT processes according to Service Provider's own requirements.

When receiving a request through the OSPE-1 or OSPE-2 interfaces, the OSPE Server decomposes the request into actions (including provisioning actions, tracing actions, and notification actions) based on Service Provider choices and the dependencies defined by service/component data maintained by the SMAC component. Provisioning actions, tracing actions,

and notification actions are sent to the Provisioning Agent, Tracing Agent, and Notification Agent respectively using interfaces OSPE-4, OSPE-5, and OSPE-6.

5.3.1.2 OSPE Agents

The OSPE Agents receive messages through the OSPE-4, OSPE-5, or OSPE-6 interfaces, and cooperate with target resources to perform the OSPE actions. The following is a list of responsibilities for OSPE agents:

- Provisioning:
  LCM operations and data configuration actions for target resources;
- Tracing:
  SLT-related actions and reporting tracing data for target resources;
- Notification:
  Interacting with target resources for actions based on notification received.

Any of the OSPE Agents can be implemented in the server entities of a target resource, or stand alone and interact with target resources by proprietary or I2 interfaces. Some details of these two implementation types can be found in the informative Appendix C.

5.3.1.3 SMAC

In order to support the functions mentioned in Section 5.3 (such as being able to compute the dependencies), the SMAC can parse and traverse Service Models, and supports registration and management of Service Deployment Instances, Service Packages, and Partner Service Provider Data. Elements of the Service Model can have Inherent Attributes (for example its life-cycle state, its version, or its location) and/or Service Attributes (for example maximum number of concurrent users). SLT and LCM activities such as Provisioning are example users of this component. The SMAC understands the syntax of Service Attributes but not their semantics, or meaning.

NOTE: Appendix B contains an informative description of the data model maintained by SMAC.

For example, assume the existence of a Service Attribute conveying the maximum number of concurrent users. Also assume that this Service Attribute is qualified with a Qualified Attribute with the value “provisioning”. A provisioning process (e.g. executed in the OSPE Server) will be able to query Service Attributes and filter for Service Attributes with a certain Qualified Attribute. Assume the provisioning process retrieves the Service Attribute conveying the maximum number of concurrent users; it may use this value when executing the provisioning process since the provisioning process is expected to syntactically and semantically understand Service Attributes relevant for and qualified as “provisioning”.

5.3.2 Interfaces

5.3.2.1 OSPE-1 (OSPE LCM requests interface)

OSPE-1 is the interface for exposing OSPE’s service LCM and component LCM functions to requesters. OSPE Server receives the LCM requests and returns corresponding results to the requestor through OSPE-1 interface.

The OSPE-1 interface shall support following functions:

- Services LCM requests (e.g., installing, activating/deactivating, modifying, packaging, or withdrawing a service).
- Components LCM requests, (e.g., installing, upgrading, configuring, starting/stopping/restarting, and removinge a component).
- Service data/component data configuration requests.
- Service data/component data retrieving requests.

5.3.2.2 OSPE-2 (OSPE SLT requests interface)

OSPE-2 is the interface for exposing OSPE's SLT related functions to requesters. OSPE Server receives the SLT requests and returns corresponding results to the requestor through OSPE-2 interface. The OSPE-2 interface shall support following functions:

- Service tracing requests (i.e. activate service tracing, deactivate service tracing).
- Service tracing data reporting return (i.e. push) from OSPE Agent. The OSPE Agent will use the call-back information previously provided by the OSPE server via OSPE-5 (see 5.3.2.5).
- Services tracing data retrieving requests (i.e. pull).

5.3.2.3 OSPE-3 (Data management interface for SMAC)

OSPE-3 is the interface supporting registration and management of service and component related data for SMAC. The OSPE-3 interface supports functions including service registration / unregistration, component registration / unregistration, and data create/read/update/delete manipulations:

- Service/component registration:
  while receiving service/component registration request through OSPE-3, inserting service/component related catalogue data into SMAC. Manipulations should be verified for consistency against the Service Model.
- Service/component unregistration:
  while receiving service/component unregistration request through OSPE-3, removing the service/component related catalogue data from SMAC. Manipulations should be verified for consistency against the Service Model.
- Management of Service and Component related data (not related to Registration):
  - Creation and updating of Catalogue data. Manipulations will be verified for consistency against the Service Model.
  - Deletion of Catalogue data. Manipulations will be verified for consistency with pre-, post conditions.
  - Reading of Catalogue data (including calculating dependencies) by requests and return results.

5.3.2.4 OSPE-4 (OSPE provisioning interface)

OSPE-4 is the interface for Provisioning Agent to receive LCM commands and/or data configuration commands. OSPE-4 interface supports the following functions:

- Receiving LCM configuration messages at the corresponding Provisioning Agent of target resource.
- Receiving data configuration messages at the corresponding Provisioning Agent of target resource.

5.3.2.5 OSPE-5 (OSPE tracing interface)

OSPE-5 is the interface for receiving tracing commands for target resources and sending tracing data to the OSPE Server. The OSPE-5 interface supports the following functions:

- Receiving tracing initiation/termination messages at the corresponding Tracing Agent for target resources. The tracing initiation message will contain parameters to indicate to the OSPE Agent how/where to provide traces or pointer to traces back to the OSPE server (i.e. call-back information).
- Receiving marking commands at the corresponding Tracing Agent for target resources.

5.3.2.6 OSPE-6 (OSPE notification interface)

OSPE-6 is the interface for receiving a notification request for target resources. For example, when stopping a running component X, a notification message is used indicating the status change of the component X. Confirmation of the notification may be sent back through the same interface.

The OSPE-6 interface supports receiving notifying messages for target resources about status and/or configuration change of any service/components..

5.4 Flows (Informative)

5.4.1 Life Cycle Management

This flow describes the interaction between OSPE entities to perform LCM. It consists of:

- Component Life Cycle Management
- Service Life Cycle Management

In general, the following considerations can be applied to the following flows:

- There are several requirements stating that OSPE must provide mechanisms so that a Service Provider can make sure preconditions or post-conditions are met on LCM operations. In order to fulfil these requirements, OSPE will not check pre and post-conditions by itself, but will send notifications to the proper actors or entities during the LCM flows, so that the Service Provider can perform the proper actions before and after an LCM operation is done (e.g.: send a notification before stopping a service, so that Service Provider can check that there are no active subscriptions to that service before proceeding).
- Requirements ensure that a “roll back” for any operation requested to the OSPE enabler will be available.

5.4.1.1 Component Life Cycle Management

5.4.1.1.1 Deploying a new Component

The figure below shows an example of OSPE deploying a new component.

After physical installation and the registration of component data in the OSPE SMAC, the component deployment is ready to be completed by OSPE. When deploying a new component, the OSPE requestor will initiate a component deployment request with necessary information provided. OSPE Server will then execute the process for this request.

This deployment request might include information indicating which steps are to be considered in the deployment flow (e.g., indications of whether installation will be needed or not).

All the information needed to populate the element(s) of a Service Model should be registered, such as software version, functions and capabilities, provisioning interfaces, and tracing interfaces into OSPE server. Besides these, the dependencies between components should be declared for the new component in SMAC, e.g. POC enabler will use presence capability. If installation is to be part of the deployment process, information needed for that (e.g.software download instructions) is to be registered also in this step.

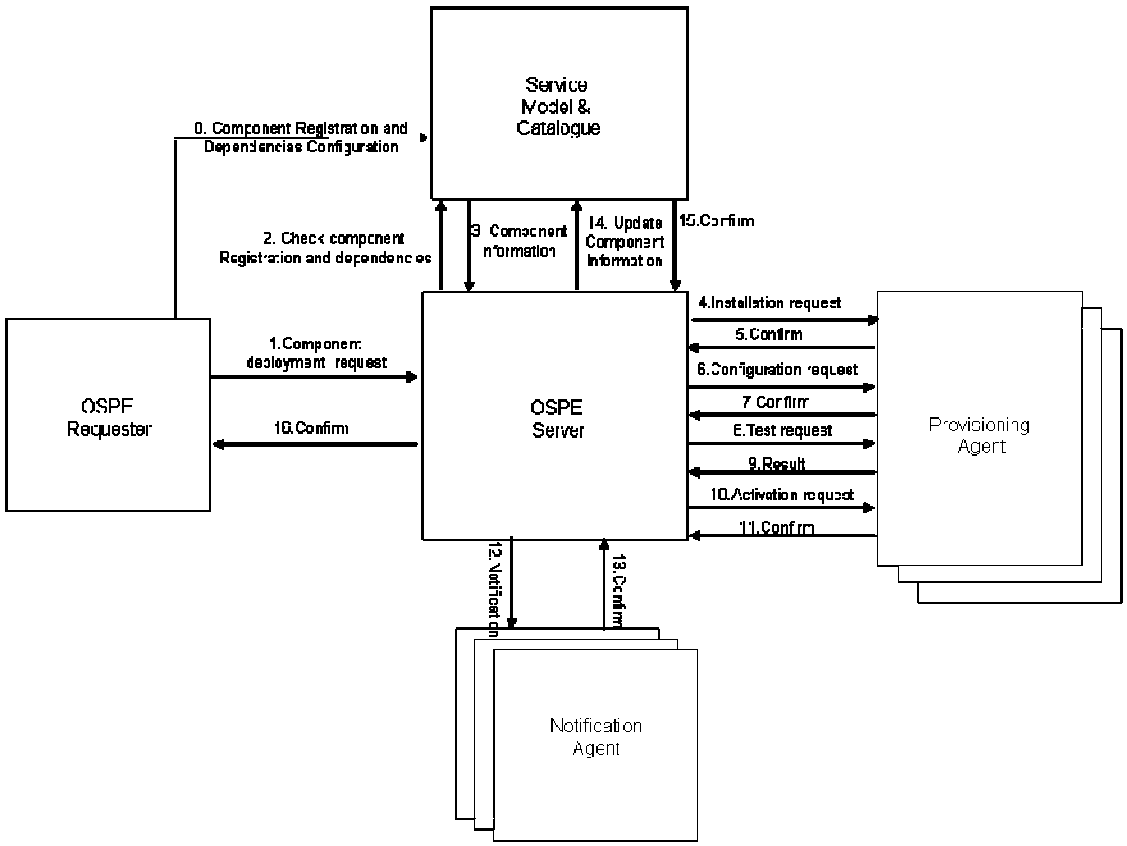

Figure 2. OSPE Deployment of a new Component

See Figure 2. First after receiving the required information from OSPE requestor, the OSPE Server issues a request to SMAC to retrieve all the information needed for the component deployment (e.g. component dependencies, tracing interfaces, interface protocols, etc.). OSPE Server verifies all necessary information is properly available. Once this information is retrieved from the SMAC, OSPE server will then generate the appropriate messages for installation and/or configuration, and send them to the Provisioning Agent for performing the requested operations. This process may be repeated several times.

Installation in this AD stands for the functionality of being able to download a piece of code from some source (e.g. a 3rd party) and install it on any given resource of the Service Provider.

After component configuration, Service Provider may choose to test the functions or performance of the component. The process of deploying the component should be terminated and an error code is returned to OSPE requestor if the component fails in the test.

If the component passes the test, OSPE Server may notify the related enablers/components/back-end systems for final preparation. Finally, the OSPE Server activates the component based on the schedule of Service Provider and responds to OSPE requestor. The activation process requires updating information in the SMAC. In such a case the OSPE server would update the proper attributes in the Component registered information in the SMAC. (This is shown as step 14 in the figure, but it may happen at other steps as well).

5.4.1.1.2 Modifying a Component

Figure 3 shows an example of OSPE modifying an existing component.

After a component is deployed into the OMA Service Provider Environment, modifications (such as reconfiguration, deactivation, reactivation, upgrade, downgrade, etc) may be required by Service Provider.

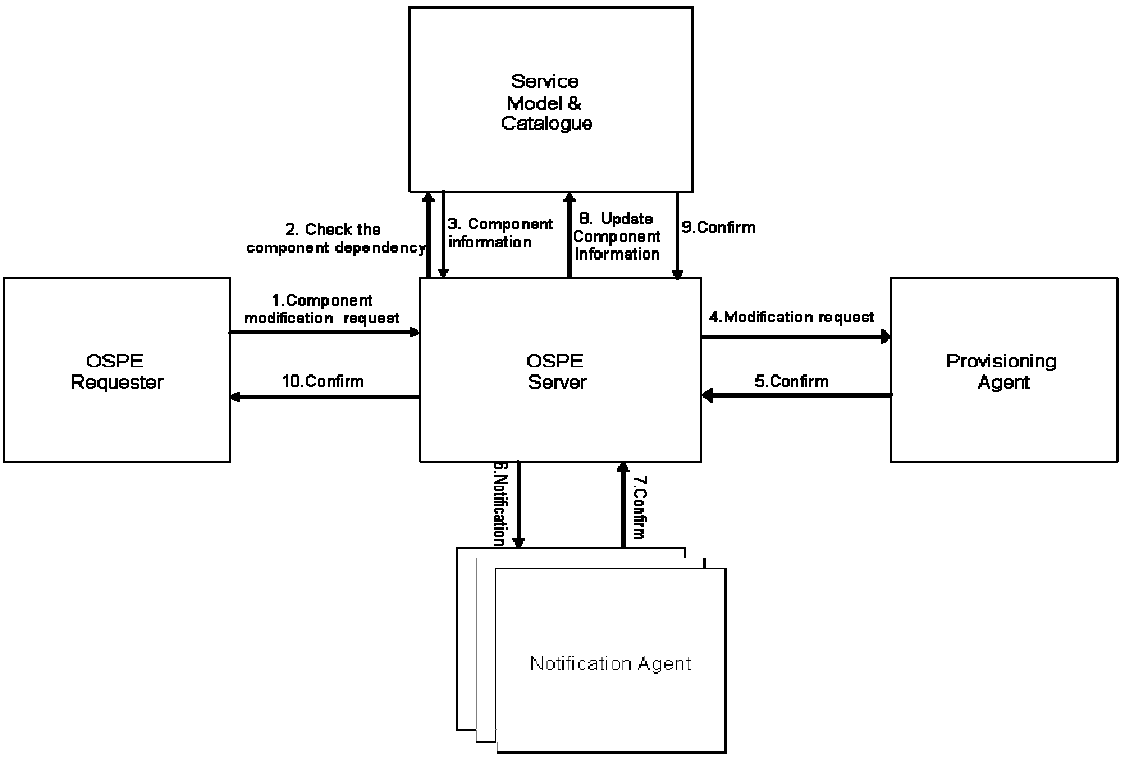

Figure 3. OSPE Modification of a deployed Component

OSPE requestor sends the modification request to OSPE Server. The OSPE Server then executes the process for this request. After checking component information (dependencies, interfaces, etc.), the OSPE Server should assure that the pre-conditions are fulfilled and take necessary actions. Then OSPE Server identifies the modification type and generates the appropriate message (such as reconfiguration, deactivation, reactivation, upgrade, downgrade, etc) according to the information provided by the SMAC. The message is sent to the Provisioning Agent. Note that several actions may be involved in one modification request. For example, the modification request may include checking the component dependency, deactivation, reconfiguration, and reactivation. After this step, OSPE Server may notify the related enablers/components/back-end systems about the change according to the dependency with or requirement from other systems.

Additionally, the component requires changing some information in the SMAC (e.g.: component status). In such a case the OSPE server would update the proper attributes in the Component registered information in the SMAC (shown as step 8 in the figure, but it may happen at other steps as well).

5.4.1.1.3 Removing a Component

Figure 4 shows an example of OSPE removing an active component.

OSPE requestor sends the component removal request to OSPE Server. OSPE Server then executes the process for this request. After checking the dependency of component, the OSPE Server may notify the related enablers/components/back-end systems about the component removal according to the dependency with or requests from other systems (e.g.: in order to check that there are no active component/services still using the component to be removed). After confirmations are collected, OSPE stops, deactivates, and possibly removes the component (if indicated so by the request).

“Remove component” in this OSPE flow stands for removing/deleting a piece of code from a resource.

There might be a need for some additional steps (not shown in figure), like, for instance, stopping the tracing activity (if it was previously activated) before stopping the component. In such a case, requests to the tracing agent would be needed.

The change of status of the component (stopped, deactivated, removed) requires some changes in the component information registered in the SMAC (e.g.: component status information).), In such a case the OSPE server would update information in the SMAC (shown in step 12 in the figure, but it may happen at other steps as well).).

Then the component will be unregistered from SMAC (corresponding component data will be removed from SMAC).

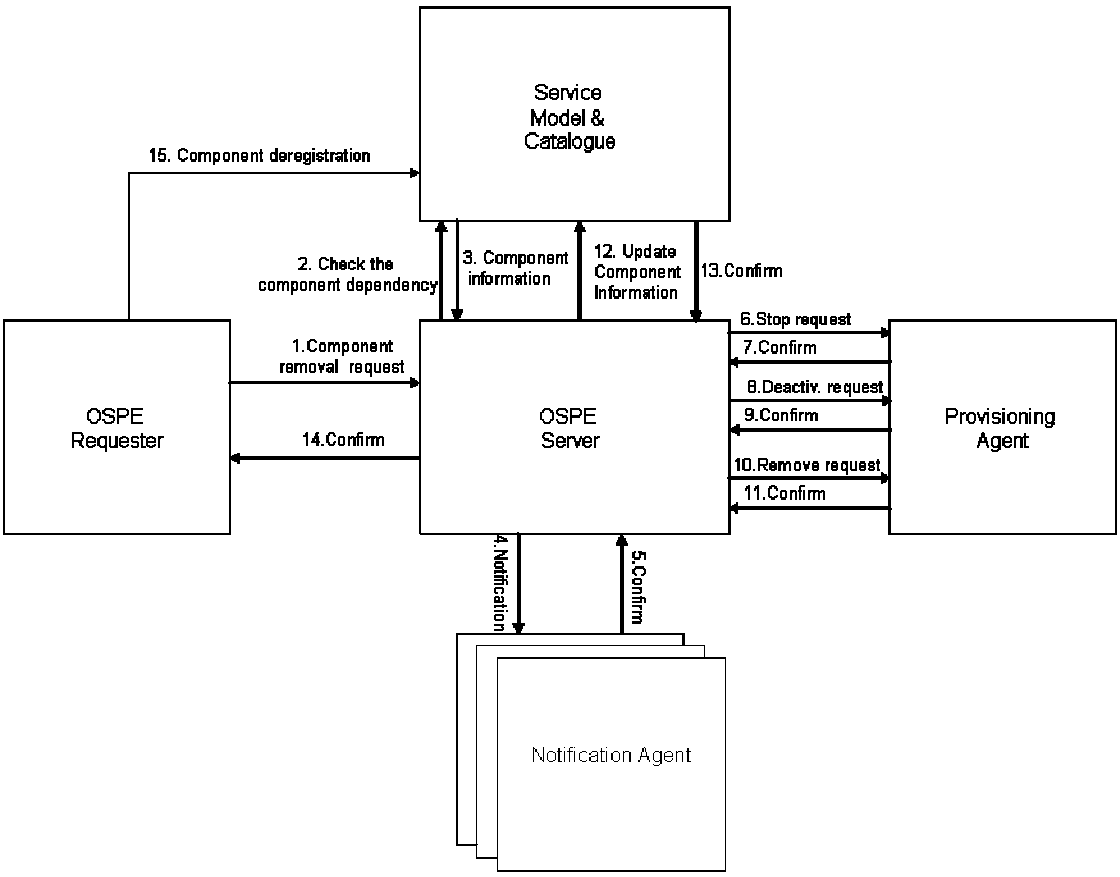

Figure 4. OSPE Removal of a deployed Component

5.4.1.2 Service Life Cycle Management

5.4.1.2.1 Deploying a new Service

Figure 5 shows an example of OSPE deploying a new service.

When a new service is to be deployed in the OMA Service Provider Environment, the OSPE requestor will initiate a service deployment request with all the information needed to populate the Service Model. OSPE Server will execute the process for this request. Administrators will be involved if approval is needed.

After physical installation of related components, OSPE Server configures and registers the service. All the required information should be registered, such as software version, functions and capabilities, the service usage interface (e.g. a URL to call the service), the service subscription provisioning interface (to properly provision user information when subscribing to the service), and accounting interfaces (needed if the service is to generate accounting information on demand from the Service Provider). Besides these, the dependencies among services and components should be declared for the new service in SMAC. Service Provider will register the dependency between the new service and other components or services, such as location capabilities, messaging, and charging.

First after receiving a request for Service Deployment, the OSPE Server issues a request to SMAC to verify that all the information needed for the Service Deployment is properly available in SMAC. Once this information is retrieved, the OSPE Server decomposes the service deployment request into several configuration actions and performs them with the help of Provisioning Agent. The configuration request may contain the SLA of the service or some parameters of the service in the related component. Note that several components may be involved in the service configuration causing steps "4 & 5" to be potentially repeated.

After the configuration is finished, the OSPE server may publish the service to the appropriate system (e.g. user portal system) by using a notification agent, after which the different components of the service would be activated. It is necessary also to update information in the SMAC (e.g.: service status info). In such a case the OSPE server would update the proper attributes in the Service Information on the SMAC. (This is shown as step 10 in the figure, but it may happen at other steps as well).

Then a confirmation is sent as response to OSPE requestor.

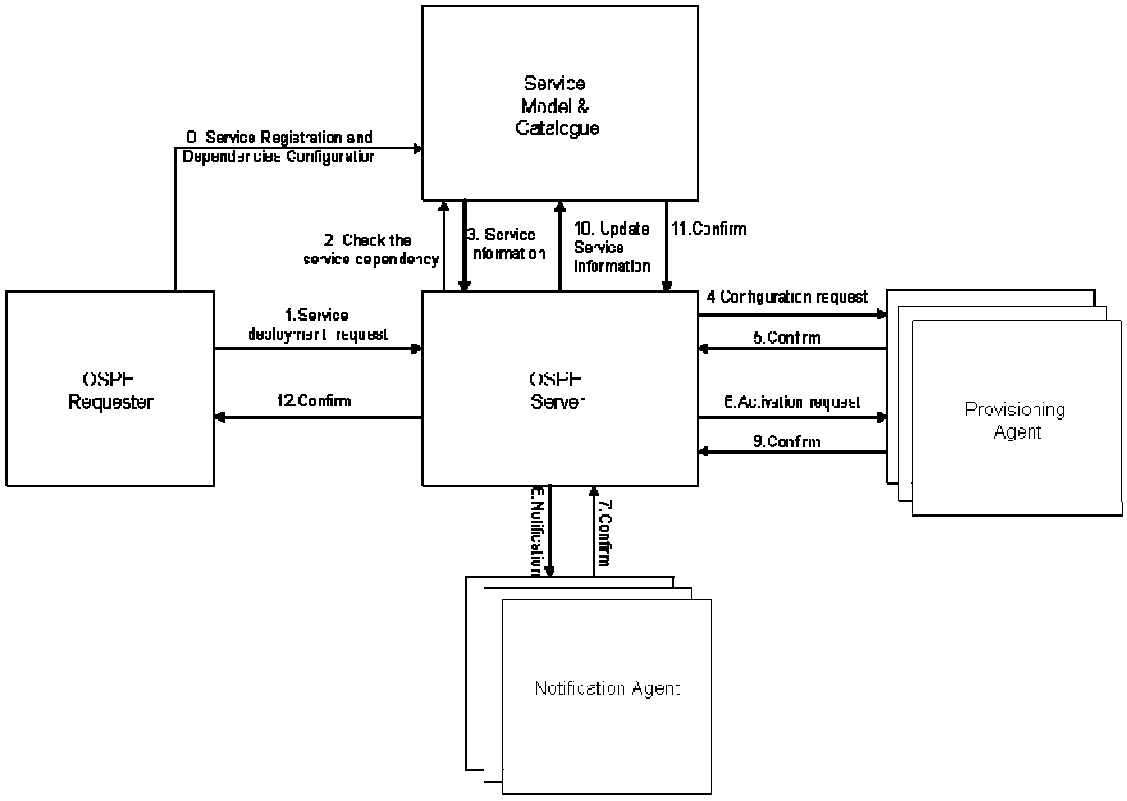

Figure 5. OSPE Deployment of a new Service

5.4.1.2.2 Modifying a Service

Figure 6 shows an example of OSPE modifying an active service.

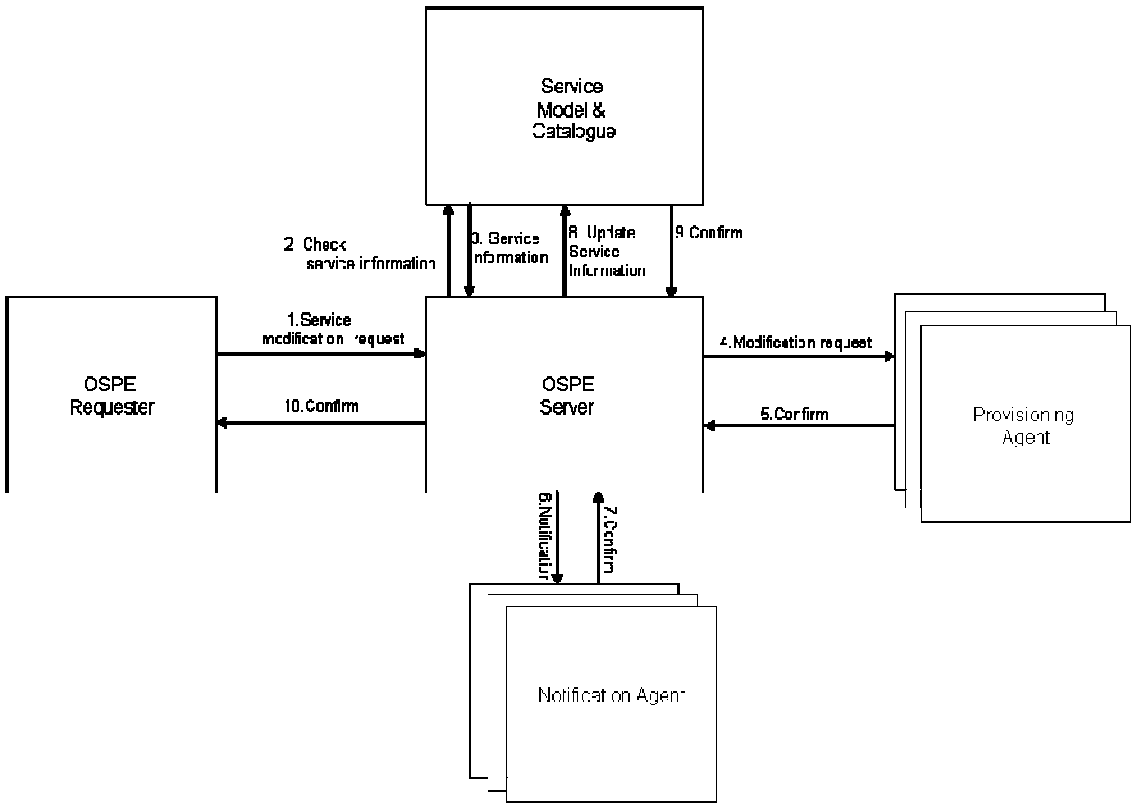

Figure 6. OSPE Modification of a deployed Service

After a service is deployed, modifications (such as reconfiguration, deactivation, reactivation, upgrade, downgrade, etc.) may be required by Service Provider. OSPE requestor sends the modification request to OSPE Server. The OSPE Server then executes the process for this request. After checking service information (dependencies, interfaces, etc.), the OSPE Server should verify the pre-conditions and take necessary actions. OSPE Server finds which components are involved based on service dependency obtained from SMAC (step 2 & 3), identifies modification type, composes suitable message (such as reconfiguration, deactivation, reactivation, upgrade, downgrade, etc.), and sends the message to involved components. This step may be repeated if several components are involved.

Note that several actions may be involved in one modification request. For example, the modification request "reconfiguration" may include deactivation, reconfiguration and reactivation. After the modification is finished, the OSPE Server will notify the related enablers/components/back-end systems about the change according to the dependency with or requirement from other systems.

Additionally, the service modification requires changing some information in the SMAC (e.g.: service status). The OSPE server would update the proper attributes in the Service information on the SMAC (shown as step 8 in the figure, but it may happen at other steps as well).

5.4.1.2.3 Withdrawing a Service

Figure 7 shows an example of OSPE withdrawing an active service.

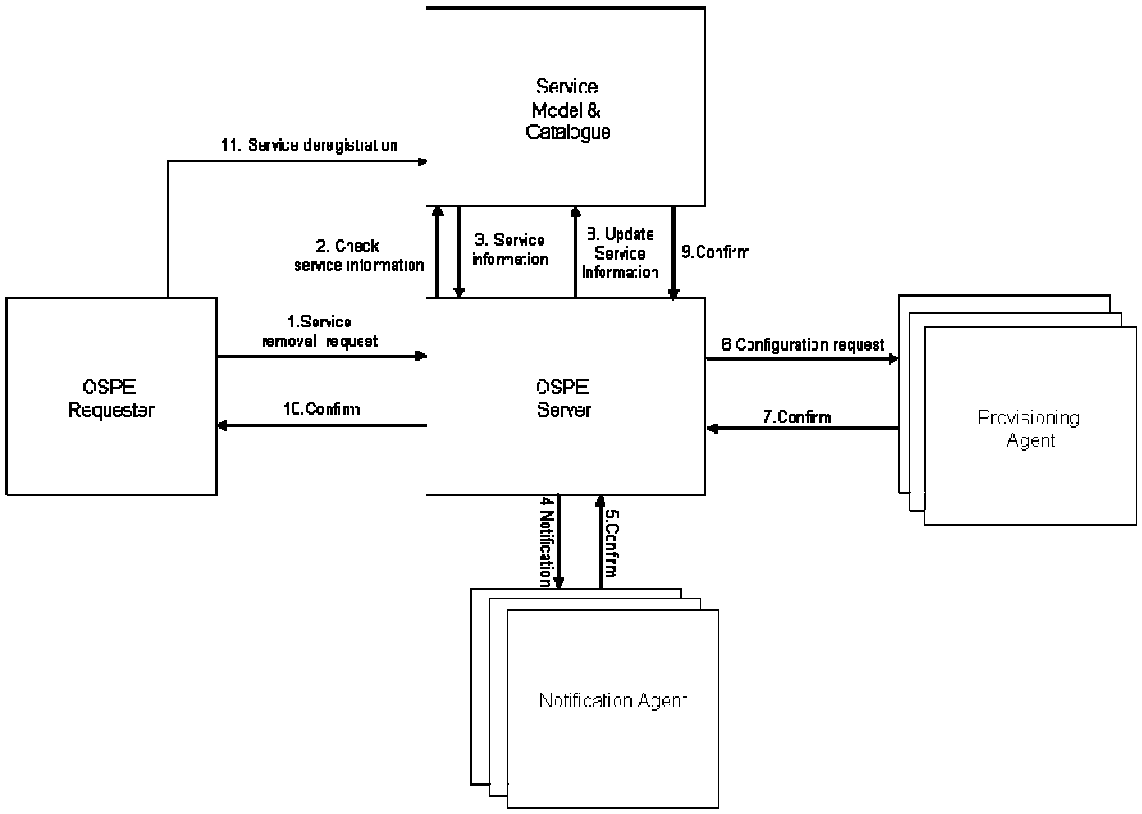

Figure 7. OSPE withdrawing of a deployed Service

OSPE requestor sends the service removal request to OSPE Server. The OSPE Server will execute the process for this request. Once Service Information is obtained (dependencies, interfaces, etc.), the OSPE Server should assure the pre-conditions by checking service dependency and take necessary actions. OSPE Server first finds which components should be involved in this service withdrawing through SMAC and notifies the related enablers/components/back-end systems about the service withdrawn (so that OSPE Server provides a mechanism that can be used by the Service Provider to make sure that there are no active subscription to the service to be removed). After that, OSPE server sends configuration request to the related components.

The change of status of the service requires some changes in the service information registered in the SMAC (e.g.: service status information).The OSPE server would update information in the SMAC (shown as step 8 in the figure, but it may happen at other steps as well).

Finally, OSPE server will return a confirmation to OSPE requestor so that it can complete the service withdrawing process (deleting all related information in the SMAC).

5.4.2 Service Level Tracing

This flow describes the interaction between OSPE Server, SMAC, and Tracing Agent to perform SLT.

5.4.2.1 Turning on Service Level Tracing Function

Figure 8 shows how Service Provider turns on ‘tracing’.

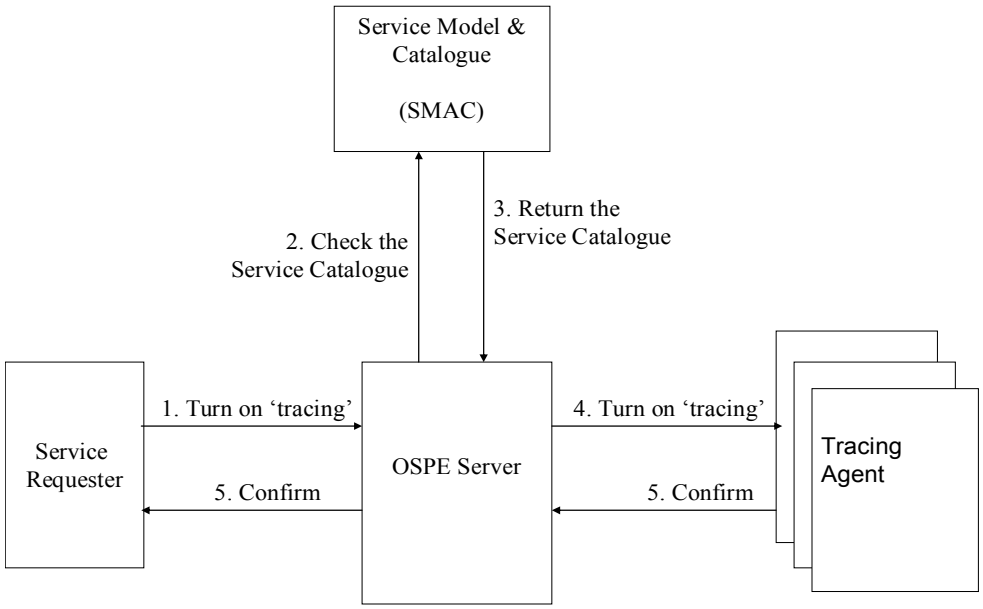

Figure 8. Turn on Service Level Tracing

When a Service Provider decides to turn on SLT for a service, the operator will send a tracing turn on request to the OSPE Server. The OSPE Server may then check the service dependency from SMAC to see which components and other resources are related with this service. After that, the OSPE Server sends the tracing turn on message separately to each Tracing Agent. Then the Tracing Agent will activate its tracing function and start to check each service request to see if it needs to start the tracing. If the tracing has been successfully activated, the confirm message is sent back to the OSPE Server.

5.4.2.2 Starting Tracing from Terminal

Figure 9 shows how the SLT is running between terminal and OSPE components.

Before starting a tracing session, the Service Provider should make sure that the tracing functions of related components are turned on.

When the Service Provider decides to start a SLT for a service, the OSPE Server will first mark the user's terminal. Then, when the user starts the service from the terminal, a trace token will be added to the message sent from the terminal to the next component. The component receives the message (including the token) and checks if it needs to start the tracing session. If the message received has been marked with the token, the component starts tracing and recording the service execution. During the tracing, the log data shall be collected. During the service execution, the related outgoing messages shall be marked by the token and thus the token will be forwarded to other components. The following component will check the token and start tracing if needed. Tracing log data will be transferred to the OSPE Server. After receiving tracing log data from Tracing Agent, the OSPE Server may correlate the log data. The log data will be finally transported to the requestor through the interface OSPE-2.

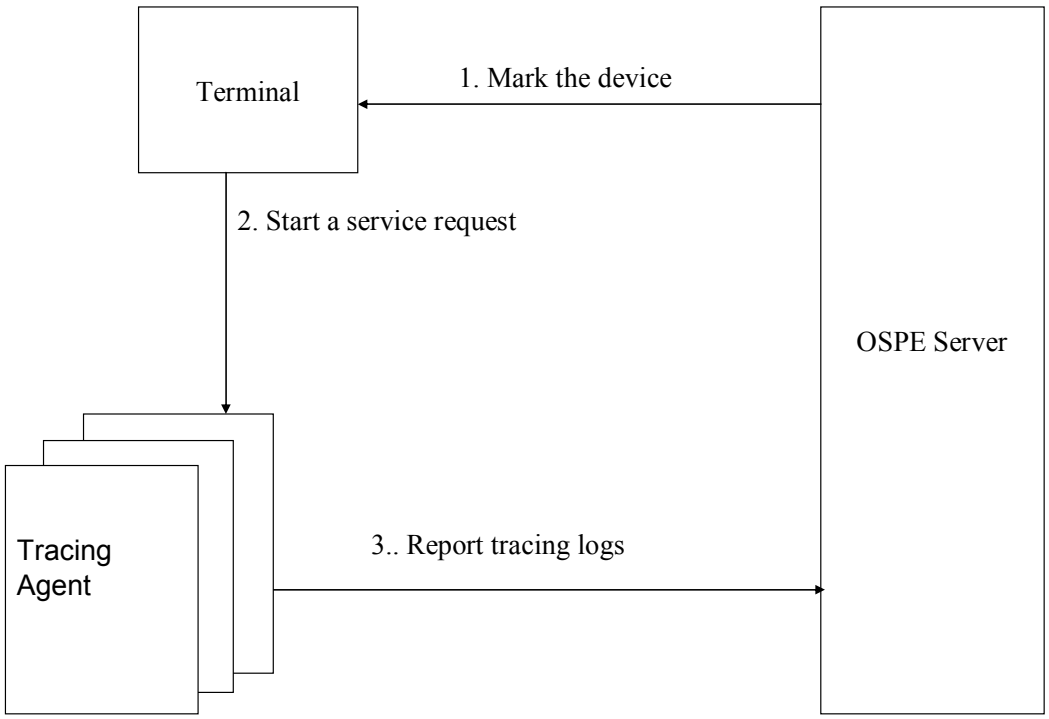

Figure 9. StartingTracing from Terminal

5.4.2.3 Starting Tracing from Component

Figure 10 shows how the SLT is running from any given component.

Before starting a tracing session, the Service Provider should make sure that the tracing functions of related components are turned on.

Sometimes the OSPE Server would like to start SLT at a given component no matter where the message comes from and no matter whether the terminal supports SLT function or not. The OSPE Server will issue a message to the Tracing Agent to mark that component. The user starts the service at the terminal. When the marked component has received a message from the terminal, a trace token will be added to all outgoing messages related to that service sequence from that component. The component starts tracing; the log data will be collected. The following components will check the token and start the tracing session if service level tracing is turned on. All of the log data will finally be transported to OSPE Server through interface OSPE-5 and to the requestor through the OSPE-2 interface.

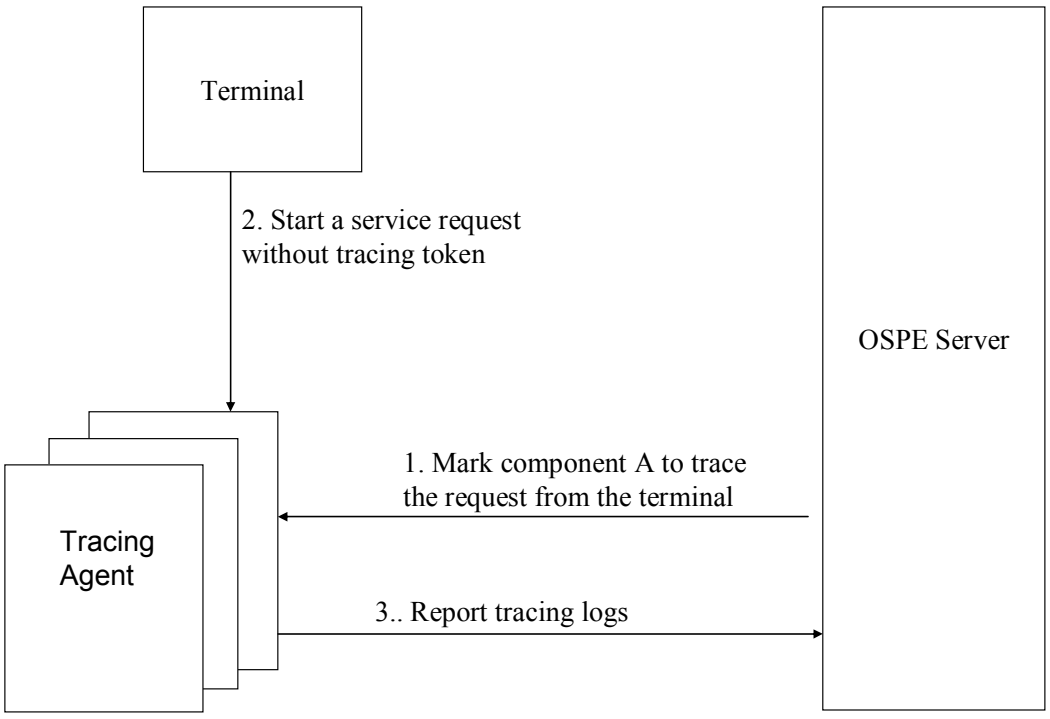

Figure 10. StartingTracing from Component

5.5 Security Considerations

Since OSPE will receive requests through its I0 interfaces and interact with OSPE agents to perform Life Cycle Management (LCM) and Service Level Tracing (SLT) activities, securing these operations is important.

Interaction with the OSPE enabler may be within the same domain or between different domains. Note that different domains may imply: different administrative domains, different security domains or security levels and/or the need to traverse insecure networks between the domains.

The OSPE requestors (e.g. OMA enabler and non-OMA resource) may reside in the same domain as the OSPE enabler and hence security measures should be considered that allow for secure intra-domain exchanges between the requestors and OSPE enabler. Alternatively the OSPE Requestors or OSPE agents may reside in a different domain from the OSPE Server domain hence security measures should be considered that allow for secure inter-domain exchanges between the requestors and OSPE enabler, and these inter-domain exchanges are thus most vulnerable to security attack.

Authentication can provide origin verification; confidentiality ensures that a third party cannot understand the contents of a message; integrity ensures that unauthorized changes to messages will be recognized. . The messages may be protected for confidentiality and integrity.

Appendix A. Change History (Informative)

A.1 Approved Version 1.0 History

Reference	Date	Description
OMA-AD-OSPE-V1_0	22 Oct 2009	Status changed to Approved by TP: OMA-TP-2009-0453-INP_OSPE_V1_0_RRP_for_Final_Approval

Appendix B. OSPE Service Meta Model (Informative)

OSPE manages resource data, Component data, Service data, Service Packages, and partner Service Provider information. The OSPE managed data is further configured to include dependencies with other resources (enablers, backend systems) that need to be configured or notified when OSPE managed data changes.

An OSPE service meta model is depicted in Figure 11; it shows the relations between elements that are used to construct end-user services: Service data, Service Package data, partner Service Provider data, Component data, and Resource data.

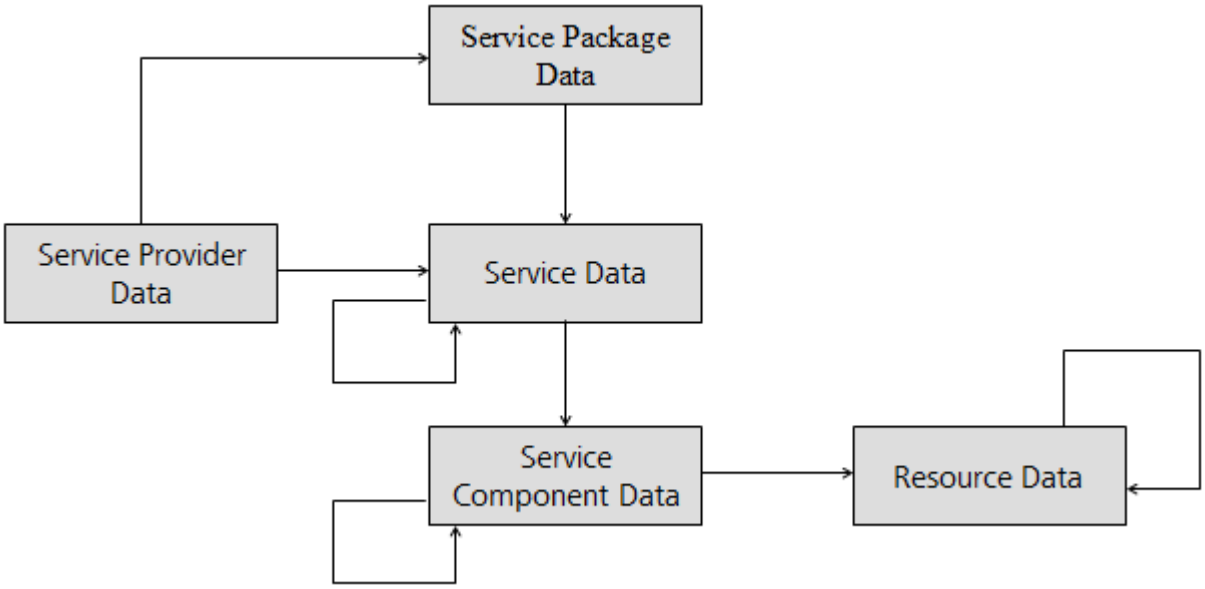

Figure 11. OSPE Service Meta Model

The relationships between data managed and/or stored by OSPE are the following:

- Relationship between Service and Service Package:
  - A Service Package consists of at least one service;
  - A Service belongs to zero or any number of Service Packages;
  - A Service may depend on other Services;
- Relationship between Service/Service Package and Service Provider:
  - A Service or Service Package is provided by one partner Service Provider;
  - An Service Provider has zero or any number of Service/Service Package;
  - A subscriber can subscribe to Services or Service Packages;
- Relationship between Service and Component:
  - A Service should use at least one Component in owner Service Provider environment;
  - A Component bears any number of Services.
  - A Component may depend on other Components;
- Relationship between Component/Service and Resource:
  - A Resource can be controlled by any number of Components
  - A Resource may depend on other Resources;

Such a generic meta model supports applications such as Service Level Tracing and Life Cycle Management, but also, for example Service Subscription Provisioning. If needed for the application, elements in the meta model will be augmented with Service Attributes. The semantics of these Service Attributes can be understood by applications, for example SLT, Service Subscription Provisioning and Life Cycle Management activities such as Provisioning.

B.1 Applying the Service Meta Model

B.1.1 PoC Service Model

Figure 12 shows an incomplete and non-exhaustive example instance of the service meta model. The purpose of this example is to demonstrate usage of the Service Model and Service Deployment Instance. We end with a brief demonstration of usage of the Service Deployment Instance by Service Subscription Provisioning.

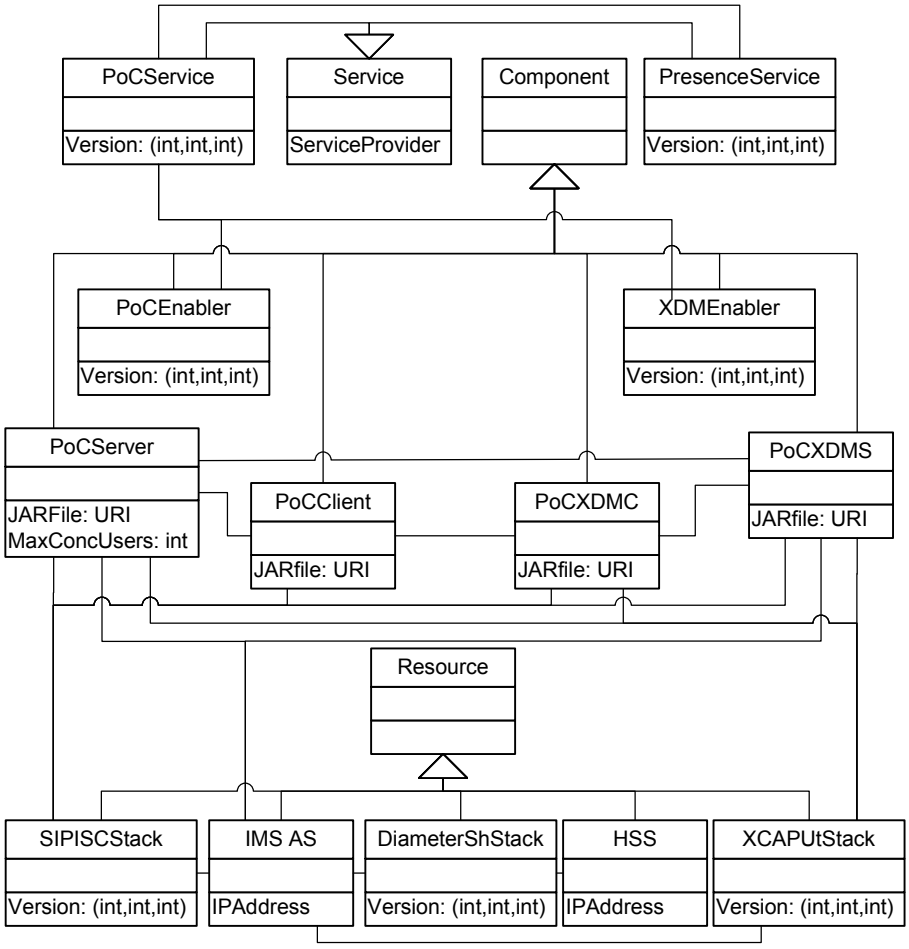

Figure 12. Example incomplete PoC Service Model

The SMAC is configured to contain Service Models of Service Provider offered services such as PoC. The PoC service is characterized by the enablers it is constituted of and by the components that realize the enabler. Attributes and types representing characteristics (such as location, version, etc) of the services, components and other resources are included in the service model. In this particular example, the components of the Presence and XDM enabler are not included for brevity. Also, in this particular example, we assume PoC is realized with IMS and the PoC Server component [PoC-AD] is hosted on a IMS Application Server (AS), which contains other resources such as realizations of ISC, Sh and Ut [IMSinOMA-AD].

For brevity, not all resources are modelled. Finally, this diagram is informational and further study of it is left as a reader's exercise.

B.1.2 PoC Service Deployment Instance

When an OSPE-3 user deploys a service, a deployment instance of a Service Model will be created. In the case of deploying PoC, this means that an instance of the PoC Service is created, its attributes will receive valid values, and pointers will be provided to existing or newly created other instances of PoC Service Model classes (such as instances of Component and/or Resources).

In addition, those services – upon which the newly deployed service depends – that are offered by partner Service Provider (in this example, the Presence Service could have been offered by a partner Service Provider) will be added to the Service Deployment Instance as a dependency.

Prior to offering services to customers (e.g. end-users), services need to be grouped under one commercial package or bundle. In Service Model terms, an instance of the Service Package is created and populated.

B.1.3 PoC Service Deployment Instance usage by Service Subscription Provisioning

When a subscriber subscribes to a Service Package, the Service Deployment Instance is traversed for execution of any appropriate actions. For example, present components and resources available on the subscriber's device must match those expected to be present according to the Service Deployment Instance. The Service Subscription Provisioning process could communicate with OMA Device Management enablers in order to realize which components and resources are available on the device. Additionally, Service Attributes and their values could indicate information that needs to be provisioned in the end user profile (e.g. traditional settings in the HLR), the need to provision some user information and preferences into a service system, the need to set charging rules for the subscriber and/or its users, the need to set policies to allow request for the user/service, etc.

B.2 Possible realization of Service Meta Model

The meta model introduced in Appendix B could be realized by TeleMagement Forum's [TMF] Shared Information/Data (SID). The SID is an object oriented information model (abstraction and representation of the elements in a managed environment) and includes definition of the elements' attributes, methods, constraints, and relationships. The SID is independent of any specific type of repository, software usage, or access protocol.

The TeleMagement Forum's SID Business Entity Framework contains eight SID Domains: Market / Sales, Product, Consumer, Service, Resource, Supplier / Partner, Enterprise, and Common Business. Logically, based on high level descriptions, the OSPE data can be mapped to SID Business Domains. Such a mapping is presented here:

- Service Provider maps to Supplier/Party
- Service Provider maps to Supplier/Party and Common Business Domains
- Service Package maps to Product, Service, and Resource Domains
- Service maps to Product, Service, Resource, and Common Business Domains
- Component maps to Product, Service, Resource, and Common Business Domains
- Resource maps to Product, Service, Resource, and Common Business Domains

Appendix C. OSPE Agents Implementation Aspects (Informative)

This Appendix describes some implementation considerations associated with OSPE Agents. As described in Section 5.3, OSPE Agents cooperate with target resources to perform the OSPE actions.

OSPE Agents can be implemented with target resources in different ways. One implementation option is that OSPE Agents are implemented in the server entities of a target resource, as shown in Figure 13. In this way, the server entity of a target resource performs OSPE provisioning actions, tracing actions, notification actions, and supports the corresponding OSPE-4, OSPE-5 and OSPE-6 interfaces.

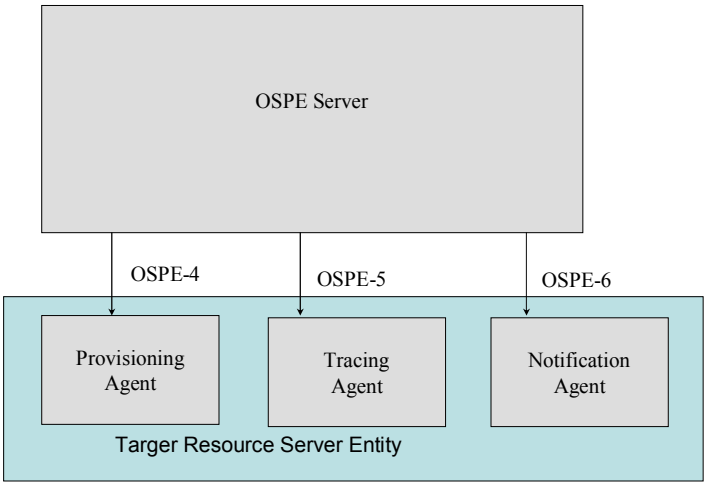

Figure 13. OSPE Agent implemented within the Server Entity of Target Resources

Another implementation option is that OSPE Agents are not tightly integrated with target resources and OSPE Agents communicate with the target resources through target resource management interfaces (out of scope). This is shown in Figure 14; the Provisioning Agent, Tracing Agent, and Notification Agent are components that receive messages through respectively the OSPE-4, OSPE-5, and OSPE-6 I0 interfaces and corresponding functions can be performed by target resource management interface invocations.

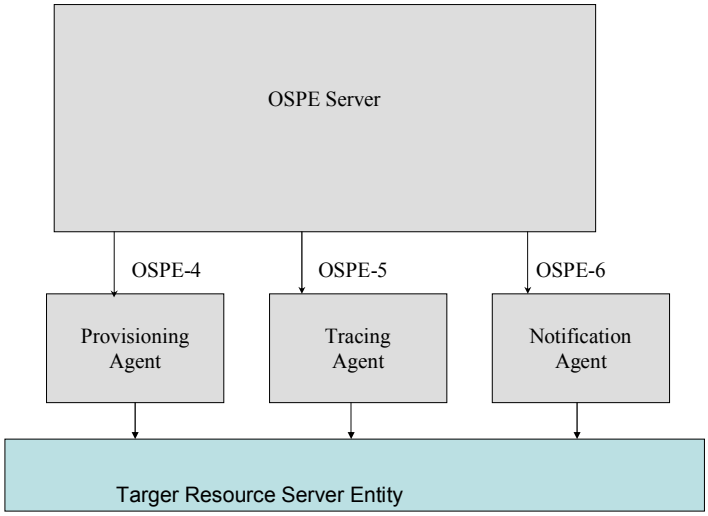

Figure 14. OSPE Agents stands alone