The OSS through Java™ API Roadmap

Version 3.1 (Public Final)

Copyright © 2001 – 2006. The Members of the OSS through Java™ Initiative.

All rights reserved. Use is subject to license terms.

OSS/J Product Team

January 2006

Editor:

Srinivasa Samudrala, Motorola Inc

Contributors:

While this document represents the combined efforts of the membership of the OSS through Java™ Initiative, the following individuals were active participants in the development of the OSS/J API Roadmap. Without their tireless efforts and invaluable contributions, this document would never have been possible.

Andreas Buschmann,	*Vodafone DE*
Andreas Ebbert	*Nokia Networks*
Dave Raymer	*Motorola Inc.*
Eric Dillon,	*4DH Consulting*
Pierre Gauthier,	*Metasolv Software Inc.*
David Milham,	*BT exact*
Michel Pedneault,	*MetaSolv Software Inc.*
Vincent Perrot,	*Sun Microsystems*
Antonio Plutino,	*Sun Microsystems*
John Reilly,	*MetaSolv Software Inc.*
John Wilmes,	*Ceon Corporation*

Copyright Acknowledgements:

The eTOM (enhanced Telecom Operations Map) is the copyrighted, intellectual property of the TeleManagement Forum (http://www.tmforum.org). This document makes heavy use of the eTOM business process taxonomy; the OSS through Java™ Initiative would like to acknowledge and thank the TeleManagement Forum for allowing the use of the eTOM material.

Revision history

Version	Date	Purpose/Changes from Previous/Status
0.1	2001	Initial roadmap as mapped to TOM
0.11	June 5, 2002	eTOM mapping
0.12	October 16, 2002	Adding sections on OSS/J roadmap and Business scenarios
0.13	November 8, 2002	Ensuring consistency of document as per John Reilly's comments
0.14	November 25, 2002	Integration of the feedback from the architecture board
0.2	January 27, 2003	Complete rewrite of roadmap to 'defocus' functional aspects and focus on API description, and distinguish between core APIs and others.
0.3	January 28, 2003	Integrated Common team review from London meeting.
0.4	February 7, 2003	Integrated Common and Business team review to prepare for final draft.
1.0	February 21, 2003	Release of the Roadmap approved by the OSS/J architecture board.
1.1	March 10, 2003	Integrated feedback from OSS/J business team.
1.2	January 24, 2004	Update to account for OSS/J work in 2003
1.3	February 13, 2004	Included feedback from Common team
1.4	March 5, 2004	Updated API mapping to eTOM v4.0
1.5	March 26, 2004	Included feedback from OSS/J Advisory Council
1.6	April 1, 2004	Included feedback from OSS/J Advisory Council and Architecture board
1.7	April 5, 2004	Version submitted for the OSS/J Architecture Board approval.
1.8	April 6, 2004	Included additional Architecture Board comments to emphasize the CBE vision within the roadmap document
2.0	April 22, 2004	Release of the Roadmap approved by the OSS/J Architecture Board and the OSS/J Program Management team
3.0	March 2005	New release, many formatting changes and section of "areas for future exploration" removed; comments from OSS/J member companies incorporated.
3.0.1	March 2005	Final Public Release
3.0.2	November 2005	Draft Release for review by Product Team
3.0.3	December 2005	Updated after comments from Product Team
3.0.4	January 2006	Minor updates after comments from Product Team
3.1	January 2006	Public Final Release

List of Acronyms

ACK	Acknowledgement
API	Application Programming Interface
B2B	Business to Business
CBE	Common Business Entities
CRM	Customer Relationship Management
DSL	Digital Subscriber Line
EMS	Element Management System
eTOM	enhanced Telecom Operations Map™
GUI	Graphical User Interface
IDE	Integrated Development Environment
IP	Internet Protocol
ITU	International Telecommunication Union
J2EE	Java™ 2 Platform, Enterprise Edition
J2SE	Java™ 2 Platform, Standard Edition
JCP	Java Community Process℠
JSR	Java Specification Request
MIB	Management Information Base
NE	Network Element
NMS	Network Management System
OSS	Operation Support System
OSS/J	OSS through Java™ Initiative
PM	Performance Management
QOS	Quality of Service
RM&O	Resource Management and Operations
SID	Shared Information/Data Model
SLA	Service Level Agreement
SM&O	Service Management and Operations
S/PRM	Supplier/Partner Relationship Management
TMN	Telecommunication Management Network
VPN	Virtual Private Network

Table of Contents

1 OVERVIEW 9

1.1 Mapping to Standards 12

1.2 Information Model Extensions 13

2 OSS/J ROADMAP 16

2.1 Architectural Grouping of Interfaces 16

2.2 Roadmap 18

3 CORE APIS 22

3.1 Customer Management 22

3.2 Order Management 24

3.3 Service Activation 26

3.4 Testing 27

3.5 Product Inventory 29

3.6 Service Inventory 30

3.7 Resource Inventory 33

3.8 Service Discovery 35

3.9 Resource Discovery 36

3.10 Trouble Ticketing 37

3.11 Service Problem Resolution 39

3.12 Customer SLA Management 41

3.13 Process Quality Management 43

3.14 Service Quality Management 45

3.15 Fault Monitoring 47

3.16 Performance Monitoring 49

3.17 Billing 51

3.18 Billing Mediation 53

3.19 Product and Offer Pricing 54

4 BUSINESS SCENARIOS 56

4.1 DSL provisioning 56

4.2 Generic DSL Activation 57

4.3 Fault Management 59

4.4 Performance Management 61

4.5 Bill Generation Management 62

4.6 Inventory Management 63

Figures

Figure 1: Mapping of OSS through Java™ APIs to TeleManagement Forum™ eTOM 13
Figure 2: OSS through Java™ APIs and supporting models..... 14
Figure 3: Use of TeleManagement Forum™ SID components by OSS through Java™ APIs..... 15
Figure 4: Core API Priorities 19
Figure 5: Core API Dependencies 20
Figure 6: Customer Management Interactions..... 23
Figure 7: Order Management Interactions 25
Figure 8: Service Activation Interactions..... 27
Figure 9: Testing Interactions 28
Figure 10: Product Inventory Interactions 30
Figure 11: Service Inventory Interactions 32
Figure 12: Resource Inventory Interactions 34
Figure 13: Service Discovery Interactions 35
Figure 14: Resource Discovery Interactions 36
Figure 15: Trouble Ticketing Interactions..... 38
Figure 16: Service Problem Resolution Interactions..... 40
Figure 17: Customer SLA Interactions..... 42
Figure 18: Process Quality Management Interactions 44
Figure 19: Service Quality Management Interactions 46
Figure 20: Fault Monitoring Actions 48
Figure 21: Performance Monitoring Interactions..... 50
Figure 22: Billing Interactions..... 52
Figure 23: Billing Mediating Interactions 53
Figure 24: Pricing API Interactions 55
Figure 25: Generic DSL Activation..... 57
Figure 26: Generic DSL Cancellation 58
Figure 27: Fault Resolution Scenario with Manual Repair..... 59
Figure 28: Fault Resolution Scenario with Automated Network Correction..... 59
Figure 29: Automated Fault Detection and Resolution Scenario..... 60
Figure 30: Performance Management Scenario 61
Figure 31: Bill Generation Scenario 62
Figure 32: Inventory Management Scenario..... 63

Tables

Table 1: OSS through Java™ list of core APIs 12
Table 2: OSS/J API coverage 18
Table 3: JSR priorities 21

1 Overview

Current OSS technology cannot cope with the rapidly increasing scale of networks, the diversity of communications technology, shortened time to market for new services, and heightened expectations for availability and reliability. In short, service providers need carrier-grade OSS solutions in Internet time.

To meet this increase in demand, service providers need a new approach to providing operations support system (OSS) solutions. The OSS through Java™ Initiative defines a set of APIs, with client access either by tightly or loosely coupled mechanisms, to foster an OSS component market. The goals of the initiative are to develop, through the Java Community Process℠ (JCP) program, API specifications, reference implementations, and technology compatibility kits, for OSS integration and deployment.

The OSS/J APIs make up a full OSS solution that supports flow-through Service Fulfillment, Assurance, and Billing. This document is a roadmap of the APIs that are being defined, the APIs that still need to be defined, and how they fit together. Further more, the presentation of the OSS/J APIs is done in the context of the TeleManagement Forum eTOM. It is not the goal of this document to supplant any aspect of the eTOM suite. The goal of this document is to describe the OSS/J APIs and show, using a service oriented approach, where those APIs fit in the business of a typical information and/or communication service provider.

Table 1 shows the list of core APIs, and their summarized descriptions. Detailed descriptions are provided in section 3. Figure 1 is providing their mapping to the TeleManagement Forum™ business process framework, called eTOM as described in section 1.1 and Figure 3 provides a view of which component in the TeleManagement Forum™ Shared Information/Data Model, called SID, is used by these APIs.

Section 2 describes the OSS/J roadmap with the list of intended JSRs and the APIs they cover. It also provides a status of the advancement of work and the next priorities. Finally it provides an overall picture of where the initiative is going by showing the API dependencies.

Section 3 details each core API, and how they interact with each other. Each API identified in this section is defined in terms of a general description, how the API interacts with other APIs, and how it maps to comparable standards.

API[1]	Summarized description
Customer Management	The Customer Management API provides interfaces, as specified by the OSS/J Design Guideline[2], for creating, modifying, suspending, and terminating customers.
Order Management	The Order Management API provides interfaces, as specified by the OSS/J Design Guideline, for creating, modifying, suspending, and canceling orders and order activities.
Product[3] Activation	Subsumed by Order Management API, JSR 264
Service Activation	The Service Activation API provides interfaces, as specified by the OSS/J Design Guideline, to activate services as defined by TeleManagement Forum™ eTOM and SID.
Resource Activation	Subsumed by Order Management API, JSR 264
Testing	The Testing API provides interfaces, as specified by the OSS/J Design Guideline, for creating, modifying, suspending, and canceling testing jobs, as well as querying test job state and results.
Product Inventory	The Product Inventory API provides interfaces, as specified by the OSS/J Design Guideline, for populating, querying and updating the Product Inventory repository.
Service Inventory	The Service Inventory API provides interfaces, as specified by the OSS/J Design Guideline, for populating, querying and updating the Service Inventory repository.
Resource Inventory	The Resource Inventory API provides interfaces, as specified by the OSS/J Design Guideline, for populating, querying and updating the Resource Inventory repository.

[1] Information on whether the API is available, under active development or in the plans is provided in Section 2.
[2] This document provides all the design principles that all API's implemented under the OSS through Java™ Initiative's umbrella will abide to. It is available on the OSS/J web site.
[3] Note that the terms of Product, Service and Resource are used in this document in alignment with the TM Forum Shared Information/Data Model use and definition.

Service Discovery	The Service Discovery API provides interfaces, as specified by the OSS/J Design Guideline, for managing the data collection process. The Service Discovery API provides access to agents that discover and update services and service topology information that populates the Service Inventory.
Resource Discovery	The Resource Discovery API provides interfaces, as specified by the OSS/J Design Guideline, for managing the data collection process. The Resource Discovery API provides access to agents that discover and update the physical resources and topology information that populates the Resource Inventory.
Trouble Ticketing	The Trouble Ticketing API provides interfaces, as specified by the OSS/J Design Guideline, for creating, tracking, and deleting trouble tickets.
Service Problem Resolution	The Service Problem Resolution API provides interfaces, as specified by the OSS/J Design Guideline, for querying, creating, executing, or deleting actions, be notified of these events, and configuring rule-based decision-making engines.
Customer SLA Management	The Customer SLA Management API provides interfaces, as specified by the OSS/J Design Guideline, for creating, tracking, and deleting SLAs.
Process Quality Management	The Process Quality Management API provides interfaces, as specified by the OSS/J Design Guideline, for creating, updating and deleting process affecting metric objects.
Service Quality Management	The Service Quality Management API provides interfaces, as specified by the OSS/J Design Guideline, for querying, creating, updating and deleting Service Level Specifications objects, Service Quality Objective objects, and Service Quality Report objects, as well as subscribing for notifications on object violation events and availability of new service quality reports.

Fault Monitoring	The Fault Monitoring API provides interfaces, as specified by the OSS/J Design Guideline, that allow clients to collect and acknowledge alarms. The API enables reception of alarms, state changes, and threshold crossing alerts from the network and maintaining a list of active alarms. It also ensures availability of the most current view of the alarm state of the network.
Performance Monitoring	The Performance Monitoring API provides interfaces, as specified by the OSS/J Design Guideline, for creating and deleting metric and threshold objects. The API supports the collection of performance data from the network, setting thresholds, and generating/forwarding threshold crossing events.
Billing	The Billing API provides interfaces, as specified by the OSS/J Design Guideline, for creating and deleting billing objects. The API enables for rating services and calculating billing records, sending invoices to customers, processing their payments, and performing payment collections. Also, the Billing API handles inquiries by the customer about bills, and handles billing problems.
Billing Mediation	The Billing Mediation API provides interfaces, as specified by the OSS/J Design Guideline, for creating and deleting mediation jobs. This API enables for matching usage to individual services for usage-based services.
Pricing	The Pricing API will provide a central point to consult and determine offers and prices based on a variety of criteria including a specific customer profile, location, current promotions, other parties in the transactions, factors peculiar to the request, and any other set of complex, possibly inter-related points.

Table 1: OSS through Java™ list of core APIs

1.1 Mapping to Standards

Figure 1 identifies how the OSS through Java™ core APIs map to the TeleManagement Forum™ eTOM (enhanced Telecom Operations Map™) processes, version 4.0 of the document.

The main difference in scope between the OSS through Java™ roadmap of APIs and the eTOM is that the roadmap defines the APIs that are essential to automatic, flow-through service management. In essence the APIs cover a subset of the eTOM, which identifies general processes regardless of whether they are automatic or must be accomplished by humans. There is also currently a limited support on the Service Problem Resolution and Supplier/Partner Relationship Management as these areas are currently considered out of scope of this roadmap.

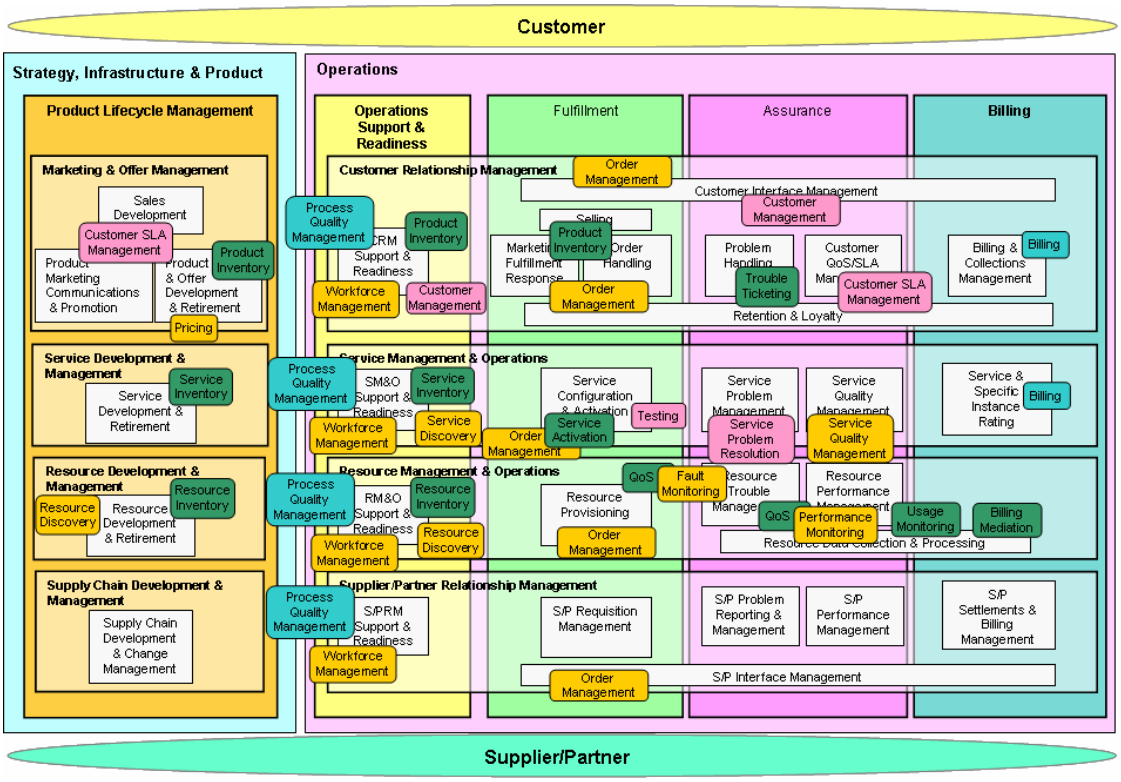

Figure 1: Mapping of OSS through Java™ APIs to TeleManagement Forum™ eTOM[4]

Care has also been taken in section 3 to map each API to a function set in the ITU M.3400 standard, TMN Management Functions and other appropriate standards.

1.2 Information Model Extensions

The OSS through Java™ Initiative is improving the efficiency of API developers and maintaining consistency across APIs by defining, modeling and implementing core concepts, such as product, service, resource, customer, and so forth. These concepts form

[4] Note that the eTOM background picture in Figure 1 is partial and provides only a view of the relevant parts of the framework from an OSS/J perspective. The eTOM background is an extract of the TeleManagement Forum™ GB921 version 4.0 document. APIs indicated in red are new APIs recently initiated.

the basis of the Core Business Entities (CBE) model and associated Java packages[5]. This work is based on part the TeleManagement Forum™'s New Generation OSS (NGOSS) Shared Information/Data (SID) Model and represents an implementation view of the NGOSS architecture.

As a core model, it is natural that there should be a strong relationship between the CBE model and the OSS/J Common API. For example, as a new JSR/API is developed the CBE model will be extended to include the business entities within the scope of the new API that are also shared by several APIs and then the business entities, and their interfaces will be passed on for inclusion in the Common API. Similarly, XML schema representations of the CBE will be included in the Common API. The additional advantage of that approach, as illustrated by Figure 2, is that specific OSS/J APIs rely on the existence of Core Business Entities. These can be extended to account for vendor specific or technology specific information without breaking the API functionality. This design pattern to enable support of data extension is currently being documented in a white paper[6] and OSS/J APIs are progressively adapted to accommodate it.

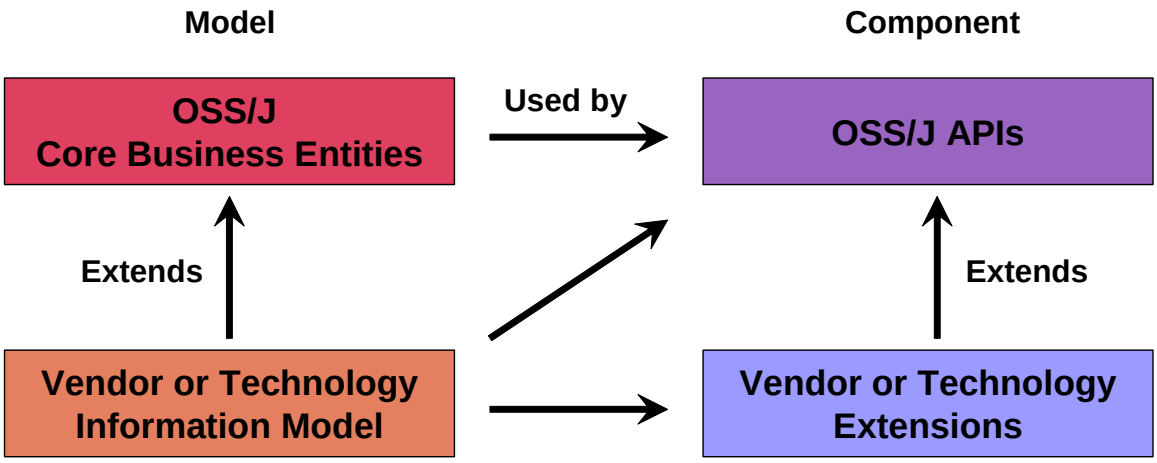

Figure 2: OSS through Java™ APIs and supporting models

The CBE model will adopt SID entities as needed by OSS/J APIs. The model will at all times strive to be consistent with the SID model. In some cases a subset of the SID model will be used; at other times the SID model may be extended as needed. As part of the collaboration between the OSS through Java™ Initiative and the TeleManagement Forum™, CBE entities not present in the SID model will be contributed to the TeleManagement Forum™ SID model. The adoption of the SID into the OSS/J APIs will

[5] For additional information on the topic please refer to the Core Business Entities Model White Paper published by the OSS through Java™ Initiative.

[6] The Core Business Entities Model White Paper (COM-ARCH_Model.1.0.doc) is currently available on the OSS through Java™ website.

occur over a period of time. In the interim some of the current API information models will be mapped to the CBE/SID model[7].

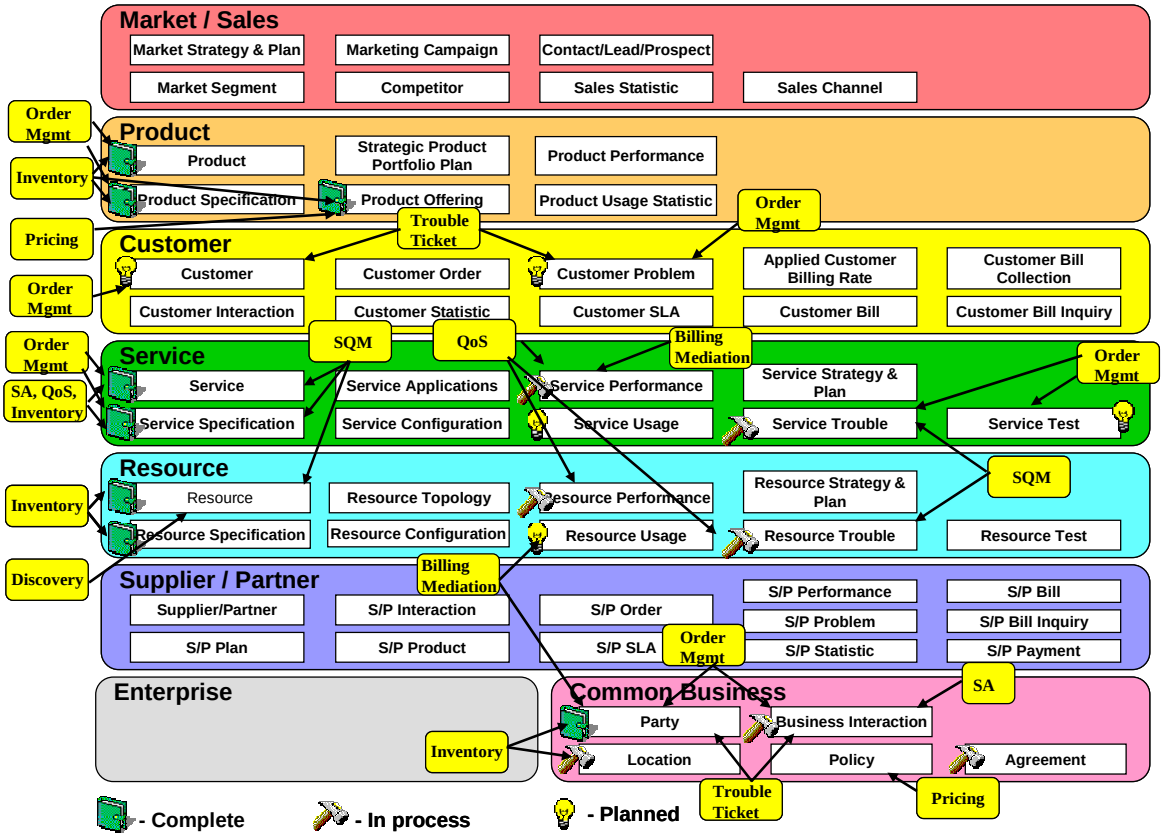

Figure 3: Use of TeleManagement Forum™ SID components by OSS through Java™ APIs

Figure 3 depicts the SID model framework and a mapping of current OSS/J APIs information requirements to the framework.

[7] Please refer to the Common API and SID Adoption Strategies White Paper.

2 OSS/J Roadmap

This section provides a roadmap of the APIs introduced in section 1. The actual detailed description of the APIs is provided in later sections. The roadmap is currently limited to the core APIs as described in section 3.

2.1 Architectural Grouping of Interfaces

The proposed groupings in the Table 2 are an optimization of the number of APIs taking into account design similarities as well as reasonable scope of work. Each grouping represents a committed or potential Java Specification Request (JSR) to be executed through the Java Community Process[SM]. In the comments column Table 2 adds references to JSRs completed and underway as well as the interface names to be used in diagrams in the sections 3.

JSR	API	Comments
Order Management	Order Management Product Activation Service Activation Resource Activation	JSR 89, this JSR is currently available although it does not yet fully cover the intended scope of order management and resource activation. JSR 264 has been initiated to extend the functionality defined by JSR 89 to encompass Order Management and provide support for use in a resource activation scenario. Interface code: OM&A
QOS	Fault Monitoring Performance	JSR 90 Interface code: QOS

	Monitoring Usage Monitoring[8]	QoS would be discontinued and be replaced by Fault Management and Performance Management APIs in 2006.
Fault Management	Fault Monitoring	JSR 263 This JSR is undertaken to enable the separation of the FM functionality from the QoS API to allow the implementation of FM without the implementation of PM, Interface code: FM
Trouble Ticket	Trouble Ticket	JSR 91 Interface code: TT
Billing Mediation	Billing Mediation	JSR 130. Note that this JSR covers some of the functionality defined in Billing (section 3.17) Interface code: Billing Mediation
Inventory	Product Inventory Service Inventory Resource Inventory	JSR 142. Note that compliance still takes into account the business views of the inventories. Interface code: Inv

[8] Usage Monitoring is not implemented as an identifiable interface within the QoS API Specification, rather is it supported through the existing Fault and Performance Monitoring functionality specified by the QoS API

Service Quality Management	Service Quality Management	JSR 210 Interface code: SQM
Pricing		JSR 251 Interface Code: Pricing
Customer SLA Management	Customer SLA Management	Interface code: SLA
Service Problem Resolution	Service Problem Resolution	Interface code: SPR
Testing	Testing	Interface code: Testing
Discovery	Service Discovery Resource Discovery	JSR 254 Interface code: Discovery
Customer Management	Customer Management	Interface code: CRM
Billing & Invoicing	Billing	Interface code: Billing
Process Quality Management	Process Quality Management	Interface code: ProcMetrics

Table 2: OSS/J API coverage

2.2 Roadmap

2.2.1 Core API Priorities and Dependencies

Figure 4: Core API Priorities provides an overview of the OSS/J roadmap of APIs as defined through their corresponding JSRs and within a Fulfillment, Assurance and Billing context[9]. It enables to understand the current scope of work as well as the future one according to the color legend. The APIs whose corresponding JSR have been approved with final release or are in maintenance mode are highlighted in green. The ones highlighted in orange have their corresponding JSR approved by the Java Community and their specification as well as reference implementation are being developed through

[9] Fulfillment, Assurance and Billing refers to the (enhanced) Telecom Operations Map as developed by the TeleManagement Forum™.

the Java Community Process[SM]. The ones highlighted in pink have been identified by the OSS/J architecture board and advisory[10] council as the next priority to tackle. Finally the APIs highlighted in blue have been identified as future work that will be required to offer a complete API coverage for the OSS domain.

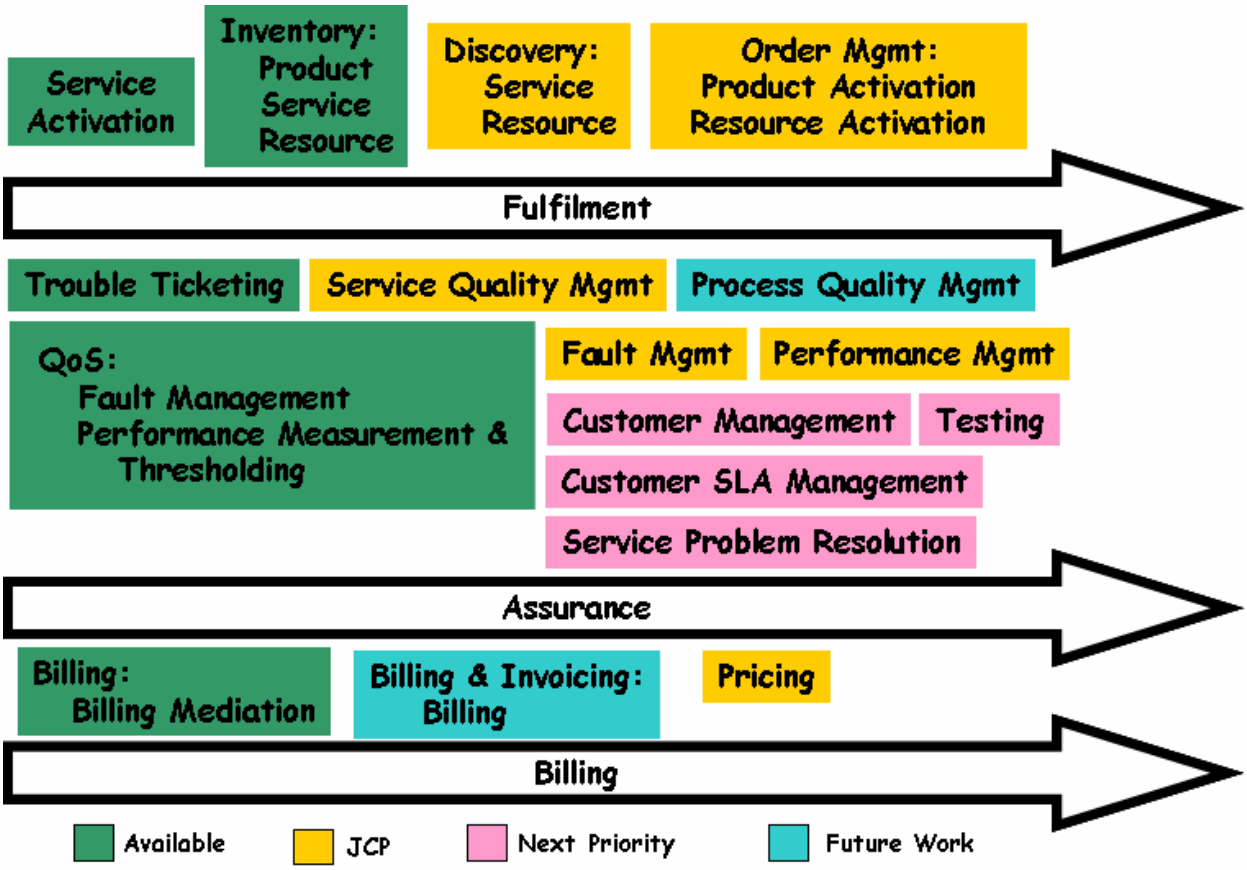

Figure 4: Core API Priorities

Figure 4: Core API Priorities, in conjunction with the coverage it represents as illustrated in Figure 1, also conveys the value of bringing together this consistent set of APIs to ensure smooth integration in the OSS/BSS market. This last aspect is further reinforced by Figure 5 that shows the dependencies between the API's reusing the same color scheme as in Figure 4: Core API Priorities.

[10] The OSS/J Advisory Council gathers service providers whose mission is to accelerate the adoption in the industry of OSS/J APIs by helping to shape the strategic direction of the OSS/J Initiative, sharing business and operational experiences, and aligning standards approaches.

2.2.2 Core API priorities

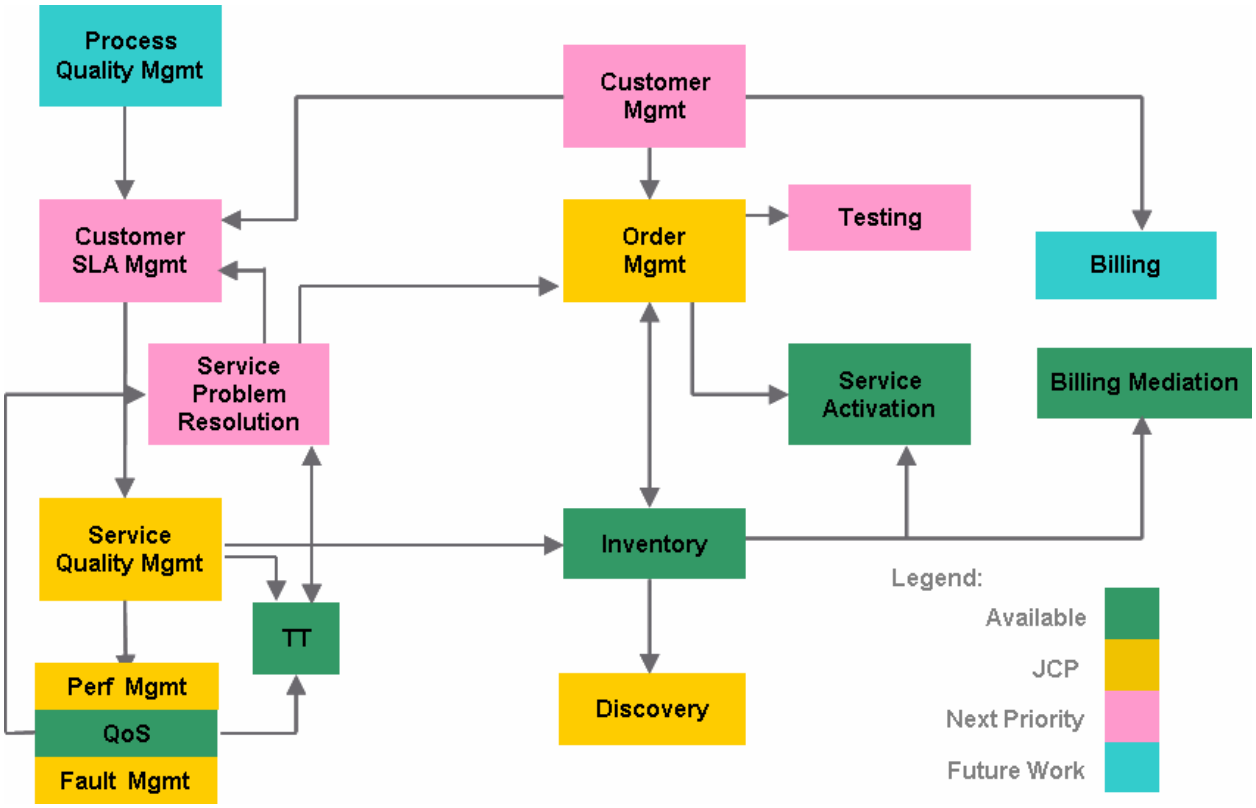

Figure 5: Core API Dependencies

The OSS through Java™ Initiative Architecture Board has highlighted priorities in its target API program taking into consideration the priorities of the service providers composing the OSS/J Advisory Council. These priorities are shown in Table 3. They are there to ensure a consistent building of the overall set of APIs according to a balance between the internal strategic plan that core OSS/J members have outlined and the needs of the service providers advising the initiative. These priorities may be altered at any time due to market demand or opportunity.

Completed or publicly available	QOS, Trouble Ticket, Billing Mediation, Inventory, Service Activation, Common
Specifications being built through the Java Community Process[SM]	Service Quality Management, Discovery (Resource), Pricing, Fault Management, Performance Management, Order Management
Next priority[11]	Customer Management, Customer SLA Management (highest priorities), Testing, Service Problem Resolution
Future work	Billing & Invoicing, Process Quality Management

Table 3: JSR priorities

[11] These are the priorities for the short term future. They are of course dependent on having willing sponsors to build the APIs.

3 Core APIs

3.1 Customer Management

3.1.1 Description

The Customer Management API provides interfaces, as specified by the OSS/J Design Guideline, for creating, modifying, suspending, and terminating customers. It is also used to manage customer information such as address, contact information, account structure and billing information.

The API provides interfaces that allow clients to:

- create, modify, suspend, and terminate Customers
- maintain customer information
- query customer information
- create, modify, and delete accounts and associated hierarchies
- query account information and hierarchies
- maintain account information and hierarchies

3.1.2 Mapping to Standards

TMF eTOM®: Customer Relationship Management {eTOM.pid = OPS-1}

ITU-T M.3400: Customer Identification function set – supports interactions with the customer to determine customer's name, address, and any distinguishing characteristics about the customer such as current services, line of business and past usage and supports updates of customer database as appropriate.

3.1.3 Interactions

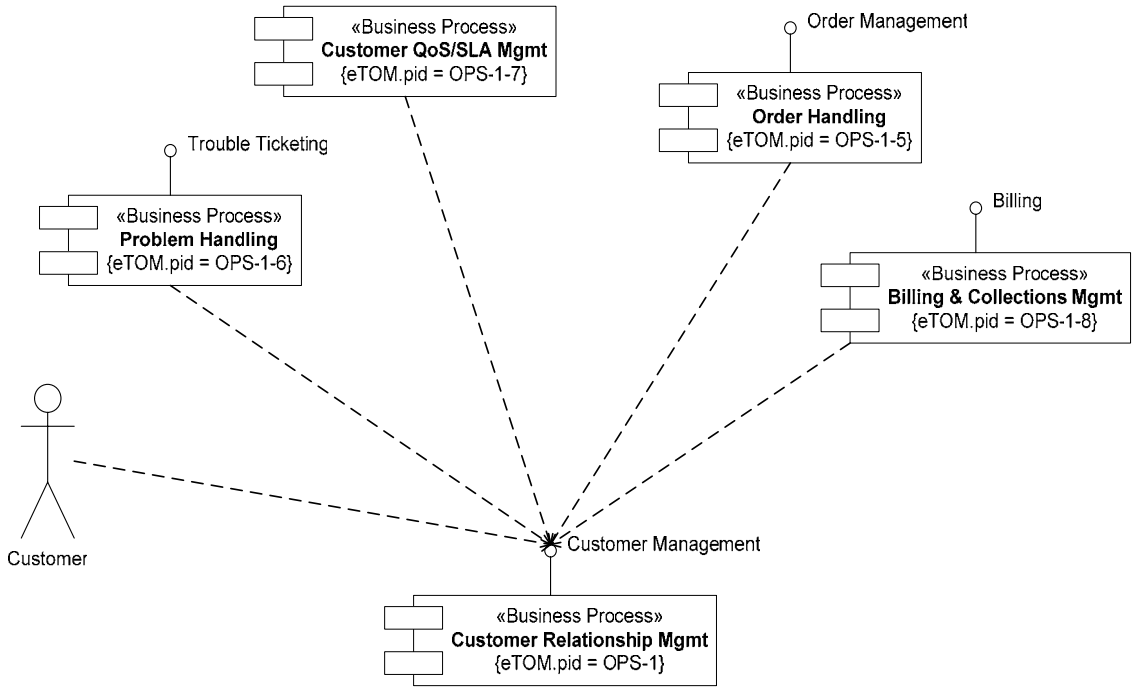

Figure 6: Customer Management Interactions

3.2 Order Management

3.2.1 Description

The Order Management API provides interfaces, as specified by the OSS/J Design Guidelines, for creating, modifying, suspending, and canceling orders. This API is used to create and track orders for a product, a service or a resource, and is used to capture the customer-selected service details. Information that is captured as part of an order may include customer account information, product offering and quality of service details, SLA details, access information, and scheduling information.

The Order Management API provides interfaces that allow clients to:

- create, modify, suspend, and cancel orders and order activities
- query orders and activities and their states
- schedule orders activities
- create, modify, and delete bulk orders and order activities
- be informed of the progress of an order and its order activities

The Order Management API can be used by customers for self-service, e.g., via the web/portal, by sales agents sitting at a GUI taking orders over the phone, or by other Service Providers in an B2B transaction.

3.2.2 Mapping to Standards

TMF eTOM®: Order Handling {eTOM.pid = OPS-1-5}

ITU-T M.3400: Service Status Administration function set

3.2.3 Interactions

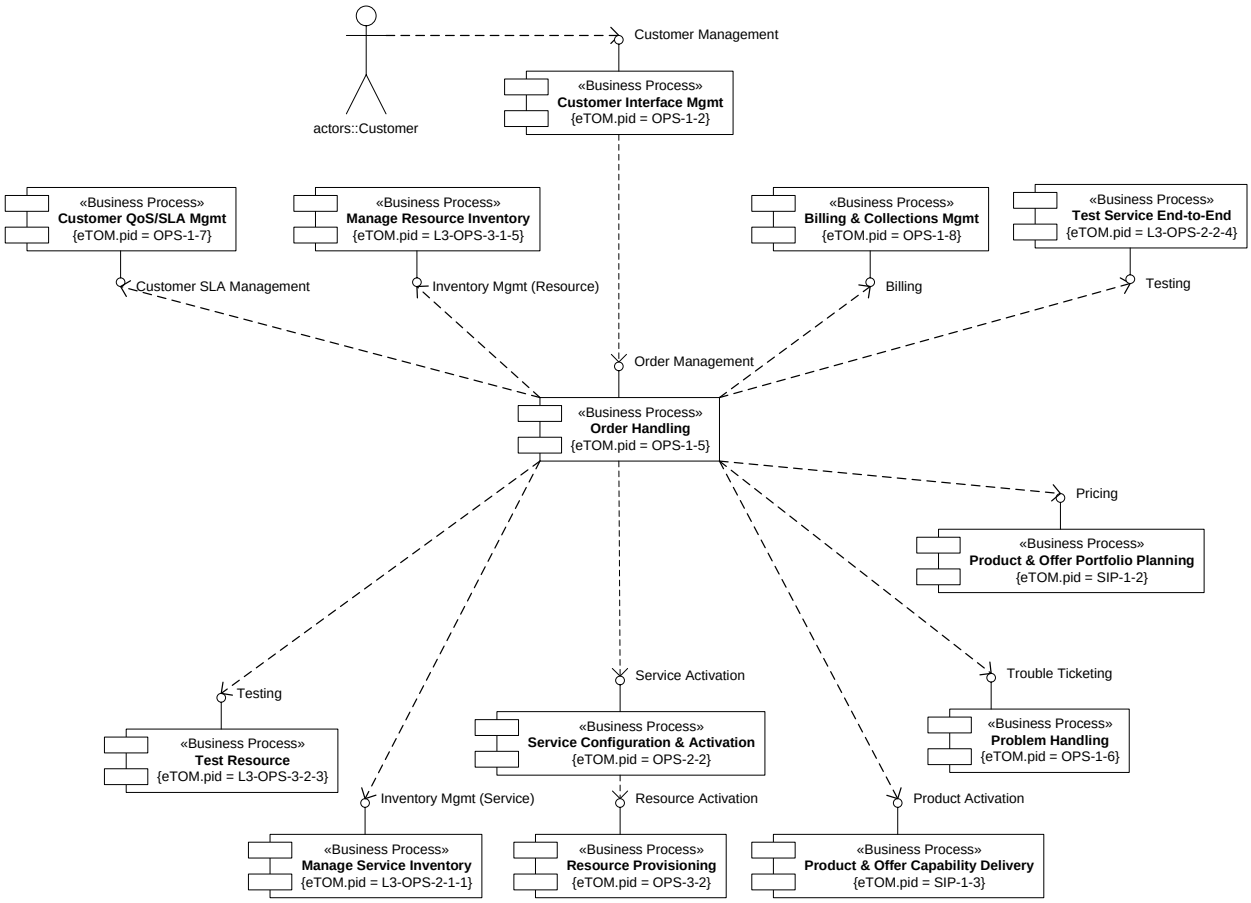

Figure 7: Order Management Interactions

3.2.4 References

Note that a limited implementation of the order management API already exists, see the OSS/J JSR 89 “Service Activation API”, v1.0, April 2, 2002.

3.3 Service Activation

3.3.1 Description

The Service Activation API[12] provides interfaces, as specified by the OSS/J Design Guideline, for activating services.

The Service Activation API provides interfaces that allow clients to:

- activate services

3.3.2 Mapping to Standards

TMF eTOM®: Service Configuration & Activation {eTOM.pid = OPS-2-2}

ITU-T M.3400: Request for Service Function Set

[12] More accurately the CBE design pattern will enable the implementation of the Service Activation API as the OSS/J Activation API applied to the CBE Service model. This activity is in progress.

3.3.3 Interactions

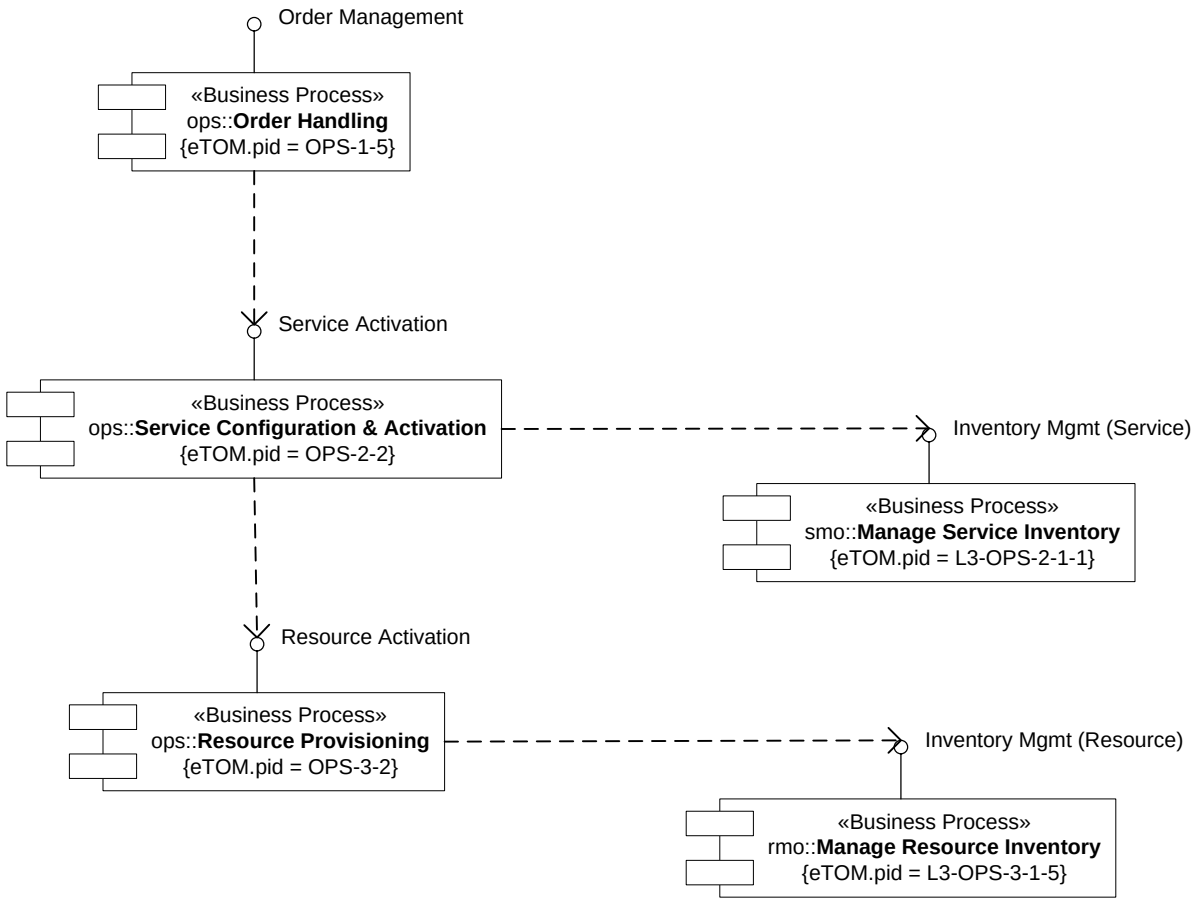

Figure 8: Service Activation Interactions

3.3.4 References

For details on current implementation stage of the Service Activation API, see the OSS/J JSR 89 "Service Activation API", v1.0, April 2, 2002.

3.4 Testing

3.4.1 Description

The Testing API provides interfaces, as specified by the OSS/J Design Guideline, for creating, modifying, suspending, and canceling testing jobs. The testing API may return raw test results or else apply some algorithmic computations to the raw test results to produce a more meaningful result. From the perspective of the TMF eTOM

The Testing Management API provides interfaces that allow clients to:

- create, modify, suspend, and cancel testing jobs
- query testing jobs and their state
- schedule tests
- be informed of the progress of a test
- query the type of tests available
- query the results of a test (by circuit number or other identifier)
- retrieve stored test results

3.4.2 Mapping to Standards

TMF eTOM®: Test Service End-to-End {eTOM.pid = L3-OPS-2-2-4},
Test Resource {eTOM.pid = L3-OPS-3-2-3}

ITU-T M.3400: Testing Function Set Group

3.4.3 Interactions

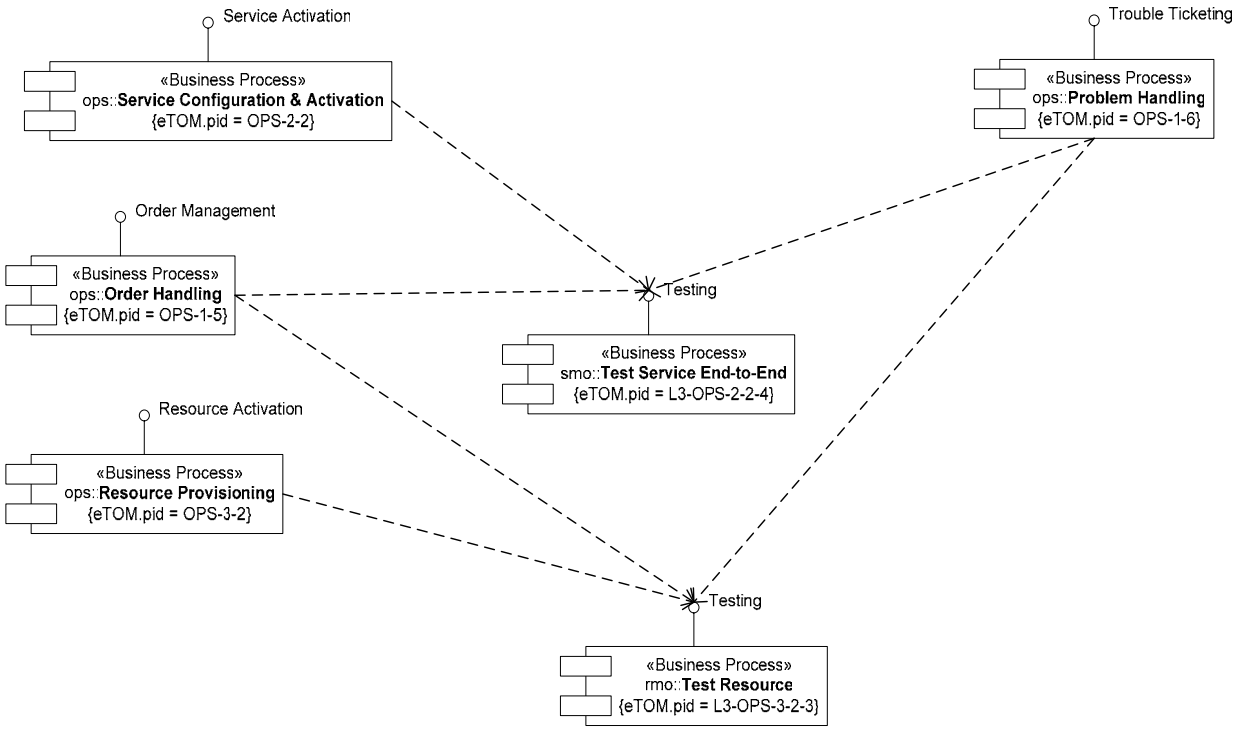

Figure 9: Testing Interactions

3.5 Product Inventory

3.5.1 Description

Product Inventory is implemented as the OSS/J Inventory API applied on the CBE Product model. It provides interfaces, as specified by the OSS/J Design Guideline, for populating, querying and updating the Product Inventory repository. Mainly the Inventory API, applied on the CBE Product model, enables clients to manage the Product Catalog and keep track of the product subscriptions. The Product Catalog defines the product offering from marketing perspective and consists of collections of product specifications. Each product specification describes a product. Several product specifications may be defined for the same product type. For example the Product Catalog may define Gold, Silver and Bronze VPN products. Product specifications are associated with service specifications, stored in the Service Catalog, thus capturing the relationship between a product and the set of services bundled by this product.

Each subscription is captured in the Product Inventory through a product instance associated with the corresponding specification in the catalog. The product instance is also associated with the Subscriber (via Party Role) of the product and the related Subscriber (Party) account information.

The Inventory API provides persistency service for shared CBE entities and, applied on the CBE Product model; it defines interfaces that allow clients to:

- create/remove product entities, specifications and associations
- update product entities, specifications and associations
- query product entities, specifications and associations
- manage a catalog of product specifications

3.5.2 Mapping to Standards

TMF eTOM®: Product & Offer Development & Retirement (Product Specification) and Order Handling (Product)

ITU-T M.3400: Customer account administration function set
Customer profile administration function set

3.5.3 Interactions

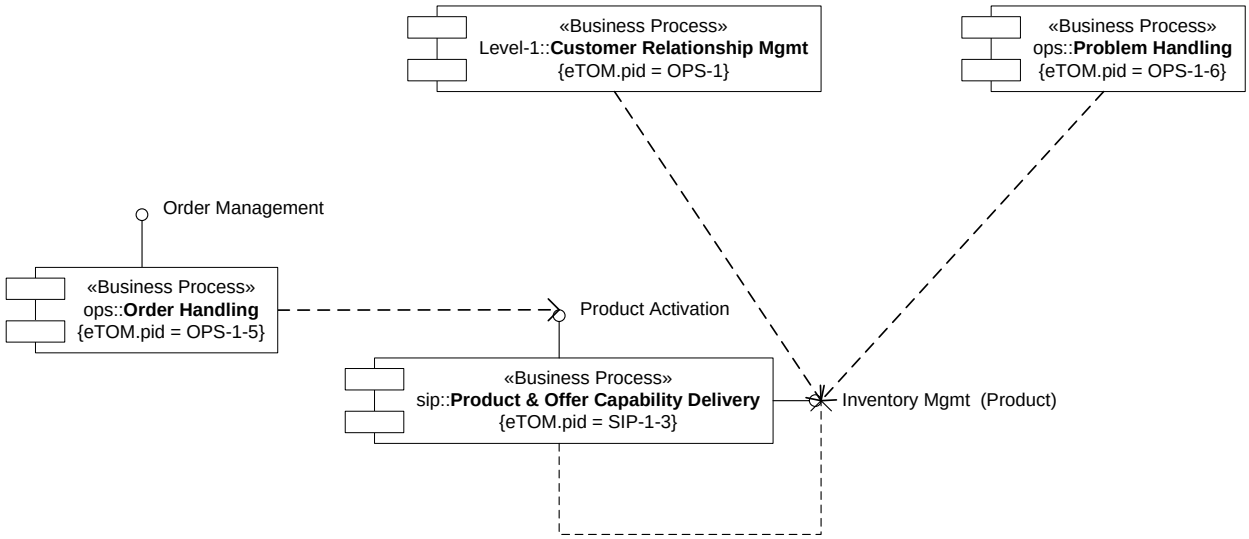

Figure 10: Product Inventory Interactions

3.5.4 References

For details on the Inventory API interfaces, see "JSR 142: OSS Inventory API".

3.6 Service Inventory

3.6.1 Description

Service Inventory is implemented as the OSS/J Inventory API applied on the CBE Service model. It provides interfaces, as specified by the OSS/J Design Guideline, for populating, querying and updating the Service Inventory repository. The Service Inventory is populated by Service Planning and Provisioning Control. It is also updated trough reconciliation using Service Discovery. Mainly the Inventory API, applied on the CBE Service model, enables clients to manage the Service Catalog and keep track of the planned, subscribed and provisioned services. The Service Catalog captures the engineering view of the Service Provider's offering and consists of collections of service specifications. Service specifications define the schema for services from an engineering perspective. Several service specifications may be defined for the same service type. For example the Service Catalog may define Gold, Silver and Bronze VPN services. Service specifications are associated with resource specifications, stored in the Resource Catalog, thus capturing the relationship between a service and the set of resources supporting this service.

Each subscription is captured in the Service Inventory through a set of service instances associated with the corresponding specifications in the Service Catalog, the subscribed product, bundling these services and the users (recipients) for these services.

The Inventory API provides persistency service for shared CBE entities and, applied on the CBE Service model, it provides operations for:

- creating/removing service entities, specifications and associations
- updating service entities, specifications and associations
- querying service entities, specifications and associations
- managing a catalog of service specifications

3.6.2 Mapping to Standards

TMF eTOM®: Manage Service Inventory {eTOM.pid = L3-OPS-2-1-1}

ITU-T M.3400: Customer profile administration function set
Circuit inventory query function set

3.6.3 Interactions

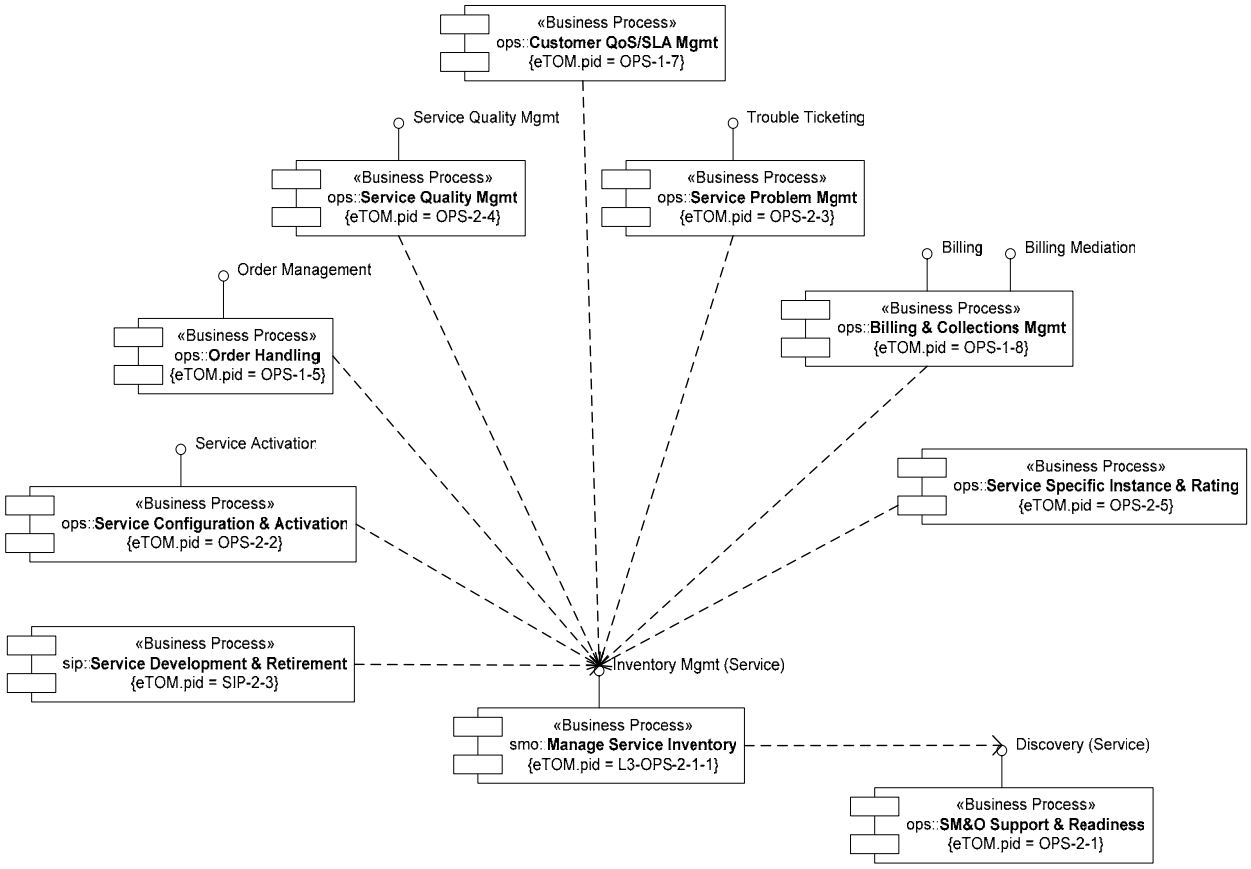

Figure 11: Service Inventory Interactions

3.6.4 References

For details on the Inventory API interfaces, see "JSR 142: OSS Inventory API.

3.7 Resource Inventory

3.7.1 Description

Resource Inventory is implemented as the OSS/J Inventory API applied on the CBE Resource model. It provides interfaces, as specified by the OSS/J Design Guideline, for populating, querying and updating the Resource Inventory repository. Resource Inventory is used by Order Management to design services and understand where capacity is available. It is also used for root-cause alarm analysis, resource activation, and discovery of network capacity. The Resource Inventory is populated by Resource Planning, Provisioning, and Problem Resolution processes. It is also updated through reconciliation using Resource Discovery.

The Inventory API provides persistency for shared CBE entities and, applied on the CBE Resource model, it provides operations for:

- creating/removing resource entities, specifications and associations
- updating resource entities, specifications and associations
- querying resource entities, specifications and associations
- managing a catalog of resource specifications

3.7.2 Mapping to Standards

TMF eTOM®: Resource Development & Retirement (Resource Specifications) and RM&O Support & Readiness (Resource)

ITU-T M.3400: Assignable inventory management function set; NE inventory query/notification function sets; Access to parameters and cross-connects in NE's function set

3.7.3 Interactions

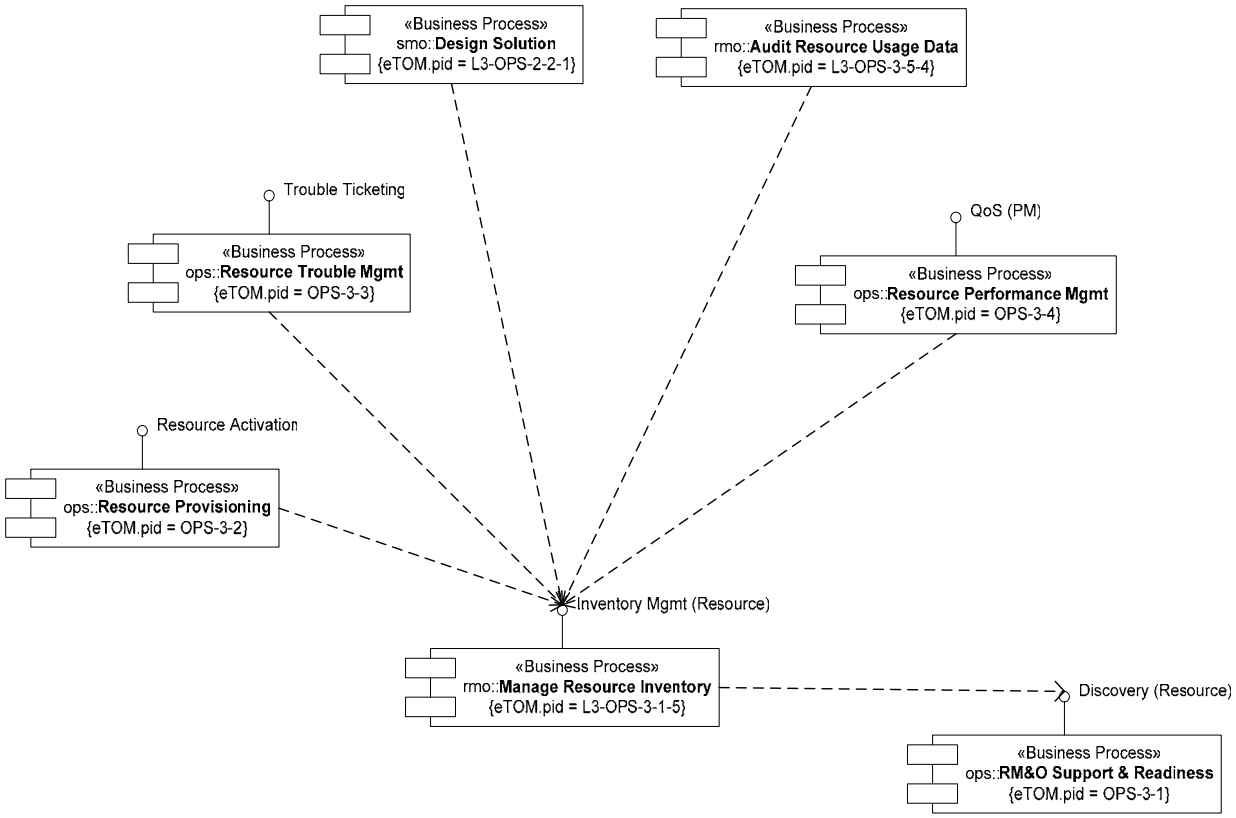

Figure 12: Resource Inventory Interactions

3.7.4 References

For details on the Inventory API interfaces, see "JSR 142: OSS Inventory API.

3.8 Service Discovery

3.8.1 Description

The Service Discovery API[13], part of the OSS/J Discovery API, provides interfaces, as specified by the OSS/J Design Guideline, for managing the data collection process. The Service Discovery API provides access to agents that discover and update services and service topology information that populates the Service Inventory. The Discovery agents query new generation Network Management Systems for the existing services deployed and produce reports of new and updated information.

The Service discovery API provides interfaces that allow clients to:

- create, remove, suspend, resume and configure service discovery agents
- query supported data types, formats and discovery agent types
- query specific service discovery agents
- be informed of service discovery events
- use XML and Messaging

3.8.2 Mapping to Standards

TMF eTOM®: SM&O Support & Readiness

ITU-T M.3400: Assignable inventory management function set; Access to parameters and cross-connects in NE's function set; Self-inventory function set

3.8.3 Interactions

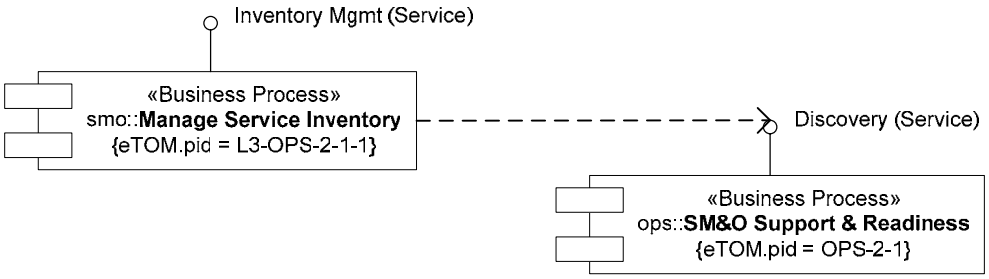

Figure 13: Service Discovery Interactions

[13] More accurately the CBE design pattern will enable the implementation of the Service Discovery API as the OSS/J Discovery API applied to the CBE Service model.

3.9 Resource Discovery

3.9.1 Description

The Resource Discovery API[14], part of the OSS/J Discovery API, provides interfaces, as specified by the OSS/J Design Guideline, for managing the data collection process. The Resource Discovery API provides access to agents that discover and update the physical resources and topology information that populates the Resource Inventory. The Discovery agents query the network resources, NMSs and EMSs in their native protocols, and produce reports of new or updated information.

The Resource Discovery API provides interfaces that allow clients to:

- create, remove, suspend, resume and configure resource discovery agents
- query supported data types, formats and discovery agent types
- query specific resource discovery agents
- be informed of resource discovery events
- use XML and Messaging

3.9.2 Mapping to Standards

TMF eTOM®: RM&O Support & Readiness

ITU-T M.3400: Assignable inventory management function set
Access to parameters and cross-connects in NE's function set
Self-inventory function set

3.9.3 Interactions

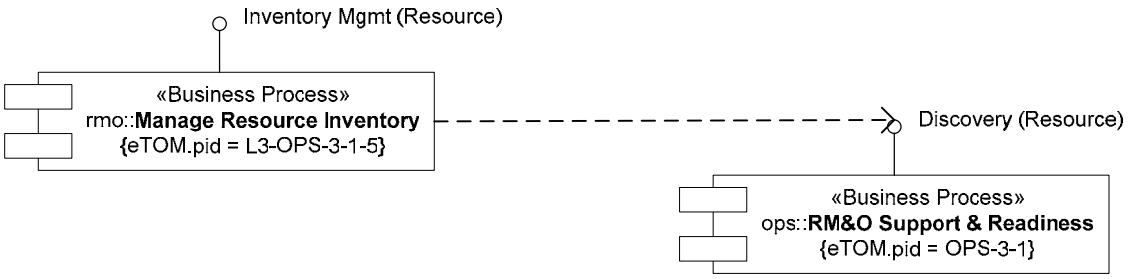

Figure 14: Resource Discovery Interactions

[14] More accurately the CBE design pattern will enable the implementation of the Resource Discovery API as the OSS/J Discovery API applied to the CBE Resource model.

3.10 Trouble Ticketing

3.10.1 Description

The Trouble Ticketing API provides interfaces, as specified by the OSS/J Design Guideline, for creating, tracking, and deleting trouble tickets. The Trouble Ticketing API enables creation and tracking of trouble tickets. It receives trouble ticket information either from the customer, or from service and network management applications such as Service Impact Analysis, and Root-Cause Alarm Analysis/Fault Monitoring. It enables tracking of the problem until resolution, and notification of clients when the problem has been resolved and the trouble ticket cleared.

The Trouble Ticket API provide interfaces that allow clients to:

- create, remove and cancel Trouble Tickets
- change the values of Trouble Tickets
- be informed of Trouble Ticket Changes via notifications

3.10.2 Mapping to Standards

TMF eTOM®: Problem Handling

NMF 501: Customer to Service Provider Trouble Administration Business Agreement

NMF 601: Customer to Service Provider Trouble Administration Information Agreement

ITU-T X.790: Trouble Management Functions for ITU-T applications

ITU-T M.3400: Trouble Administration function set group

3.10.3 Interactions

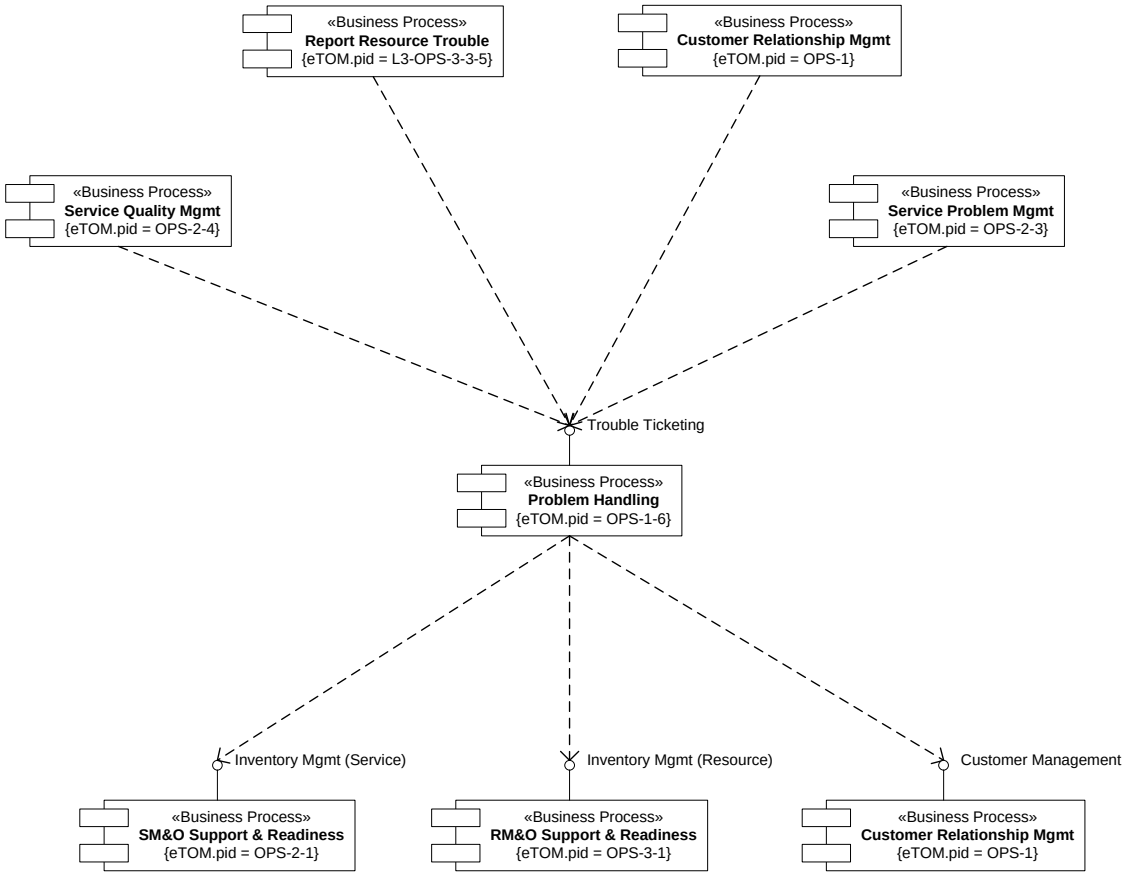

Figure 15: Trouble Ticketing Interactions

3.10.4 References

For details on the Trouble Ticketing API interfaces, see “OSS Through Java Trouble Ticket API”, JSR 091, v1.0, February 19, 2002.

3.11 Service Problem Resolution

3.11.1 Description

The Service Problem Resolution API provides interfaces, as specified by the OSS/J Design Guidelines, for manipulating actions and the rule-based engine to support automated execution of actions to support service problem resolution. In other words, the SPR API will provide a policy engine for service management and resolution of service problems at the system level, as part of a larger policy continuum. Clients to the Service Problem Resolution API can create, execute, cancel, and delete actions, and can configure the rule-based decision-making engine. Actions depend on the type of trouble, but examples are automatically re-configuring the network or creating a trouble ticket. The API takes into account the type of service, its priority, and its contracted Quality of Service (SLA) when deciding what sort of actions to take (if at all).

The Service Problem Resolution API provides interfaces that allow clients to:

- query actions, create, execute, cancel or delete actions
- configure the rule-based decision-making engine
- be notified of action creation, execution, cancellation and deletion

3.11.2 Mapping to Standards

TMF eTOM®: Service Problem Management

ITU-T M.3400: Priority service restoration function set

3.11.3 Interactions

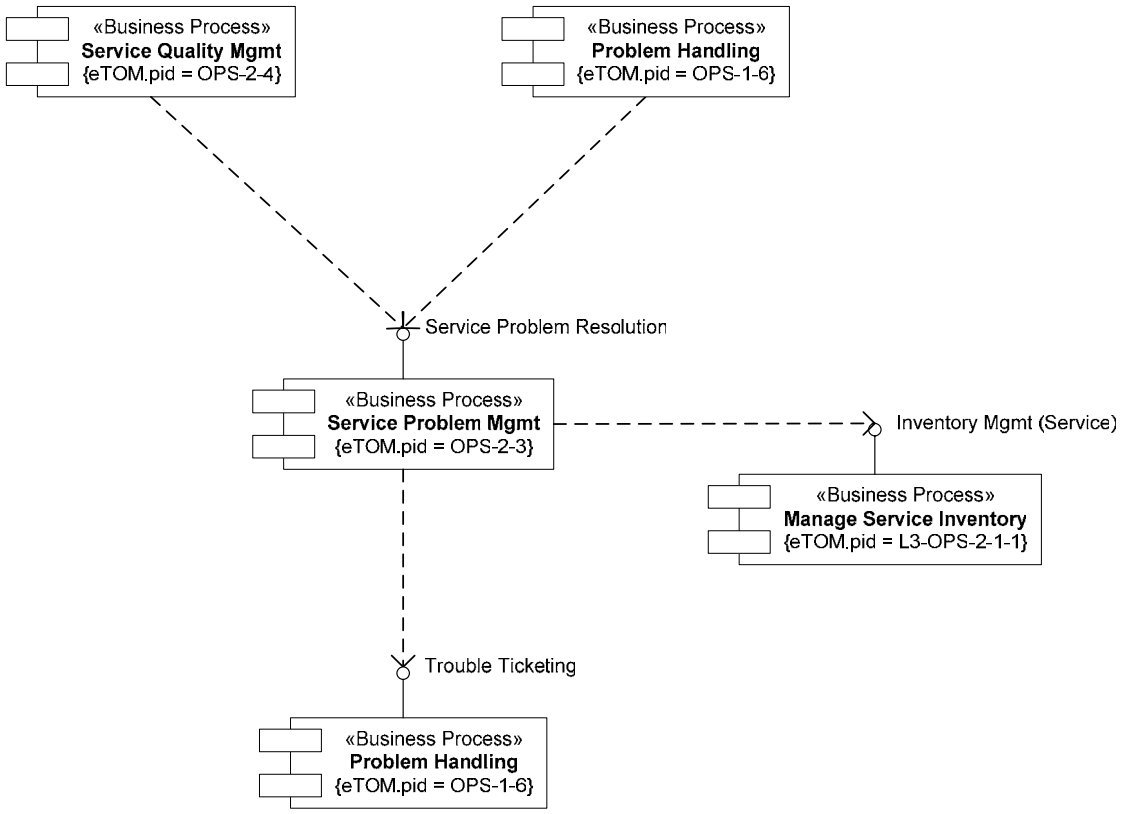

Figure 16: Service Problem Resolution Interactions

3.12 Customer SLA Management

3.12.1 Description

The Customer SLA Management API provides interfaces, as specified by the OSS/J Design Guideline, for creating, tracking, and deleting SLAs. The Customer SLA Management API enables monitoring, managing, and reporting of Quality of Service per customer/service as defined in their Service Level Agreements. This means creation, negotiation, updating, and deletion of SLA instances, and reporting of SLA violations. It also is responsible for the creation, updating, and deletion of SLA specifications.

The SLA API provides interfaces that allow clients to:

- create, remove and update Service Level Agreements
- query SLAs
- manage SLA specifications
- be informed of SLA violations

3.12.2 Mapping to Standards

TMF eTOM®: Product & Offer Development & Retirement (SLA Specifications)
Customer QoS/SLA Management (SLAs)

ITU-T M.3400: QOS performance goal setting function set
Subscriber service quality criteria function set
QOS performance assessment function set
Customer service performance summary function set
Customer traffic performance summary function set
Service availability goal setting function set
RAS assessment function set

3.12.3 Interactions

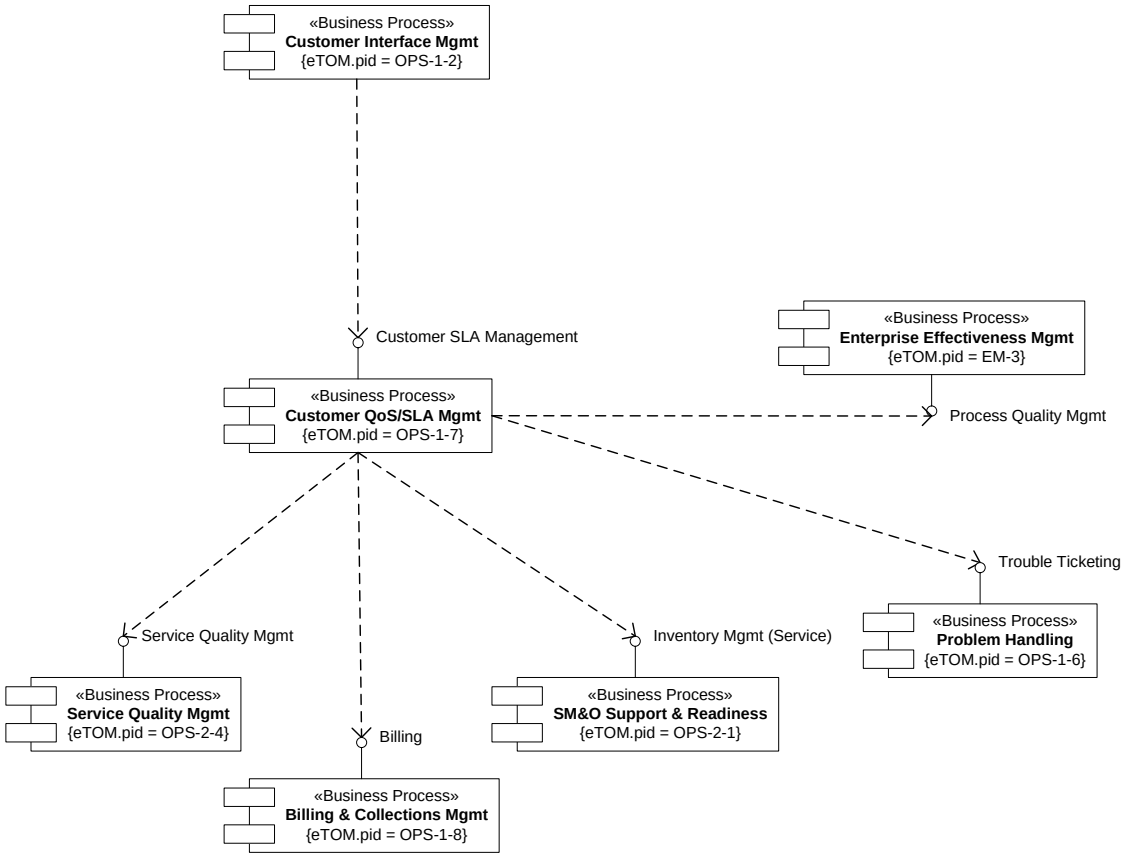

Figure 17: Customer SLA Interactions

3.13 Process Quality Management

3.13.1 Description

The Process Quality Management API supports and extends the Metrics API, which provides interfaces, as specified by the OSS/J Design Guideline, for creating, updating and deleting process affecting metric objects. These metric objects are then used by the Customer SLA Management as part of monitoring SLAs, and determining if any process-oriented violations have occurred, e.g., time to repair, time to activate, etc.

The Process Quality Management API provides interfaces that allow clients to:

- query metric objects (process affecting metric object instances and aggregated performance/usage data)
- create, update and delete metric objects
- create, update and delete threshold objects
- be notified of threshold crossing events
- be notified of metric (aggregated) data events

3.13.2 Mapping to Standards

TMF eTOM®: Operations Support & Readiness and Enterprise Management processes

ITU-T M.3400: Performance Monitoring Data Accumulation Function Set
Detection, Counting, Storage and Reporting function set
Performance Administration function set.

APQC: American Productivity & Quality Center

3.13.3 Interactions

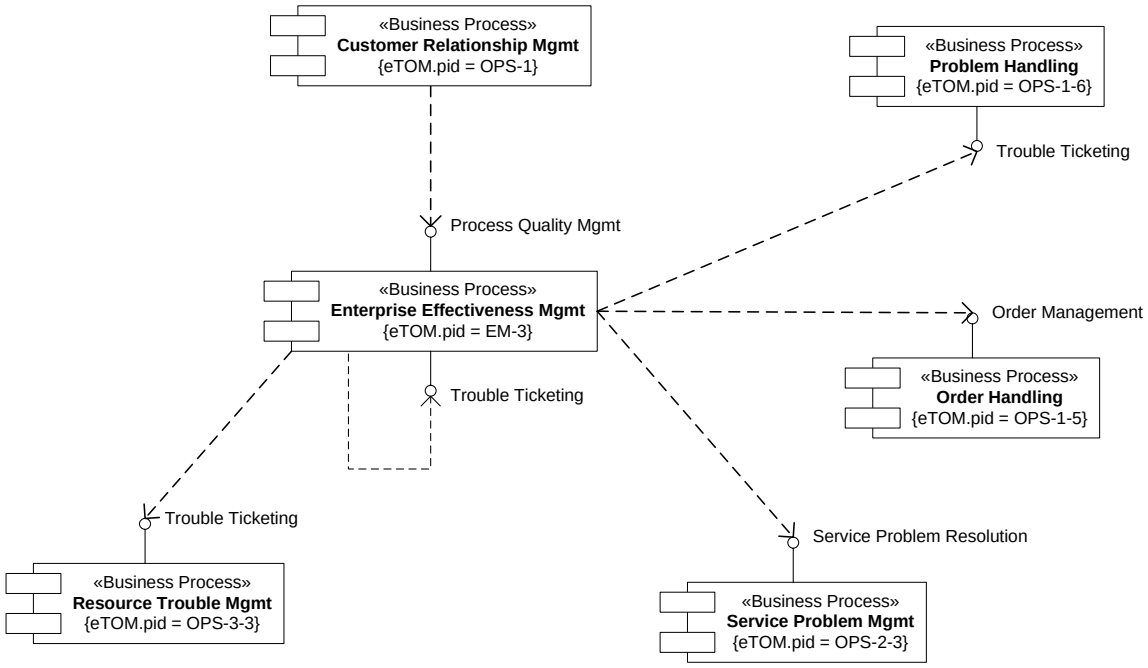

Figure 18: Process Quality Management Interactions

3.14 Service Quality Management

3.14.1 Description

The Service Quality Management API enables definition and calculation of service related quality objectives against pre-provisioned metrics. It enables correlation of service instance information, service alarms from Service Impact Analysis, and network performance metrics from Performance Monitoring to assess availability and quality metrics, and create service-oriented threshold crossing events. Service quality measurements and TCAs available through the API are consumed by Customer SLA Management to assess SLA violations.

The Service Quality Management API support and extends numerous interfaces from the OSS/J Quality of Service and it relies on the CBE service model as core information model. In particular, the Service Quality Management API provides interfaces that allow clients to:

- create, update, delete and query Service Level Specification objects, corresponding to line-items on Service Level Agreements
- create, update, delete and query Service Quality Objective objects (Service Objectives) against Key Quality Indicators (KQI) Parameters
- Create, update, delete and query Service Quality Report objects (historical reports on Service Quality Objectives and/or Service Level Specification evolution).
- subscribe for notifications on object violation events (Service Level Specifications or Service Quality Objectives being compromised)
- subscribe for notifications on availability of new service quality reports

3.14.2 Mapping to Standards

TMF eTOM®: Service Quality Management

TMF 506: Service Quality Management Business Agreement.

TMF GB923: Wireless Service Measurement Handbook

TMF GB917: Service Level Agreement Management Handbook.

3.14.3 Interactions

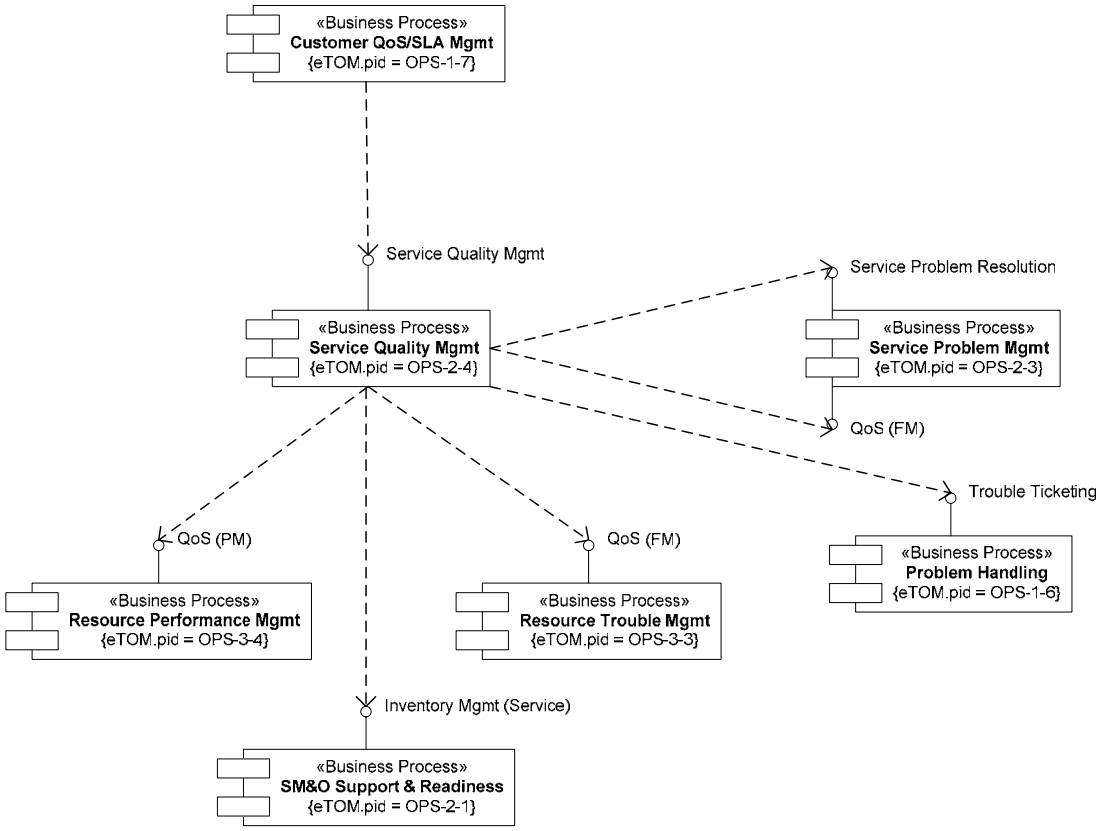

Figure 19: Service Quality Management Interactions

3.14.4 References

For details on the SQM API interfaces, see "JSR 210: Service Quality Management API"

3.15 Fault Monitoring

3.15.1 Description

The Fault Monitoring API, part of the QoS API, provides interfaces, as specified by the OSS/J Design Guideline, that allow clients to collect and acknowledge alarms. The API enables reception of alarms, state changes, and threshold crossing alerts from the network and maintaining a list of active alarms. It also ensures availability of the most current view of the alarm state of the network.

The Fault Monitoring API provides interfaces that allow clients to:

- acknowledge and unacknowledge alarms in an alarm list
- query an alarm list
- query the alarm count from an alarm list
- receive notifications for new, changed, and cleared alarms, alarm ACK state changes, and alarm list rebuild events

3.15.2 Mapping to Standards

TMF eTOM®: Resource Trouble Management

ITU-T M.3400: Alarm Reporting function set; Alarm Summary function set; Log Control function set.

3.15.3 Interactions

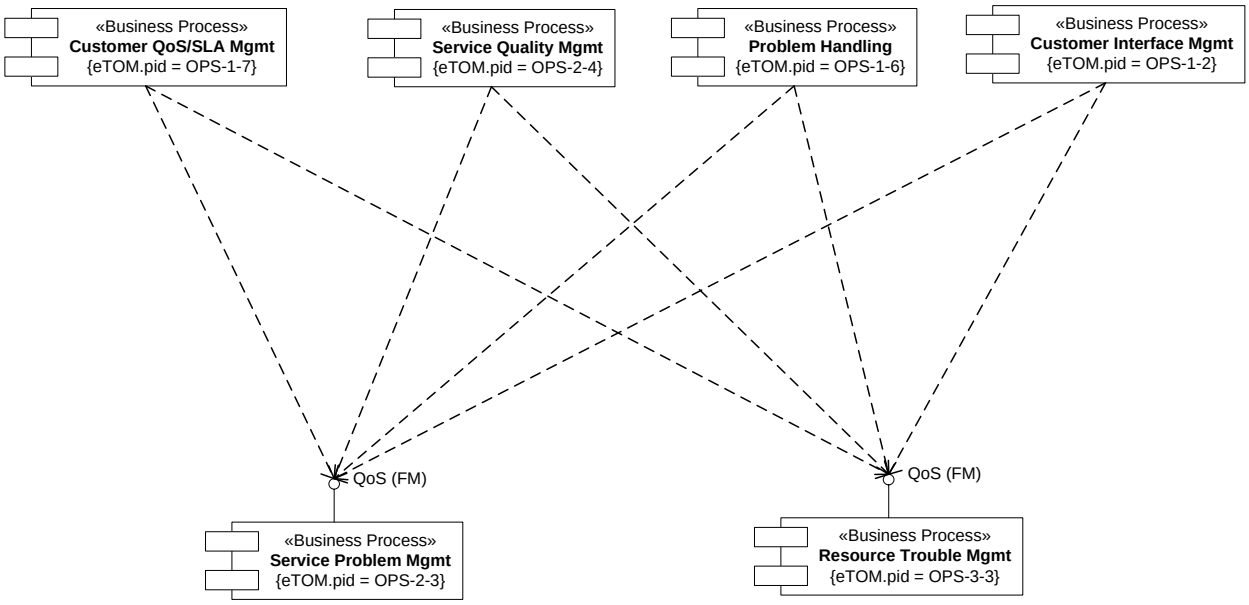

Figure 20: Fault Monitoring Actions

3.15.4 References

For details on the QoS API interfaces, see “OSS Quality of Service API”, JSR 090, v1.0, November 18, 2002.

3.16 Performance Monitoring

3.16.1 Description

The Performance Monitoring API, part of the QoS API, provides interfaces, as specified by the OSS/J Design Guideline, for creating and deleting metric and threshold objects. The API supports the collection of performance data from the network, setting thresholds, and generating/forwarding threshold crossing events. Data collected may be directly from devices (e.g., a MIB value) or may be a combination of metric values used in a calculation to create a more meaningful performance data value. The same principle applies to thresholds. Performance Monitoring may be configured to generate thresholds of thresholds.

The PM Measurement API provides interfaces that allow clients to:

- Create, remove, suspend and resume measurement jobs
- Get measurement jobs
- Get supported report formats
- Get supported observable objects
- Retrieve measurement reports
- Use bulk/mass operations

The PM Threshold API provides interfaces that allow clients to:

- Create, remove, suspend and resume threshold monitor jobs
- Query threshold monitor jobs
- Discover system performance measurements via a discovery mechanism that exposes a system's observable objects
- Determine the supported granularities of a system based on a threshold monitor job configuration
- Use bulk/mass operations

3.16.2 Mapping to Standards

TMF eTOM®: Resource Performance Management

ITU-T M.3400: Data aggregation and trending function set
Traffic performance monitoring function set
NE threshold crossing alert processing function set
NE trend analysis function set
Performance monitoring and accumulation function set
Network performance monitoring event correlation and filtering function set

3.16.3 Interactions

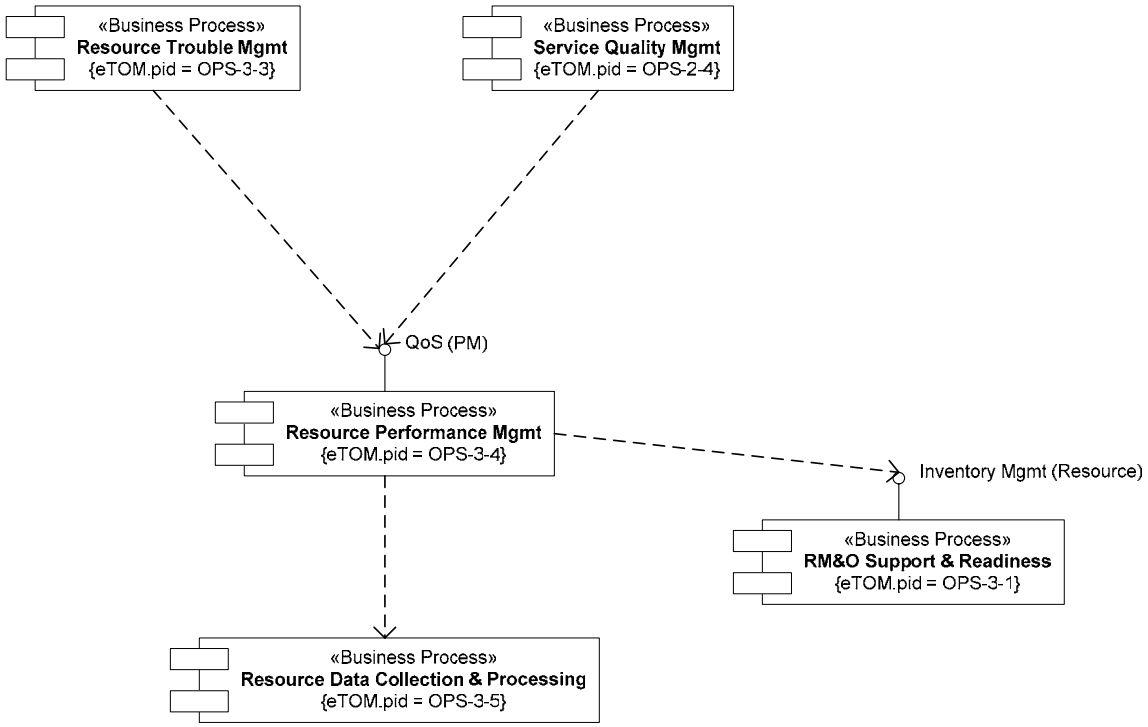

Figure 21: Performance Monitoring Interactions

3.16.4 References

For details on the QoS API interfaces, see "OSS Quality of Service API", JSR 090, v1.0, November 18, 2002.

3.17 Billing

3.17.1 Description

The Billing API provides interfaces, as specified by the OSS/J Design Guideline, for creating and deleting billing objects. The API enables for rating services and calculating billing records, sending invoices to customers, processing their payments, and performing payment collections. Also, the Billing API handles inquiries by the customer about bills, and handles billing problems.

The Billing API provides interfaces that allow clients to:

- Create, remove, suspend and resume billing jobs
- Get billing jobs
- Get supported report formats
- Use bulk/mass operations
- Assign rates, discounts, etc., to services and products

3.17.2 Mapping to Standards

TMF eTOM®: Billing & Collections Management and Service & Specific Instance Rating processes

ITU-T M.3400: Collections and Finance function set group

3.17.3 Interactions

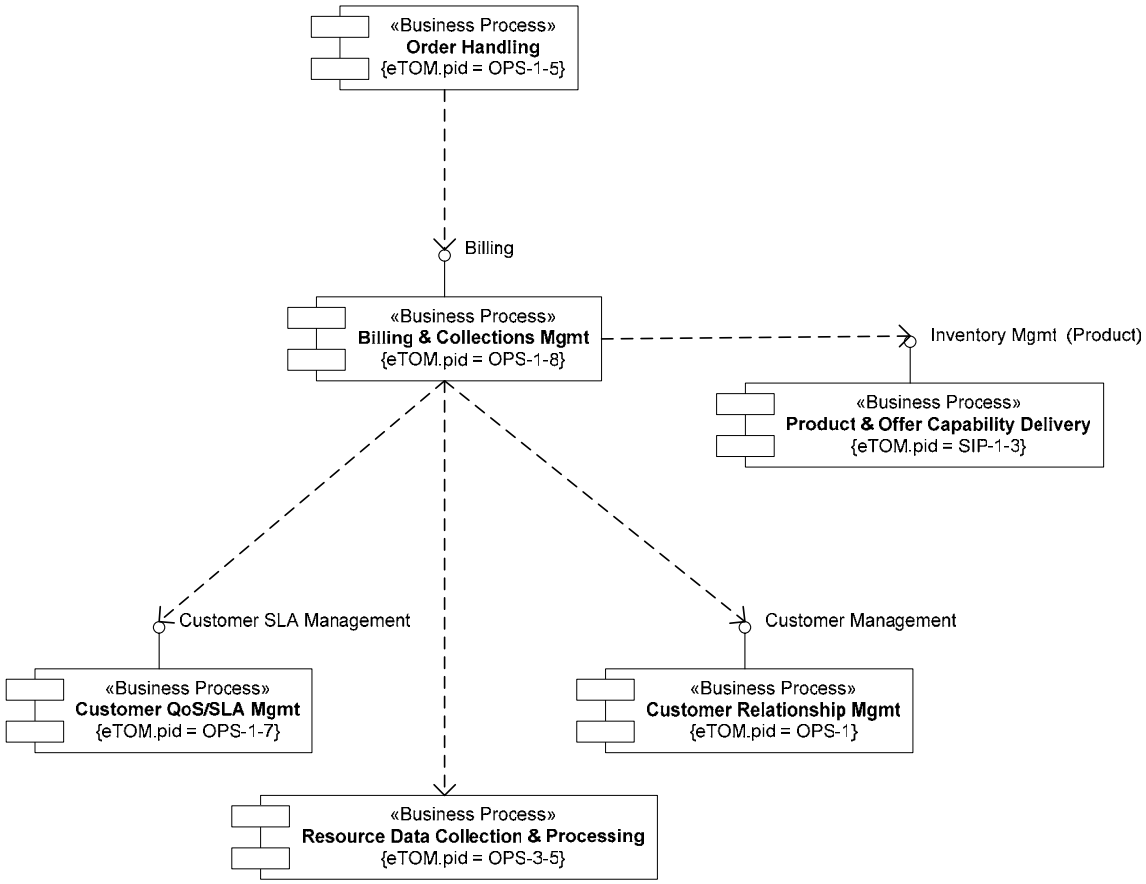

Figure 22: Billing Interactions

3.17.4 References

For details on the Billing Mediation API, see "JSR 130: OSS Billing Mediation API.

3.18 Billing Mediation

3.18.1 Description

The Billing Mediation API provides interfaces, as specified by the OSS/J Design Guideline, for creating and deleting mediation jobs. This API enables for matching usage to individual services for usage-based services. This information is then available to Billing for invoicing the customer.

The Billing Mediation API provides interfaces that allow clients to:

- Create, remove, suspend and resume mediation jobs
- Get mediation jobs
- Get supported report formats
- Use bulk/mass operations

3.18.2 Mapping to Standards

TMF eTOM®: Resource Data Collection & Processing

ITU-T M.3400: Service usage correlation function set; service usage validation set; usage distribution function set.

3.18.3 Interactions

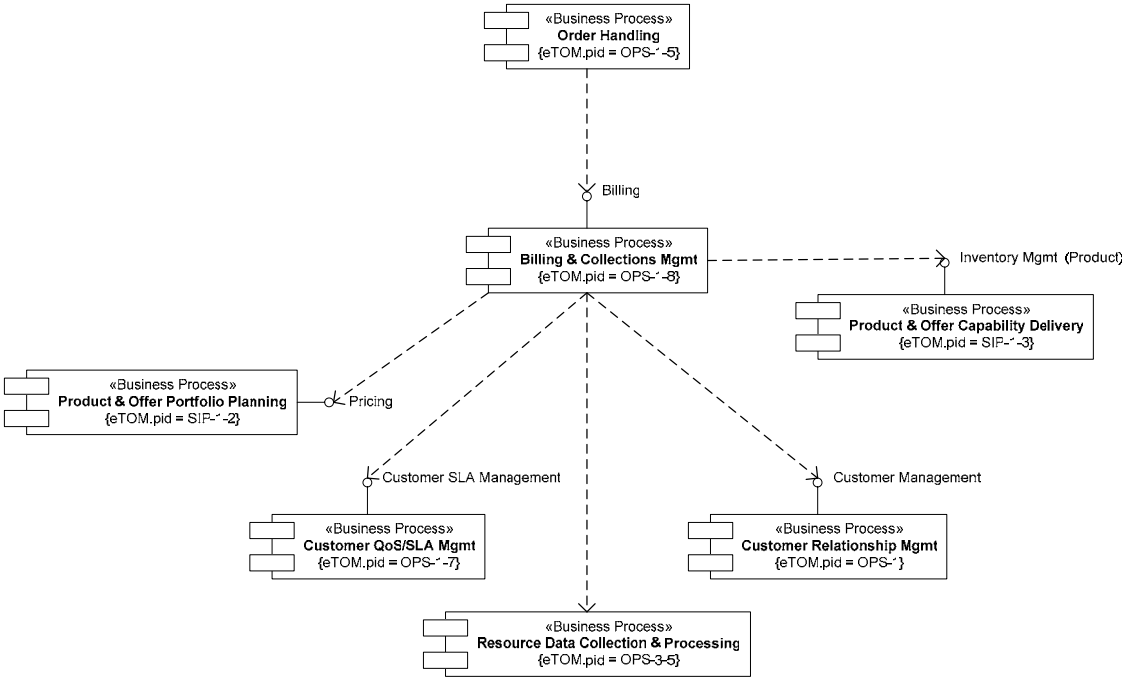

Figure 23: Billing Mediating Interactions

3.19 Product and Offer Pricing

3.19.1 Description

Product and offer pricing has historically been the domain of billing, and has recently been duplicated in Ordering, Customer Relationship Management, Sales Force Automation, and Enterprise Resource Planning. As more systems and channels are concerned with pricing and offers, it makes sense to view this vital function as a role unto itself.

The offer and pricing functions are essential for a company to have dynamic responses to changing market conditions on a real-time basis. Resource utilizing transactions could be priced based on the current scarcity of the resource. A carrier could match the prices offered by competitors on a geographic basis. Lower cost channels could receive favourable pricing vis-a-vis higher cost channels.

The Pricing API will provide a central point to consult and determine offers and prices based on a variety of criteria including the specific customer profile, location, current promotions, other parties in the transactions, factors peculiar to the request, and any other set of complex, possibly inter-related points. The subscriber systems can use the Pricing API to present a list of offers, determine either static or one-time prices, or present prices that change over time and across the customer base.

The Pricing API will, by necessity, contain the offer and pricing definitions. These definitions, in aggregate, form the Pricing Catalogue. In the static pricing scenario, once a prospect selects an offer, its price must be bound to the order. That prevents future offer and price changes from effecting current customers, if that is the business' desire.

The Pricing API covers both service and resource pricing via their association with products (e.g., both data services subscriptions and purchase of an end-user terminal device). The API will allow the expression of the pricing rules or algorithms that will be used to calculate all fees be they for product or equipment purchase, service installation, recurring subscriptions, service usage fees or cancellation fees.

To compete, a carrier needs sophisticated offer and pricing methods and technologies, much like the mechanisms that have evolved in the air transport industry. Systems such as Billing, Ordering, CRM, ERP, and SFA should be seen as subscribers to pricing, not owners of it. This API promotes Offer and Pricing Management to first-class status within the enterprise systems environment, and supports the TeleManagement Forum's (http://tmforum.org/) eTOM and its Product & Offer Portfolio Planning and Capability Delivery components.

3.19.2 Mapping to Standards

TMF eTOM®:

ITU-T M.3400: Tariffing/pricing
Pricing strategy function set
Tariff and price administration function set
Settlements policy function set
Feature pricing function set
Provision of access to tariff/price information function set
Rating usage function set
Totaling usage charges function set

3.19.3 Interactions

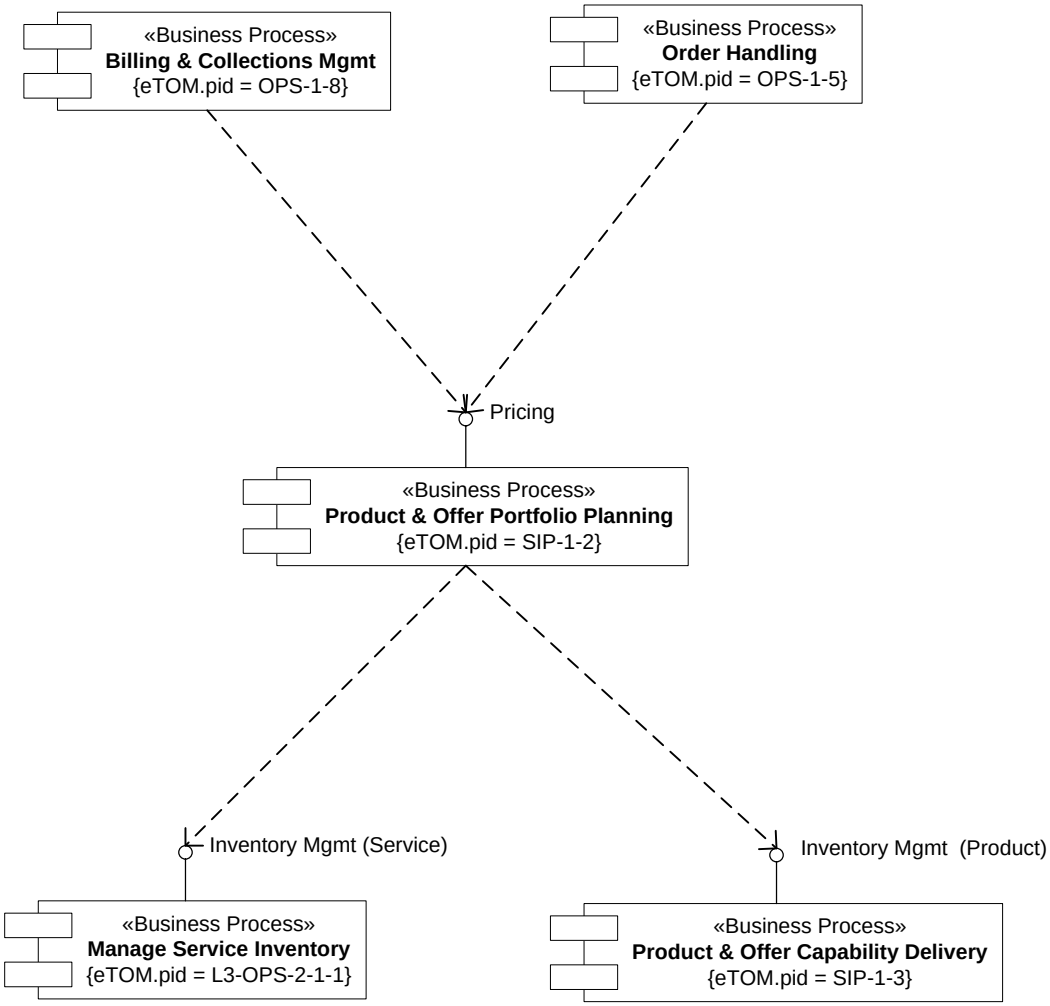

Figure 24: Pricing API Interactions

3.19.4 References

For details on the Pricing API interfaces, see “Pricing API”, JRS 251 v1.0, August, 2004

4 Business Scenarios

This section provides typical business scenarios providing a context to the OSS/J APIs as defined in section 22. Please note that the coloring scheme for the figures part of this section is the same as the one in Figure 4: Core API Priorities These business scenarios are provided as examples to illustrate the use and completeness of the current roadmap of APIs. Each scenario has valid alternatives that may be preferred by Service Providers to run their operations.

4.1 DSL provisioning

"DSL" refers to Digital Subscriber Line high speed service. In the generic DSL provisioning scenario we assume:

The service provider does not own the local loop. Therefore a Local Service Request is required to acquire the local loop from a trading partner. Note that, if the service provider owns the local loop, this only simplifies the scenario.

There is no voice service provisioning as part of the DSL provisioning. This would only make the scenario more complex without adding value.

The switch is owned by the service provider.

The black and white bars in, Figure 25: Generic DSL Activation, Figure 26, Figure 27, Figure 28 and Figure 30 refer to manual tasks (white) and automated tasks (black) outside the current scope of this roadmap.

4.2 Generic DSL Activation

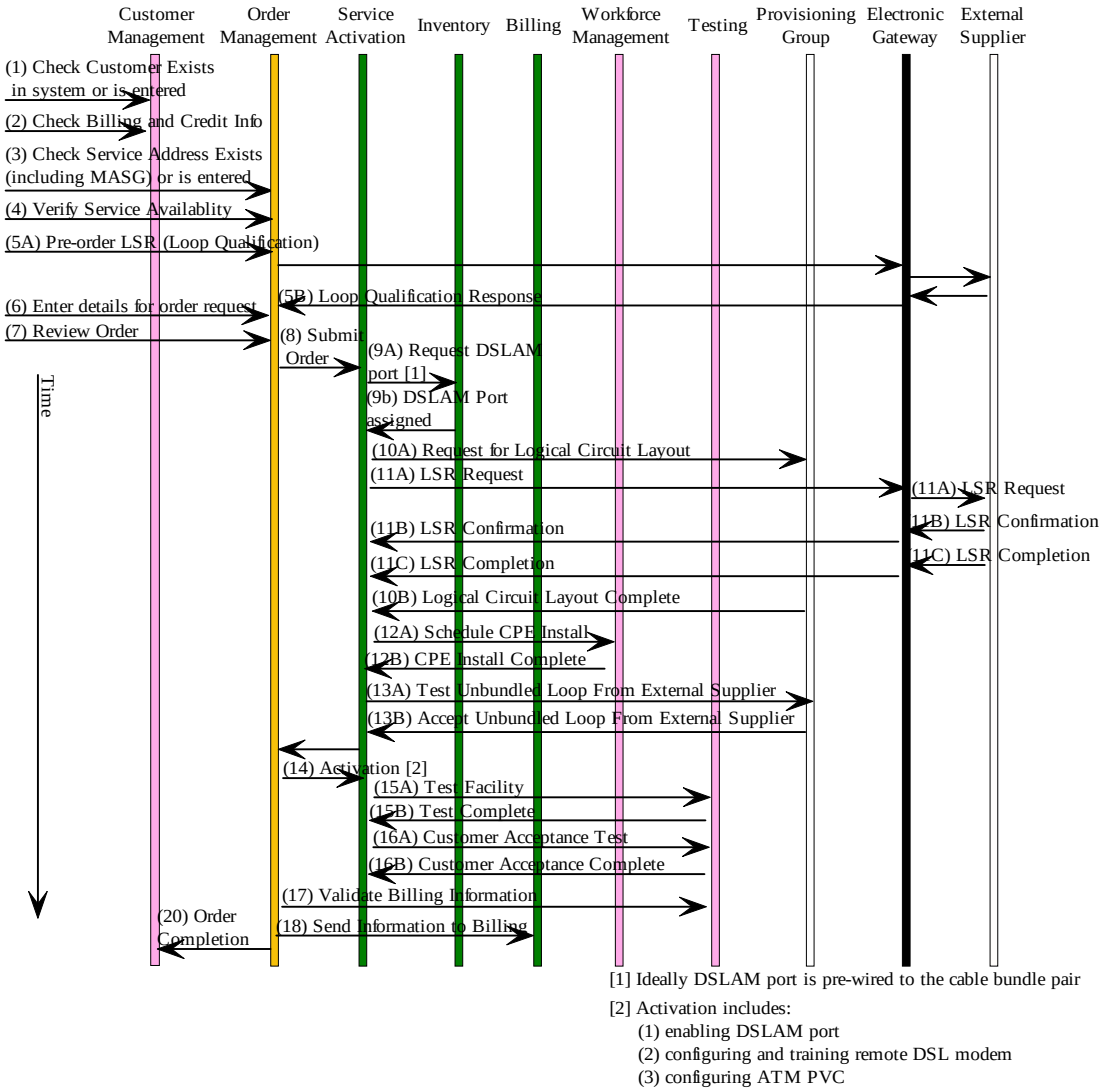

Figure 25: Generic DSL Activation

4.2.1 Generic DSL Cancellation

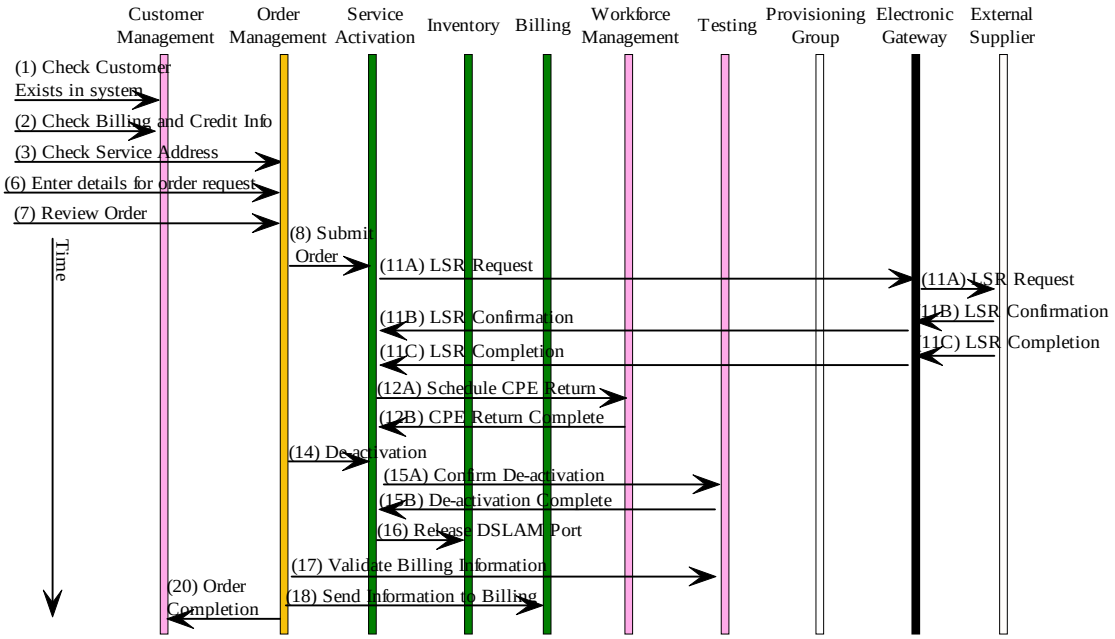

Figure 26: Generic DSL Cancellation

4.3 Fault Management

Fault reported by the network; fault is resolved with a manual repair; client is advised.

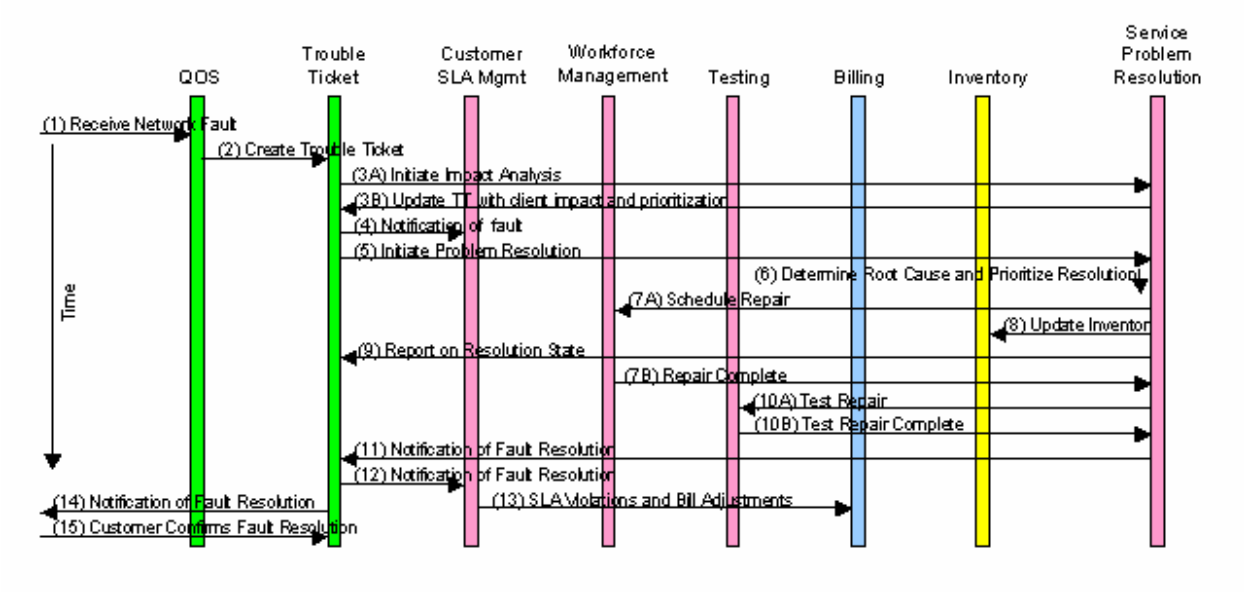

Figure 27: Fault Resolution Scenario with Manual Repair

Fault reported by the network; fault resolution is manually initiated but handled through automated network re-routing; client is advised.

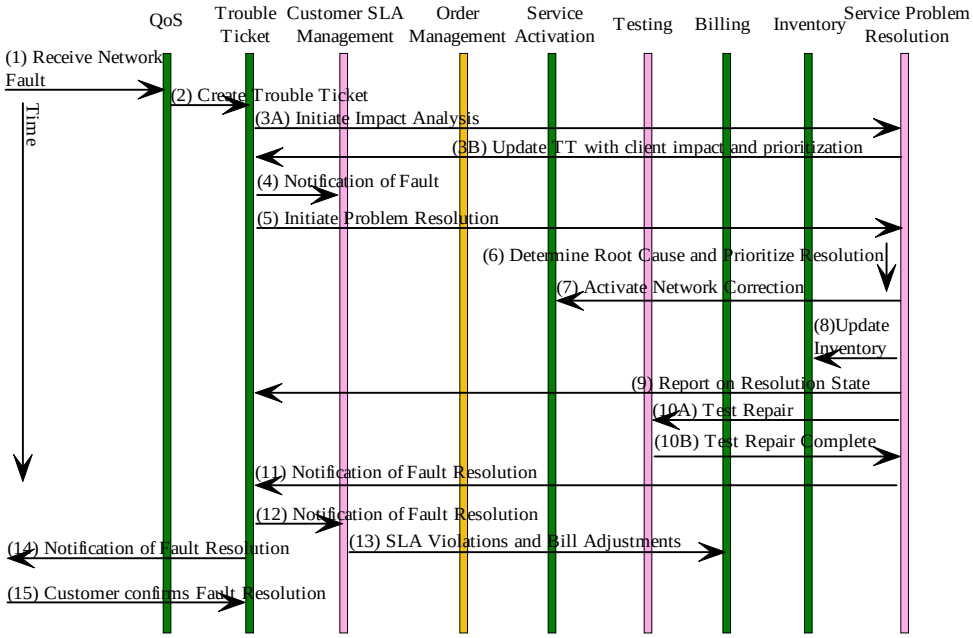

Figure 28: Fault Resolution Scenario with Automated Network Correction

Note that Figure 28 differs from Figure 27 in having Resource Activation (Order Management & Activation) replacing Workforce Management, i.e., step 7 changed and step 9 suppressed.

Fault detected by the network; fault resolution is completely automated.

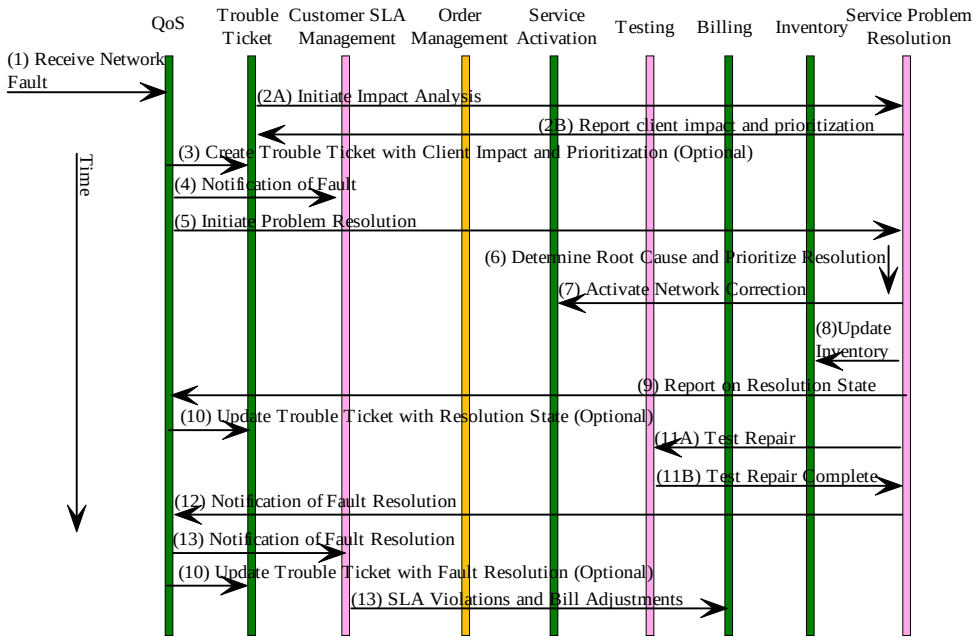

Figure 29: Automated Fault Detection and Resolution Scenario

Note that Figure 29 differs from Figure 28 in having the resolution steps initiated from the QoS system, the Trouble Ticket logging as an optional step, and the customer is not directly made aware of the incident.

4.4 Performance Management

The customer complains about performance and the complaint is resolved.

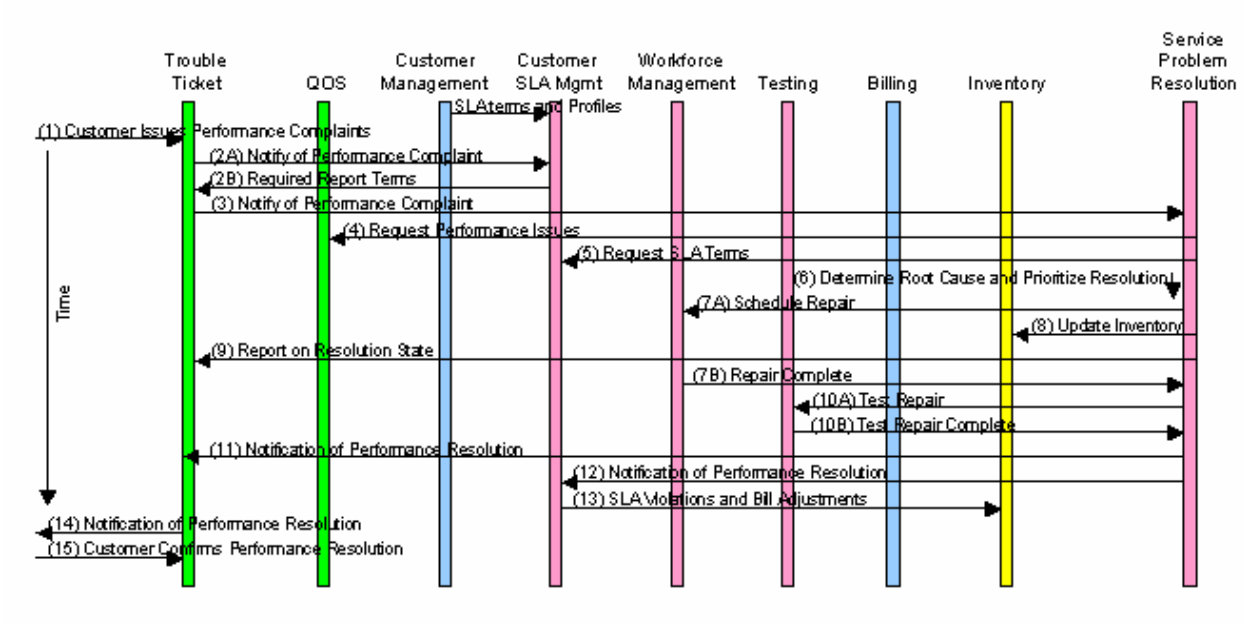

Figure 30: Performance Management Scenario

4.5 Bill Generation Management

The bill generation scenario accounts for flat rate usage billing and SLA violations.

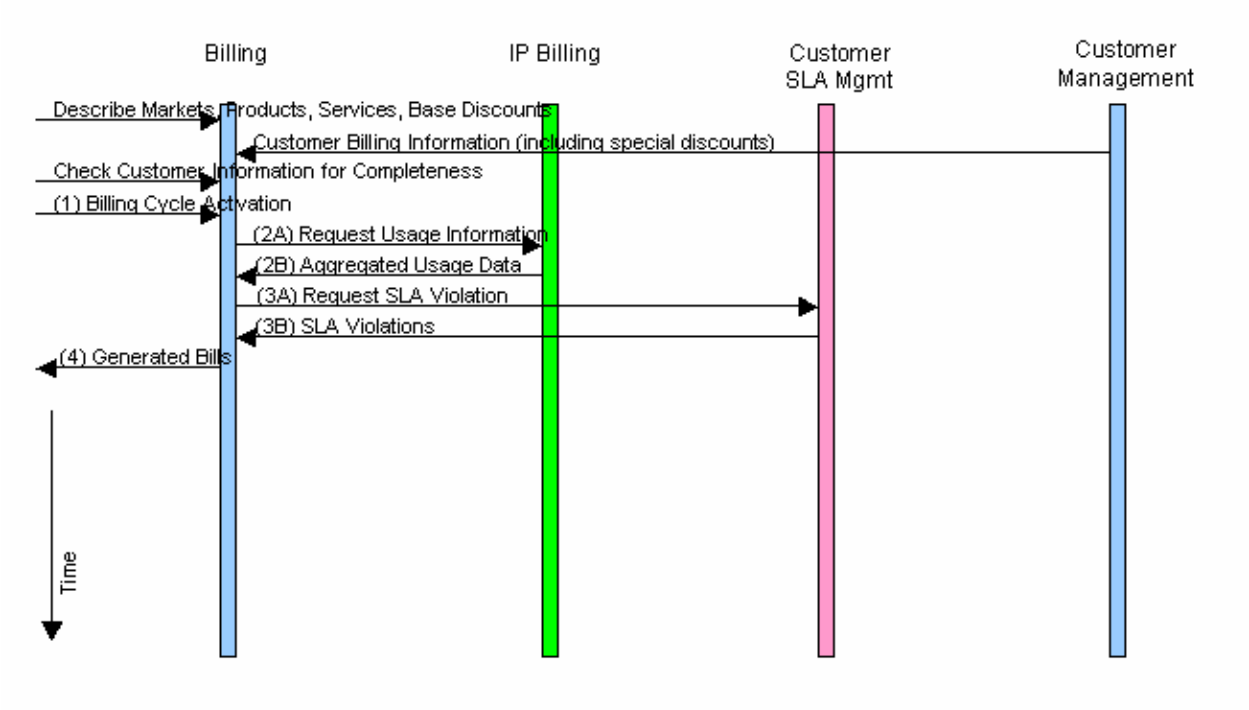

Figure 31: Bill Generation Scenario

4.6 Inventory Management

The inventory management scenario accounts for resource and service discovery in the network and reconciliation with the inventory data.

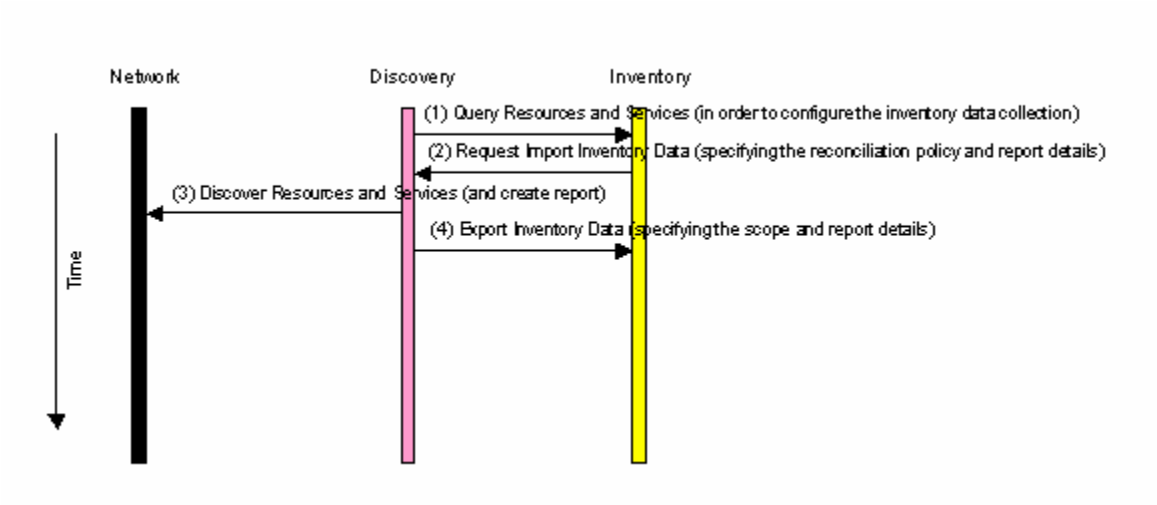

Figure 32: Inventory Management Scenario