Service Activation - DDP BA - Part 2: Service Activation Interface (SAI)

TMF518_SA_2

Version 1.2

September, 2011

Notice

No recipient of this document and code shall in any way interpret this material as representing a position or agreement of TM Forum or its members. This material is draft working material of TM Forum and is provided solely for comments and evaluation. It is not "Forum Approved" and is solely circulated for the purposes of assisting TM Forum in the preparation of final material in furtherance of the aims and mission of TM Forum.

Although it is copyrighted material of TM Forum:

- Members of TM Forum are only granted the limited copyright waiver to distribute this material within their companies and may not make paper or electronic copies for distribution outside of their companies.
- Non-members of the TM Forum are not permitted to make copies (paper or electronic) of this draft material other than for their internal use for the sole purpose of making comments thereon directly to TM Forum.
- If this material forms part of a supply of information in support of an Industry Group Liaison relationship, the document may only be used as part of the work identified in the Liaison and may not be used or further distributed for any other purposes

Any use of this material by the recipient, other than as set forth specifically herein, is at its own risk, and under no circumstances will TM Forum be liable for direct or indirect damages or any costs or losses resulting from the use of this material by the recipient.

This material is governed, and all recipients shall be bound, by all of the terms and conditions of the Intellectual Property Rights Policy of the TM Forum (http://www.tmforum.org/Bylaws/1094/home.html) and may involve a claim of patent rights by one or more TM Forum members or by non-members of TM Forum.

Direct inquiries to the TM Forum office:

240 Headquarters Plaza,
East Tower – 10th Floor,
Morristown, NJ 07960 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

Table of Contents

List of Requirements

List of Use Cases

List of Figures

List of Tables

Executive Summary

This document is Part 2 of the MTOSI Service Activation Document Delivery Package (DDP). It covers requirements and use cases for a service activation interface where it is assumed that one side of the interface supports and understands the eTOM's Customer Relationship Management (CRM) concepts, e.g., product and customer, and the other side of the interface supports and understands the eTOM Service Management & Operations (SM&O) concepts, e.g., customer facing service and service order. This interface also allows for the management of service orders that are created as a result of a service activation request.

1 Introduction

1.1 DDP Structure

In order to allow for more efficient release delivery, the previous monolithic BA, IA and SS documents have been partitioned into smaller self-contained (though not independent) units called Document Delivery Packages (DDPs).

This is similar to the 3GPP concept of Integration Reference Point (IRP). The basic idea is that the Interface, which is specified by the entire document set (of a release), is partitioned into DDPs where each DDP specifies “a certain aspect” of the Interface, which needs to be very clearly scoped.

There are three kinds of DDPs:

- the FrameWork DDP (FMW) – this DDP contains the generic artifacts that are applicable to all the other DDPs.
- Data Model DDP (DM-DDP) – a DDP that concerns a data model (entities, data structures, attributes, state, but no operations)
- Operation Model DDP (OM-DDP) – a DDP that concerns a computational model (operations, notifications, transactions) for a given functional area (such as resource inventory management)

The unified deliverables structure for any given MTOSI / MTNM product release is as follows:

- Product Release Notes:
  - a scope specification for the type and extent of the delivered product,
  - the partitioning of the release into DDPs (i.e., definitions of various aspects of the release),
  - and an overview of the release’s (delta) deliverables;
- For each DDP:
  - Business Agreements (BAs): a business view specification
  - Information Agreements (IAs): a system view specification
  - Interface Implementation Specifications (ISSs): implementation and deployment view specification per supported enabling technology (mapping of the IA to either CORBA (IDL, services usage) or XML (WSDL, XSD, bindings…)
  - Supporting Documentation: normative and informative supporting documents.
- Reference Implementation (optional) of core IIS fragments for selected interfaces and enabling technologies.

1.2 Document Overview

This document is Part 2 of the MTOSI Service Activation Document Delivery Package (DDP). (The other two parts are Service Activation - DDP BA - Part 1: Overview and Service Activation - DDP BA - Part 3: Service Component Activation Interface (SCAI)). It covers requirements and use cases for a Service Activation Interface where it is assumed that one side of the Interface supports and understands the

eTOM's Customer Relationship Management (CRM) concepts and the other side of the interface supports and understands the eTOM Service Management & Operations (SM&O) concepts. This interface also allows for the management of service orders that are created as a result of a service activation request.

1.3 Document Structure

This document has the following sections:

- Section 1 is the introduction.
- Section 2 defines the business problem and scope.
- Section 3 contains the requirements.
- Section 4 has the use cases.
- Section 5 provides a traceability matrix between use cases and requirements, and vice versa.
- Section 6 provides a summary and list of open issues.
- Section 7 lists the references used in this document and notes any IPR claims.
- Section 8 notes the contact for this document and has administrative information such as the document version history and list of acknowledgements.

1.4 Terminology Used In This Document

The terminology used in this document is covered in Service Basic DDP BA and SD0-1.

In addition, the following term(s) are noted

- The "***requesting OS***" in this document is the OS that is using the interface. It is assumed that requesting OS has Customer Relationship Management (CRM) capabilities (at least enough intelligence to formulate the Service Activation Interface (SAI) requests).

2 Business Problem Description, Project Scope

2.1 Project Scope

The TM Forum Integration Program is responsible for all of the interface and business services work within the TM Forum. In some cases, interface work is delegated to other teams but the final verification for technical uniformity and integrity is the responsibility of the TM Forum Integration Program.

Initially, the TM Forum Integration Program was formed to coordinate the various existing TM Forum interfaces activities (as shown in **Figure 2-1**). In particular, the responsibility for maintaining MTOSI and MTNM is now covered by the MTOSI-MTNM Users Group which is a team within the TM Forum Integration Program. The long term plan (which is already well under progress) is to migration the various input work to a single harmonized suite of interfaces.

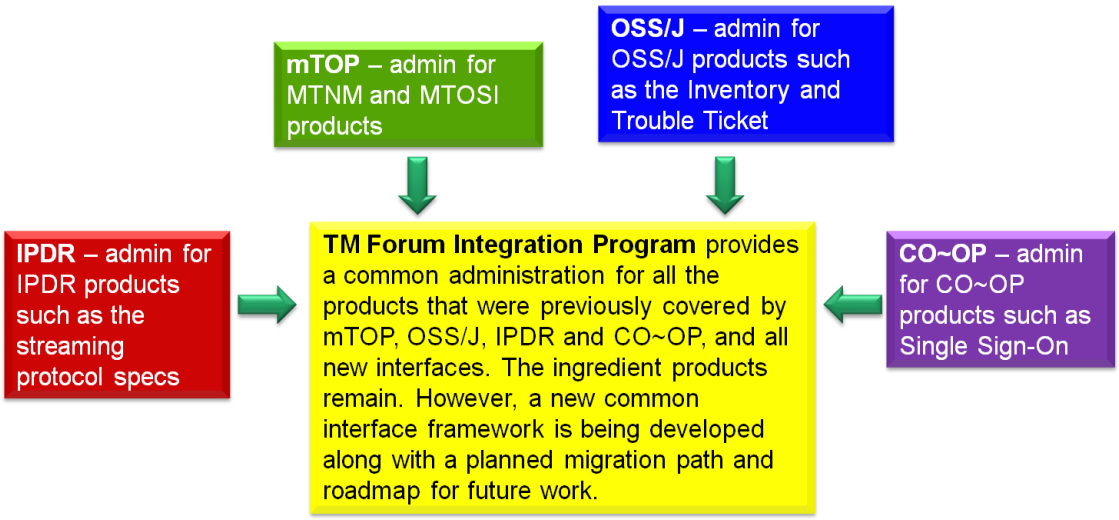

Figure 2-1. Inputs to the TM Forum Integration Program

Figure 2-2 provides a summary of the team within the TM Forum Integration Program as well as a few teams outside of the program but which also do some interface work. In terms of MTOSI and MTNM, the main input for updates come from the Resource and Service Management Team.

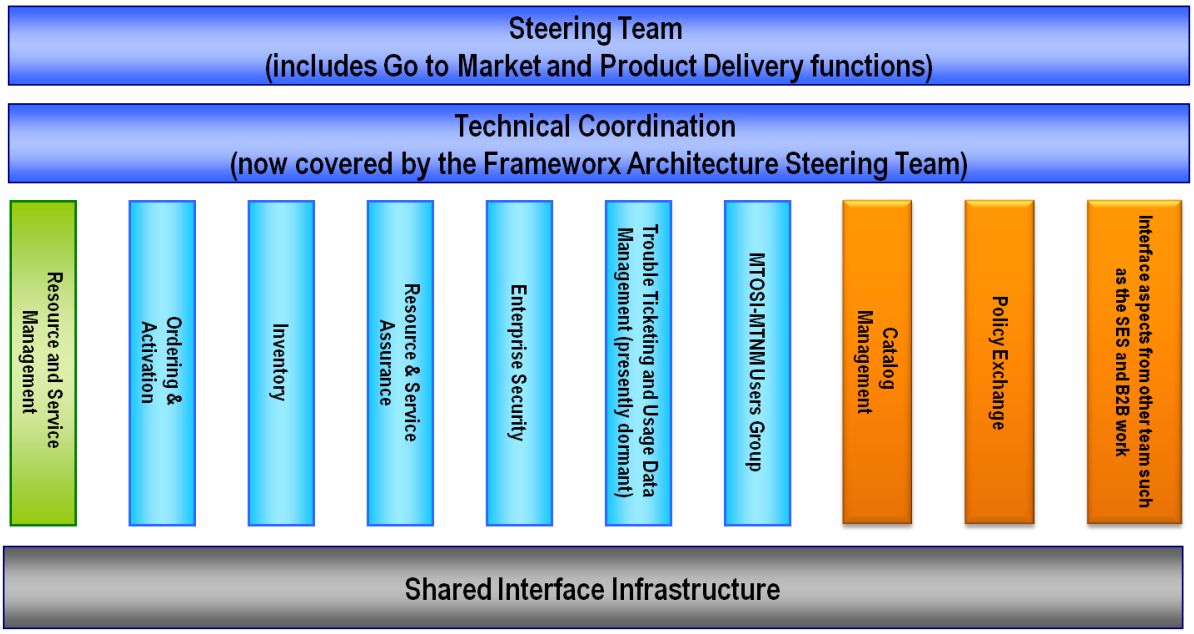

Figure 2-2. TM Forum Integration Program

2.2 Supported Business scenarios

This document covers several basic scenarios, i.e.,

- the establishment of a set of services in support of a given product instance (the steps involve feasibility check, design, reserve (optionally), provision and activate)
- the tear-down of a set of services in support of a given product instance (the steps entail deactivate, termination and retiring)
- the modification of a set of services in support of a given product instance.

All of the above scenarios have both a service order aware and a service order unaware case since some OSSs do not support orders and others do.

2.3 Benefits

The Service Activation Interface (SAI) defined in this document is intended to benefit service providers by offering an extensible interface that can be used for service activation between a CRM and SC&A application or to be more specific between one OSS that supports CRM capabilities and another OSS that supports SC&A capabilities. The goal is to provide the service provider with a single interface (the SAI) that can be used in lieu of the many pair-wise proprietary interfaces that current exist between CRM and SC&A applications. It is expected that a reduction in the number of interfaces (from many to hopefully one) will lead to lower integration and maintenance times and associated costs.

MTOSI and MTNM provide a set of Interface specifications that allow for resource and service management (with only MTOSI covering service management, but with MTOSI and MTNM both covering resource management, using very much the same information model).

These specifications are intended to lower design, implementation, Verification Validation & Testing (VVT), and maintenance costs for management interfaces. These Interfaces are intended for use by service providers, suppliers of equipment and OSS suppliers. The intention is to also encourage system integrator usage of management systems that make use of the Interfaces.

In particular, the followed approach tends to minimize the cost of integration, provide access to all necessary information and control, and support all vendor/operator differentiation. The intent of the interface is to provide compatibility among different version, for a detailed description see SD2-6_VersioningAndExtensibility.

2.3.1 Service Provider Benefits

The service provider benefits are as follows:

- One stop shopping concerning feature requests for much of the TM Forum contract specification work is part of the defined Change Control Group (CCG) process that TM Forum makes available in order to control the interface.
- The technical deliverables are also of high value to the service provider. The Interface specifications allow for an open, multi-supplier environment, shorten delivery times and lower integration costs.
- The MTOSI and MTNM products provide an integrated, multi-technology interface with support for most key layer 1 and layer 2 transport technologies. This is in contrast to earlier approaches where each technology-specific forum provided a single-technology management interface. The service provider was faced with having to use many different, uncoordinated management interfaces.
- These products are not bound to any one middleware, transport or computing language. So, the service provider will be able to evolve to new technologies as they arise.

2.3.2 Supplier Benefits

The supplier benefits are as follows:

- Fewer Adapters leads to Lower Costs – in as much as MTOSI and MTNM gain market penetration (and there has already been significant market acceptance of these interfaces), the supplier is faced with the need to build fewer adapters between their products and the products of their partners. A supplier can also directly see cost savings in the use of the Interfaces among its own products (as the need for an open interface arises).
- Lower Middleware Transitions Costs – the Interfaces are defined to be middleware and transport independent. So, the supplier can migrate from one middleware or transport technology to another without changing the supporting business logic in the code.
- Increase Usage by System Integrators (SIs) – a supplier's support of their own "open" interfaces goes only so far to encourage SIs. Clearly, an SI would like to make use of supplier products (both equipment and OSS suppliers) that make use of well supported standard interfaces rather than supplier specific interfaces. The latter case forces the SI into a situation characterized by many pair-wise negotiations between various suppliers.
- Lower Training Cost – in as much as a supplier re-uses the Interfaces for multiple products and for multiple customers, the various training costs are lower because the designers, system engineers, developers and testers are using the same Interfaces over and over again.

2.4 Mapping of Processes to eTOM Business Framework

The eTOM is a business process framework, or model, that provides the enterprise processes required for a service provider. There are various levels of granularity, or decomposition, of processes that have been defined. The MTOSI service management initiative has considered a Level 3 decomposition of the eTOM SM&O layer in order to fully evaluate the process flow for service configuration and activation.

The eTOM Level 3 decomposition takes the process called “Service Configuration & Activation” found at Level 2 and breaks it down into the following sub-processes:

- *Design Solution* – Develop an end-end specific service design which complies with a particular customer's requirement
- *Allocate Service Specific Parameters to Services* – Issue service identifiers for new services.
- *Track & Manage Service Provisioning* – Launch all the operational tasks needed to fix each solution requirement.
- *Implement, Configure & Activate Service* – Ensure service provisioning activities are assigned, managed and tracked efficiently.
- *Test Service End-to-End* – Test specific services to ensure all components are operating within normal parameters, and that the service is working to agreed performance levels
- *Issue Service Orders* – Issue correct and complete service orders
- *Report Service Provisioning* – Monitor the status of service orders, provide notifications of any changes and provide management reports.
- *Close Service Order* – Close a service order when the service provisioning activities have been completed
- *Recover Service* – Recover specific services that are no longer required by customers.

These processes are illustrated below in the context of their relationship with the CRM and the RM&O layers. The identified interfaces include: A) Service Activation interface, B) Resource Activation interface, C) Service Inventory interface, and D) Resource Inventory Interface. **It should be emphasized that this particular document only covers Interface A shown in Figure 2-3.**

Note: Not all Layer 3 processes are shown, in particular those that would not be part of the MTOSI-SA interfaces.

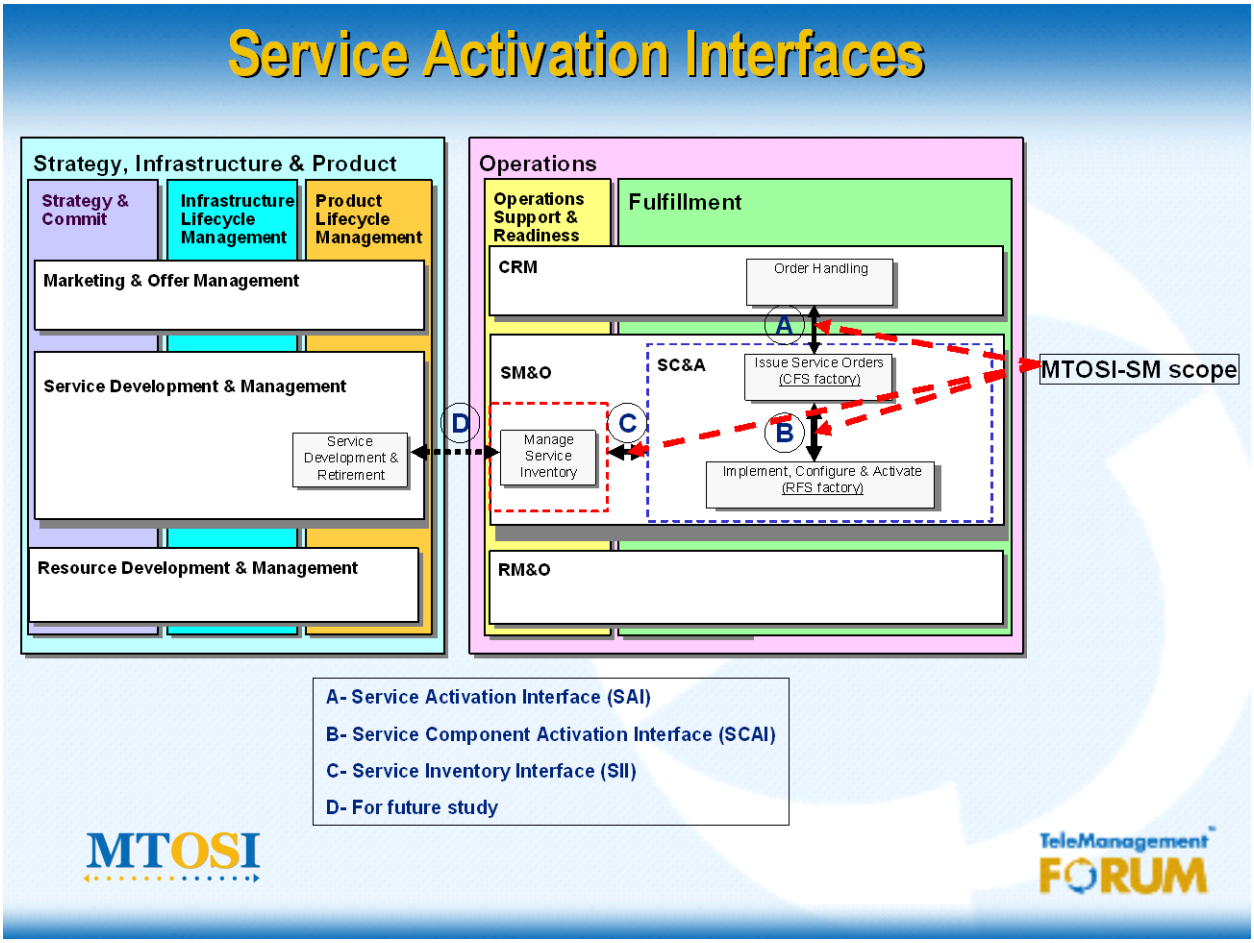

Figure 2-3. eTOM Laye3 Decomposition (Inventory and Activation)

Table **2-1** provides a high-level mapping between the eTOM processes identified earlier in this section and the support of these processes in terms of the operations defined in this document.

Table 2-1. eTOM to SAI Mapping

eTOM L3 Process	**Support by Operations and Notifications in this document**
Design Solution	This process is supported by the design request in R_TMF518_SA_2_II_0031.
Allocate Service Specific Parameters to Services	This process is supported by the design request in R_TMF518_SA_2_II_0031.
Track & Manage Service Provisioning	This process is supported by the provision request and associated responses (i.e., progress events) in R_TMF518_SA_2_II_0033.
Implement, Configure & Activate Service	This process is supported by the design, provision and activate requests in R_TMF518_SA_2_II_0031, R_TMF518_SA_2_II_0033 and

	R_TMF518_SA_2_II_0034, respectively.
Test Service End-to-End	This process is supported by the provision and test requests in R_TMF518_SA_2_II_0033 and R_TMF518_SA_2_II_0040, respectively.
Issue Service Orders	This process is supported by the service order aware version of the interface. Most of the requests in Section 3.3.1.2 have a service aware version which will create a service order as part of their execution.
Report Service Provisioning	This is specifically supported by the progress events defined in R_TMF518_SA_2_II_0033.
Close Service Order	For this interface (when using the service order aware version), the order is closed when the service request is completed. It is also possible to use the stop order (R_TMF518_SA_2_II_0053) and cancel order (R_TMF518_SA_2_II_0054) requests.
Recover Service	This process is support by the terminate (R_TMF518_SA_2_II_0036), retire (R_TMF518_SA_2_II_0037) and cancel (R_TMF518_SA_2_II_0038) requests.

2.5 Assumptions

The following assumptions are made with respect to this document.

2.5.1 Location of Product Decompostion into Services

A key assumption in this document is that the decompostion of a Product into associated Customer Facing Service (CFSs) occurs within the Service Management and Operations (SM&O) / Service Configuration and Activation (SC&A) process. The SAI assumes that the requesting OS supports a CRM application which does not know the relationship between a product instance and its associated CFSs. So, the SAI requests contain product related information and it is expected that the receiving OS takes this information and determines the associated CFSs. In the future, a full service level (in this case, SM&O – SC&A) to service level activation interface may be provided (including support for both CFSs and RFSs). Note that the initial version of the SCAI is a service level to service level interface but does not include support for RFSs

2.5.2 MEP Independence

The operation signatures provided in this document are intended to be independent of the support Message Exchange Pattern. With regard to operations signatures, this document simply provides the request and a list of the events that are generated as a result of the request. Various MEPs can be used to support the operations signatures in this document. The reader should consult *Service Activation - DDP BA - Part 1: Overview* for further details on the set of support MEPs.

2.5.3 Activation Mode

All activation operations over the Interface are supposed to be executed immediately, conditioned only by the availability of processing resources. The only exception that has a deferred implication is the Reserve operation (see additional details in the corresponding section).

2.5.4 Order Aware Version of Interface

It is assumed that all requests using the order aware version of the SAI will necessarily have a corresponding service order. To state this differently, a user of the order aware version of the SAI cannot ask for an associated order to be created in some cases and not in others. This may be a useful option, however, and could be considered in a future version of the SAI.

2.5.5 Single Product

It is currently assumed that the SAI will only support requests for a single product. In cases of bundled products, a service request will be needed for each individual product.

2.5.6 Resource Activation

The SID product model defines a direct relationship between Product and Resource, i.e., the ProductRealizedByResource association. However, the operations in the SAI (e.g., provision) only address the service level requirements for a product instance and not the resource level needs. So, for example, the provision method only provisions the CFSs in support of a given product instance and does not provision the resources directly associated with the product instance. Provisioning and activation of resources directly associated with a product instance are beyond the scope of the SAI and needs to be covered by a yet to be defined resource activation interface.

3 Business Processes

3.1 Business Requirements

The following business requirements apply:

R_TMF518_SA_2_BR_0001	The Interface shall be used between Operations Systems (OSs) for service activation within the same administration (i.e., service provider).
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_BR_0002	The Interface shall be technology and service agnostic and thus allow for implementation by any type of service using any kind of support technology. Example of such services include (amongst others): • Network Transport Services: DSL, MPLS VPN, E-Line, E-LAN, Frame Relay, and ATM • Signaled Services: Voice over IP (VoIP), Voice (Mobile and Fixed), Video on Demand (VoD) • Application Services : Email, SMS, Voice Mail, etc. • Triple Play Services: VoIP, IPTV, and Internet Access.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_BR_0003	The Interface shall support end-to-end, specification based service activation across the CRM and SM&O layers.
Source	TMF518_SA_2, Version 1.0

It should be emphasized that R_TMF518_SA_2_BR_0003 only states that the requesting OS support CRM capabilities (sufficient to construct the SAI requests) and the target OS support SM&O capabilities (sufficient to understand and respond to SAI requests). The requirements does not mean to restrict the type of systems that may offer or use the SAI.

R_TMF518_SA_2_BR_0004	The Interface shall support the activation of atomic services, composite services, and in the future bundled services (as defined in the SID service model).
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_BR_0005	The object model associated with the Interface shall keep traceability with the TM Forum Shared Information/Data (SID)

	model, to the extent possible, without compromising the functionality, efficiency, and flexibility, and performance required.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_BR_0006	The Interface shall support the processing of “service requests” based upon product related information from a CRM layer OS.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_BR_0007	The Interface shall support interactive service requests. In interactive mode, the service request is executed as soon as there are available resources to process the request by the OS receiving the request.
Source	TMF518_SA_2, Version 1.0

3.2 Category I: Static and Structural Requirements

Related static requirements can be found in TMF518_SB, Service Basic - DDP BA.

3.3 Category II: Normal Sequences, Dynamic Requirements

In the following requirements, the follow conventions are used:

- The OS sending a request over the Interface is called the “requesting OS” and the OS receiving the request is called the “target OS”.
- Unless stated otherwise, the term “Interface” refers to the Service Activation Interface (SAI).

3.3.1 Service Activation

It should be noted that the Interface under study in this document, i.e., the Service Activation Interface (SAI), is between a requesting OS that understands the concepts of the eTOM CRM process and a target OS that understands the concepts of the eTOM SM&O layer. The CRM process covers products and the SM&O process covers services. Given the different focus of the requesting and target OS some comprise is needed to allow for communication. There are several possible approaches:

1. Require that the target OS understands the CRM concepts as well as the SM&O concepts. In this case, the interface requests are only related to products and the supporting CFSs are never mentioned.
2. Require that the requesting OS understands the SM&O concepts as well as the CRM concepts. In this case, the interface requests are only related to services (CFSs in particular) and products are never mentioned.

3. Some combination of #1 and #2 above.

The MTOSI Service Management (SM) team decided on Option #3. In particular, the requesting OS makes all of its requests over the Interface in terms of product parameters. The target OS determines the set of one or more CFSs required to fulfill the request. The target OS provides reports on the progress of the supporting CFSs (in terms of state transition notifications). It is left to the requesting OS to determine what this means in terms of the product. The requests over the interface are referred to as ***service requests*** since what is delivered in response to a request pertains to specific services (CFSs in this case). This is a convention and the MTOSI SM team could have just as easily decided to call them product requests since the request are cast in terms of products. One could argue either way here, but in the end, the MTOSI SM team did need to make a decision and choose in favor of the term "service request."

3.3.1.1 General Behavior and State Machine

The requirements that follow are driven by the business requirements with respect to the Interface which are described in R_TMF518_SA_2_II_0008 below. As such the following requirement is most basic, with all other requirements flowing from it.

The concept of Resource Facing Services (RFSs) is used in the definitions below in order to align with the SID terminology. It is important to note, however, that the RFSs are not visible over the SAI and the intention is not to imply a specific implementation in the target OS, i.e., the OS supporting the SM&O process.

The following definitions talk about service orders and RFS orders. A service order pertains to the set of CFSs in support of a given product instance. In the order aware version of the SAI, the service orders are visible. The RFS orders are for the RFSs in support of a given CFS. The RFS orders are not visible over the SAI. The RFS orders are mentioned to describe conceptually what may happen in the target OS when it enters a particular state.

R_TMF518_SA_2_II_0008	With respect to the building and tearing-down of product instances, the Interface shall support the following requests: 1. ***Feasibility check*** for services in support of a given product instance – this request determines if it is feasible to build the CFSs in support of a given product instance. A service order is not associated with this state. When this request is complete, the associated CFS objects are created and it is known whether the service request can be fulfilled. An estimated completion date may also be provided (this is optional). The CFS objects associated with the product instance are created at this point and retrievable over the Interface. It should be emphasized that this is just an initial feasibility check (e.g., to see if a customer's loop is able to support DSL or whether service is even available in the customer's location) in which the underlying RFSs and resources are also checked in harmony with CFS feasibility check. Although nothing is reserved at this point, the target OS may optionally provide an offeredActivationTime which is an estimate of the time it will take to complete the service order if the requesting OS should decide to go forward with the

activation.

2. ***Design*** services for a product instance – this request creates a service order and CFSs in support of a product instance.
   - The service order should indicate that the CFSs are to be transitioned to either the Provisioned_Inactive or Provisioned_Active state.
   - The service order is not yet started at this point.
   - Only service objects are created with no associated reservation.

   When this request is complete, a service order and associated CFS objects have been created.
   - It should be noted that the CFS objects may have already been created during the feasibility check.
   - The service order is for all the CFSs in support of the given product instance. (The target OS may also create RFS orders for the RFSs supporting the CFSs at this point. If RFS are created, they would not be running at this point.)
   - The CFS state transitions are in all cases visible across the Interface. The service order, however, is only visible over the Interface in the order aware case.

   The following items illustrate the need for this particular state:
   - For example, this (state) is useful if a service provider wants to put the CFSs in support of a product instance in a planning state while the customer decides if they want to proceed with an order.
   - As another example, consider a planning OSs which only handles the design of CFSs and then hands-off the order to another OS. So, the first OS can only create the CFS objects and the other OS needs to start from the point where the CFS objects have already been created and then to the associated reservation, provisioning and activation for the CFSs.
3. ***Reservation*** of services for a given product instance – the designed CFS instances, and possibly the resources that the CFS would use through its supporting RFSs that CFS, are reserved for future provisioning in support of a given

	product instance. (Reservation of RFSs and associated resources is an implementation issue.) These resources don't necessarily mean network resources. It is also possible to make a reservation for the termination (release) of the CFSs associated with a product. For example, a customer might want to reserve release of their Internet service three weeks from the present because that is when they plan to move from their current home. Further, the reservation can be for services in association with time-based products where the product is made available only during specified time intervals. The one-time activation and termination reservations are seen a special cases of the general scheduling required for time-based products. See R_TMF518_SA_2_II_0013, R_TMF518_SA_2_II_0014, R_TMF518_SA_2_II_0015 and R_TMF518_SA_2_II_0016 for more details concerning time-based product availability schedules. The reservation may or may not be committed by the requesting OS. For example, the customer may agree to service reservation but (based on service provider policy) the requesting OS may not commit to the reservation until payment is received. In such cases, the requesting OS indicates to the target OS that the reservation is not committed. The target OS will supply the requesting OS with an expiration time (the time by which the reservation needs to be committed; otherwise, the reservation will be deleted at the expiration time). When this request is complete, the requesting OS has confirmation of the requested activation and/or termination date/time(s) for all associated CFSs. 4. ***Provision*** services for a given product instance – this operation entails the completion of a service order to move all the CFSs associated with a product to the Provisioned_Inactive state. Conceptually, this entails the allocation and configuration of all RFSs supporting the CFSs associated with the given product instance. When this request is complete, each CFS (associated with the product instance) shall be in the Provisioned_Inactive state with all associated RFSs (and subtending resources) configured and allocated to the appropriate CFS. At this point, the CFSs have not yet been activated for use by the customer. Note that the term “allocation” is not meant to necessarily imply a one to one mapping between CFS and RFS or between RFS and resource. It is possible for an RFS to be

	allocated to many CFSs (over-booking is also possible). A similar statement can be between about RFSs and resources. 5. ***Activation*** of services for a given product instance – this operation entails the completion of a service order to move all the CFSs associated with a product to the Provisioned_Active state. The specific semantics for "activation" will vary depending on the component services (i.e., the CFSs in support of the product instance). For a transport type service, "activation" may simply be the act of setting an attribute which in turn allows traffic to flow. When this request is complete, the CFSs in support of the product instance should all be ready for use by the customer. 6. ***Deactivation*** of services for a given product instance – this undoes the activation of the product. The CFSs remain provisioned but are inactive in the sense that they cannot be used by the customer. When this request is complete, the component CFSs (i.e., the CFSs in support of the product instance) are no longer available for use by the customer. However, the associated RFSs are still allocated to the CFSs. Possible usage • This function can be used as the first step in deleting a product instance at the CRM layer that is no longer wanted by the customer. • This function can be used to suspend service to a customer (e.g., non-payment of bills) but still leave all the supporting RFSs and associated resources in place. Once the issue with the customer is resolved at the CRM layer, the product instance can be re-activated. 7. ***Termination*** of services associated with a given product instance – this request releases all the RFSs in support of the CFSs for the given product instance. When this request is complete, the RFSs are disassociated from the CFSs and made available for use in other situations. 8. ***Retiring*** of services associated with a given product instance – this entails the deletion of the CFS objects associated with the given product instance. When this request is complete, no trace of the CFSs remains.

Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0009	The service states are as defined in Table **3-1**.
Source	TMF518_SA_2, Version 1.0

In Table **3-1**, note that FeasibilityChecked and Designed are substates of Planning. Provisioned_Inactive and Provisioned_Active, however, are separate states (i.e., not substates of Provisioned).

It is critical to understand that the various operations listed in the Trigger Events of Table **3-1**, apply to all the CFSs associated with a given product instance.

Table 3-1. Service Activation States

State name	State Description	State Characteristics	Trigger Events
Planning – FeasibilityChecked	This planning state designates the initial phase of the planning process. In this state, the planning process determines if the services required to comply with the requested Product are feasible and (potentially) when they can be activated.	In this state: • There is an indication of whether or not the CFSs in support of the given product instance can be activated. • The CFSs associated with the product instance are created at this point and retrievable over the Interface. • The supporting RFS orders **may** be created at this point.	**checkFeasibility** The **design**, **reserve**, **provision** and **activate** requests force a transition through this state if feasibility has not already been checked previously. However, these requests do not leave the associated CFSs in the Planning – FeasibilityChecked state but rather in a latter state, i.e., Planning – Designed, Reserved, Provisioned_Inactive or Provisioned_Active.
Planning – Designed	This planning state designates the phase where a service order is designed and assembled for the CFS instance.	In this state: • The service order corresponding to the request is created and made available for retrieval over the interface. However, the service order is not yet started. • Appropriate RFS orders may be generated for each CFS. If the RFS orders are created, they will not be started at this point. Note that these orders are not visible via the SAI.	**design** The **reserve**, **provision** and **activate** requests force a transition through this state if a service design has not already occurred. However, these requests do not leave the associated CFSs in the Planning – Designed state but rather in a latter state, i.e., Reserved, Provisioned_Inactive or Provisioned_Active.
Reserved	In this state, the designed CFS instance has been reserved for activation at a given date and time, along with the resources that the CFS would use through its supporting RFSs. These resources may or may not be network resources, depending on the service type. It is assume that the	In this state: • The CFS instance, and possibly the resources that would be used through its supporting the RFSs are reserved for future provisioning (or termination) with respect to a given product instance. Depending on the request, the CFSs may be reserved for transition Provisioned_Inactive, Provisioned_Active or	**reserve**

	requesting OS has committed to the reservation. However, the requesting OS does have the option of canceling. Reservation of RFSs and supporting resources is an implementation issue.	Terminated. • It is an implementation decision as to whether or not RFSs and associated resources are reserved at this point. • In the case RFSs orders are created, it is an implementation issue as to whether the orders are started or not at this point in time.	
Provisioned_Inactive	The CFS instance has its associated service order successfully executed, but it hasn't been made available for use by the customer.	In this state: • The state for each associated CFS order is Closed_Completed. • RFSs and associated resource have been allocated to the CFS. At this point, it is simply a matter of activating the CFSs in order to allow for usage by the customer (see the Provisioned_Active state).	**provision** If the CFS has not yet provisioned and an **activate** request is made, then the CFS will first need to transition through the Provisioned_Inactive state before landing in the Provisioned_Active state.
Provisioned_Active	The CFS instance is now available for use by the customer in conjunction with the given product instance.	Same as in the Provisioned_Inactive state, except that the CFS can now be used by the customer.	**activate**
Terminated	The CFS instance is deactivated and the supporting RFSs are deallocated from the CFS.	In this state: • RFSs (and associated resources) have been deallocated from the CFS (with minor exceptions). • The associated RFS Order states are Closed_Completed. Reactivation of the CFS will require the execution of another service order and its subtending RFS orders. • The CFS object still exists in the target OS.	**terminate**

R_TMF518_SA_2_II_0010	The transitions for the states defined in R_TMF518_SA_2_II_0009 are as shown in Table **3-2**. The initial (or “from”) states are listed in the first column and the final (or “to”) states are listed in the top row.
Source	TMF518_SA_2, Version 1.0

Table 3-2. Service State Transitions

	non-existent	**Planning – FeasibilityChecked**	**Planning – Designed**	**Reserved**	**Provisioned_ Inactive**	**Provisioned_ Active**	**Terminated**
non-existent		Feasibility Check	Design [via intervening states]	Reserve [via intervening states]	Provision [via intervening states]	Activate [via intervening states]	
Planning – FeasibilityChecked	Cancel	Amend	Design	Reserve [via intervening states]	Provision [via intervening states]	Activate [via intervening states]	
Planning – Designed	Cancel		Amend	Reserve	Provision [via intervening states]	Activate [via intervening states]	
Reserved	Cancel		Design	Amend [via Design]	Typically done automatically by the target OS at a predetermined time. However, it possible to force a Provision.	Typically done automatically by the target OS at a predetermined time However, it possible to force an Activate.	Typically done automatically by the target OS at a predetermined time. However, it possible to force a Terminate.
Provisioned_Inactive					Modify, Test	Activate	Terminate
Provisioned_Active				In the cases of services with an associated time schedule for availability, the target OS may automatically transition a CFS to the Reserved state during scheduled periods of unavailability.	Deactivate Also, in the cases of services with an associated time schedule for availability, the target OS may automatically transition a CFS to the Provisioned_Inactive state during scheduled periods of unavailability.	Modify, Test	
Terminated	Retire					Activate [via intervening states, typically starting with Planning – Designed]	

As indicated in R_TMF518_SA_2_II_0010, Table **3-2** is considered the normative reference for the SAI service state transitions. However, some readers may prefer to see the transitions in a diagram form (see Figure **3-1**).

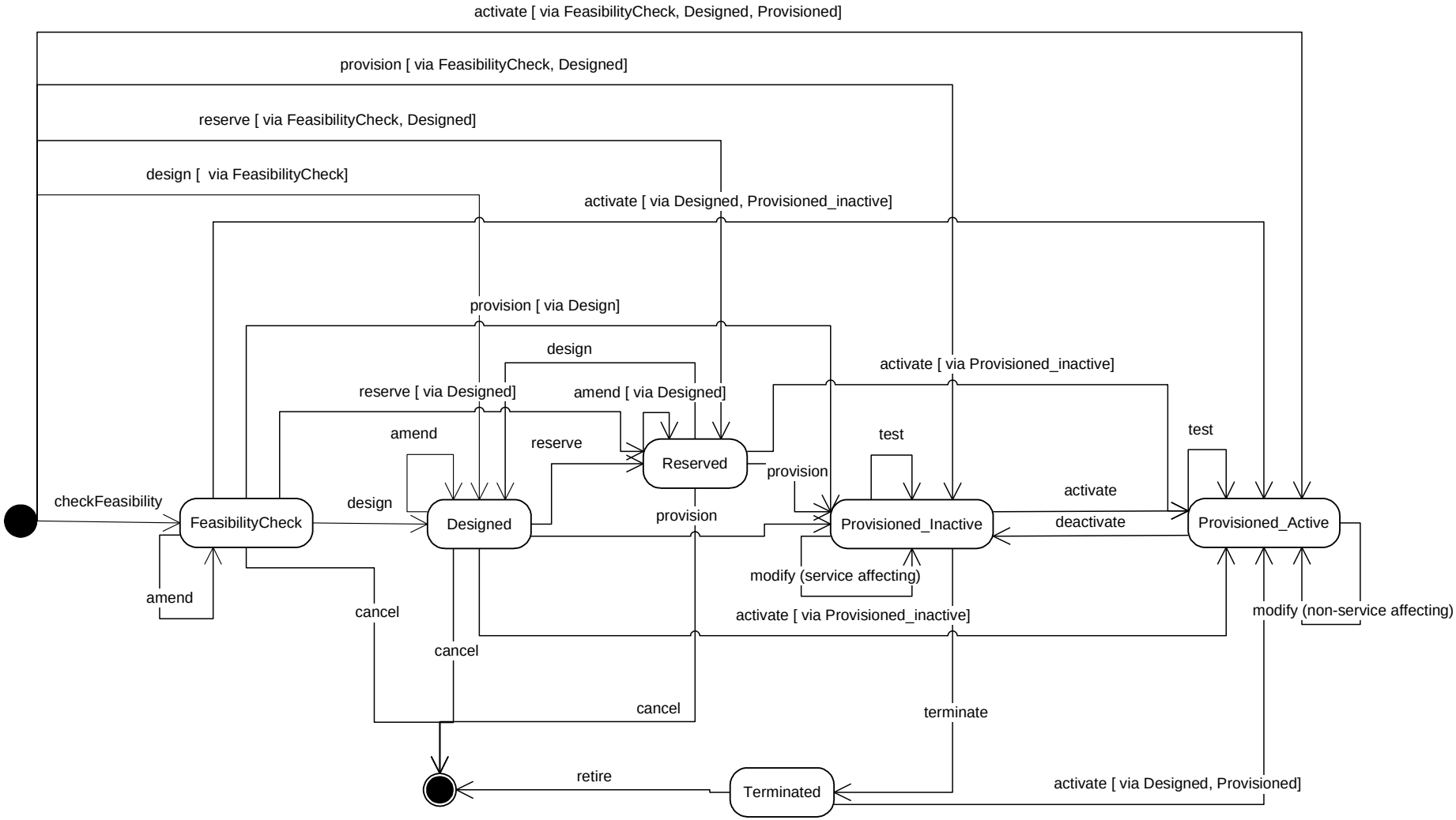

Figure 3-1. SAI Service State Transitions

R_TMF518_SA_2_II_0011	The Interface shall allow the requesting OS to **cancel** the CFSs associated with a given product instance, if all the CFSs are in the Planning (either FeasibilityChecked or Designed) or Reserved state. As a result of the cancel request, all CFS reservations (associated with the give product instance) are cancelled and the CFSs are deleted from the target OS.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0012	The Interface shall allow the requesting OS to **amend** the CFSs associated with a given product instance, if all the CFSs are in either the Planning or Reserved state. After the amend request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

A need has been identified to support products that are either available or unavailable based on a time schedule (see R_TMF518_SA_2_II_0013). This issue arises in cases where the customer does not need the service 24x7.

R_TMF518_SA_2_II_0013	The Interface shall support the activation of **time-based products** where one or more of the associated CFSs are transitioned between Provisioned_Active and Provisioned_Inactive or Reserved based on a predetermine time schedule (defined in R_TMF518_SA_2_II_0015).
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0014	The schedule that is passed over the Interface pertains to the product. It is the job of the target OS (SM&O process) to map the product schedule into an associated schedule for one or more of the associated CFSs. Note that the target OS may be able to support the product schedule by only touching a proper subset of the associated CFSs.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0015	The schedule for time-based products shall have the following characteristics: • The scheduled times for product availability and unavailability shall not overlap. • In the case of regular intervals, the schedule shall have a **start date/time**, **availability duration** (how long the product is to be available before it is made unavailable)

	and **recurrence interval**. In keeping with the previous requirement, the recurrence interval must be longer than the availability duration. Optionally, the requesting OS can provide an **end date/time** for the schedule. • In the case of irregular intervals, the schedule should have a series of **start date/times** and associated **availability durations**. In keeping with the first requirement above, the availability intervals should not overlap. • Overruns shall be supported where an overrun entails one or more extensions to an availability period (typically involving a higher fee). The allowable overrun is represented as a **sequence of extension times**. The specific logic associated with the time extensions is defined as part of the product characteristics and likely to vary depending on the product.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0016	If the target OS cannot fulfill a given request for a time-based product, the target OS shall raise an exception. The exception shall indication the start time for the request can be fulfill, if possible.
Source	TMF518_SA_2, Version 1.0

Examples of periodic and random schedules are shown in Figure **3-2** and Figure **3-3**, respectively. An extension can optionally be added to any of the availability intervals in either a periodic or random schedule.

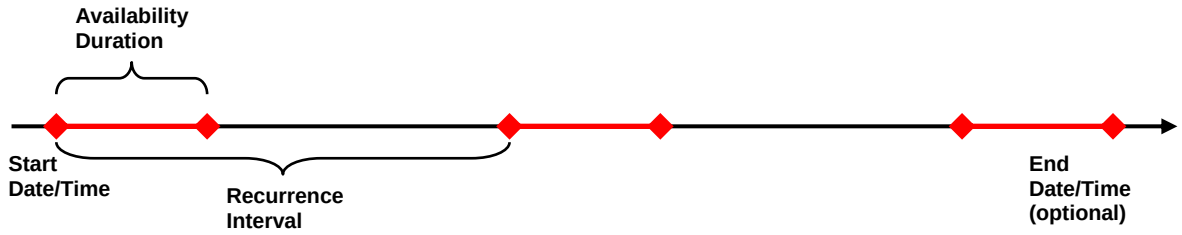

Figure 3-2. Illustration of Periodic Schedule

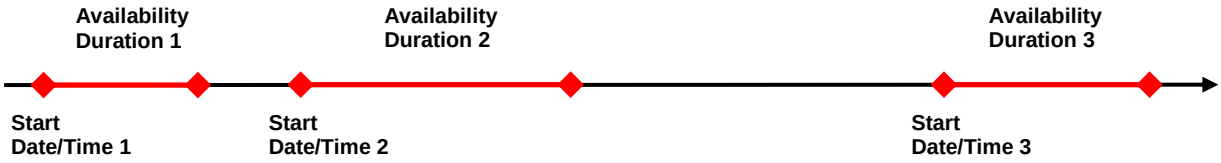

Figure 3-3. Illustration of Random Schedule

R_TMF518_SA_2_II_0017	The Interface shall allow the requesting OS to **modify** the CFSs associated with a given product instance, if all the CFSs are in either the Provisioned_Active or Provisioned_Inactive state. After the modify request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0018	The Interface shall allow the requesting OS to request the **migration** of the current set of CFSs in order to comply with another Product. This is a specific type of the modify request. After the migration request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0019	The Interface shall allow the requesting OS to request that the list of product spec characteristics and product characteristic value (associated with a product instance) be changed. This is a specific type of the modify request. After the change service characteristics request is complete, each associated CFS should be the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0020	The Interface shall allow the requesting OS to request the **addition of new SAPs** to an existing CFS. This is a specific type of the modify request. After the add SAPs request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0021	The Interface shall allow the requesting OS to request the **removal of SAPs** from an existing CFS. This is a specific type of the modify request. After the remove SAPs request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0022	The Interface shall allow the requesting OS to request the **replacement of SAPs** in an existing CFS. This is a specific type of the modify request. After the replace SAPs request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0023	The Interface shall allow the requesting OS to request the **addition of Subscribers or/and Users** to an existing CFS. This is a specific type of the modify request. After the add parties request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0024	The Interface shall allow the requesting OS to request the **removal of Subscribers or/and Users** from an existing CFS. This is a specific type of the modify request. After the remove parties request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0025	The Interface shall allow the requesting OS to request the **replacement of Subscribers or/and Users** in an existing CFS. This is a specific type of the modify request. After the replace parties request is complete, each CFS is in the same state as before the request.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0026	The Interface shall allow the requesting OS to **test** the CFSs associated with a given product instance, if all the CFSs are in either the Provisioned_Active or Provisioned_Inactive state. The test will indicate whether or not the CFSs are able to provide service to the customer. After the test request is complete, each CFS is in the same state as before the request. **Note:** There is a testing task during the provisioning process and this is automatically executed before the CFSs reach the Provisioned state. By contrast, after the CFSs have been successfully activated, at any time, the CRM may invoke a test

	operation to insure the CFSs are still working properly. So, in the first case, there is no need to invoke an explicit test operation, while in the last case there is.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0027	The Interface shall support requests to retrieve the state of all the CFSs associated with a given product instance.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0028	It shall be possible to supress the generation of CFS state change notifications at the source (Target OS). **Note**: The specific supression mechanism is currently outside the scope of this specification and would need to be done via some form of implemetation agreement between the involved parties.
Source	TMF518_SA_2, Version 1.0

3.3.1.2 Detailed Behavior

This section provides detailed operations signatures for the various SAI operations. The general pattern is a single request, followed by an initial response and then a series of events that report on the progress of the request. It should be noted that these definitions are not intended to imply a specific Message Exchange Pattern (MEP). The term “event” is used in a very general sense here (just a progress report on the request) and it does not imply a notification in the MTOSI sense. Further, the events defined below may also contain exceptions.

In service activation “phase”, there is a common pattern in the operations signatures below. The target OS reports on the CFS as they are created (perhaps in a state “before” the desired end state) and then provides CFS state change updates as the CFS is moved towards the desired state.

- It is an implementation decision as to whether the target OS will report all the intermediate transitions.
- It is a requirement, however, that the target OS report on the CFS when it enters the desired end state (see R_TMF518_SA_2_II_0029).
- In some cases, the target OS will need to report on at least some intermediate state transitions. For example, consider the case of multiple CFSs supporting a product where, for some reason, one CFS reaches the desired end state but some other CFSs get stuck in different intermediate states. How this scenario will affect the product state can only be defined by the CRM process (as supported by the requesting OS). Therefore, in this case, these intermediate state transitions, need to be reported by the target OS.

R_TMF518_SA_2_II_0029	Concerning service requests, the target OS must report when a CFS associated with the given product instance enters the requested end state.

Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0030	The **FeasibilityCheck** request shall convey the information and support the behavior described in Table **3-3**.
Source	TMF518_SA_2, Version 1.0

The SAI feasibilityCheck operation is for services, SAPs, etc. associated with a given product instance. The granularity of the SAI is at the product level and it (the SAI) does not allow one to just check for feasibility concerning an individual CFS or SAP. The SCAI could be extended at some point to do a feasibility check on a given CFS or RFS. Feasibility check on a given SAP (independent of a service) is a resource level operation.

Table 3-3. Operation Signature for FeasibilityCheck

Operation Signature	Request: ***feasibilityCheck*** (productInfo, subscriberList, userList, sapList) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If “yes”, the request is placed in a queue. Events: 1. ***beginProcessingEvent*** (serviceRequestRef, soAwareParamsRes) – this is an indication that the target OS has removed the request from the queue and has started to work on the request. For feasibilityCheck, the soAwareParamsRes is not populated. 2. ***cfsCreationEvent*** (serviceRequestRef, List Of Cfs, last) – this event is an indication that one or more CFSs (supporting the requested product) have been created and are in the Feasibility Checked state. • In the case of multiple CFSs corresponding to a single product instance, it may be necessary to send several cfsCreationEvents. Also, it is possible to report on several CFSs in one event.
Behavior	This operation verifies the feasibility of provisioning and activating the CFSs in support of a product instance. If the operation is successful, CFS objects are created and placed in the FeasibilityChecked state and the requesting OS is informed that its request is feasible, but nothing is reserved.
Pre-conditions	The CFSs supporting the product instance do not yet exist.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the FeasibilityChecked state.

	In case of failure: • In the case of an atomic request, all traces of the CFSs have been deleted. • Support for “best effort” is not recommended for this operation since this operation is fundamentally an atomic request.
Arguments	
Request	productInfo – this is the product related information supplied by the • productName – this is a unique name for the product instance. • productSpecificationName (optional) – this identifies the product specification associated with the product instance. This is used by the target OS to determine a set of one or more service templates which reference the globally set serviceSpecCharacteristics. • productBundleName (optional) – this is used in cases where the request refers to a bundle of products. It should be emphasized that the requesting OS needs to send a separate request for each product instance in a product bundle. The productBundleName is only included to provide additional context and there is no expectation that the target OS will decompose the identified product bundle into its component products. • List Of (productSpecCharacteristicRef, productCharacteristicValue) is used to convey the individually set product characteristics (i.e., those not referenced by the productSpecificationName noted above) and those globally set characteristic values that are allowed to be overridden where ○ productSpecCharacteristicRef (optional) – a unique identifier for a product specification characteristic. A product specification characteristic is a quality or distinctive feature of a ProductSpecification. ○ productCharacteristicValue (optional) – a number or text that can be assigned to an instance of a product. The target OS will map the productSpecCharacteristics to serviceSpecCharacteristics where the mapping may **not** be one-to-one. Further, ProductSpecCharValue need not be passed over the interface since it assumed the target OS can determine the associated ServiceSpecCharacteristic and ServiceSpecCharValues from ProductSpecCharacteristic alone. An example of the above terms is as follows: • ProductSpecCharacteristic – color • ProductSpecCharValue – red, silver, grey, blue, magenta, black. • ProductCharacteristicValue – black (the color a particular customer chose for his or her cell phone) **Note** that there are two options here. However, in both cases, the target OS (SM&O process) maps the product information to the appropriate service lever information. • In one case, the requesting OS (CRM process) supplies all the needed product information, i.e., productSpecificationName, productBundleName and List Of (productSpecCharacteristicRef, productCharacteristicValue).

	• In the second case, the requesting OS only supplies the productName and leaves it to the target OS to retrieve the additional product information for a product inventory OS. For this current release of the SAI, only the first option is possible since a product inventory retrieval interface has not yet been defined. subscriberList (optional) – this parameter provides a list of the subscribers associated with the request. The subscriberList applies to the given product instance as opposed to being applied to a subset of the CFSs associated with the product instance. userList (optional) – this parameter provides a list of the users associated with the request. It is assumed that all on the userList apply to the given product instance as opposed to being applied to a subset of the CFSs associated with the product instance. sapList (optional) – this parameter provides a list of the SAPs associated with the request.
Response	serviceRequestRef – this is the name that the target OS assigns to the request. In the IIS, this is handled in the message header. list of Cfs – this is a list of the CFSs that have been created or updated as a result of the request. accept – this is an indication (“yes” or “no”) that the target OS has accepted the request and has put the request in a queue (awaiting processing) cfs – depending on a agreement between the supplier and user of this interface, this is either just the name of the CFS or the entire structure. last (optional) – this Boolean indicates whether or not the target OS has sent the last event in response to the request. In some cases, not all the CFSs will be in the desired end state when the target OS decides to stop work on the request. In such cases, the target OS will typically send an exception to the requesting OS.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).

R_TMF518_SA_2_II_0031	The **design** request shall convey the information and support the behavior described in Table **3-4**.
Source	TMF518_SA_2, Version 1.0

Table 3-4. Operation Signature for Design

Operation Signature	Request: ***design*** (productInfo, subscriberList, userList, sapList, soAwareParamsReq) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** (serviceRequestRef, soAwareParamsRes) – this is an indication that the target OS has removed the request from the queue and has started to work on the request. In the case of the order aware interface, this event returns the service order ID and an initial value for the expectedCompletionDate (which could change as the order is processed). 2. ***cfsCreationEvent*** (serviceRequestRef, List Of Cfs, last) – this event is an indication that one or more CFSs (supporting the requested product) has been created and is in the Feasibility Checked or Designed state. • In the cases of multiple CFSs corresponding to a single product instance, it may be necessary to send several cfsCreationEvents. Also, it is possible to report on several CFSs in one event. • For CFSs that are created in the Designed state, no further events will follow with respect to this request. 3. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CfsNewState), last) – this event is an indication that the target OS has progressed one or more CFSs from the Feasibility Checked to the Designed state. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance. Note that the "last" parameter is set to true in the event that reports on the last CFS being transitioned to the Designed state. This could be either a cfsCreationEvent or a cfsStateChangeEvent, but not both for the same request.
Behavior	This task designs the CFSs, RFSs and underlying resources in support of a product instance. If feasibility has not already been checked, it will be checked as part of this operation. The target OS will create, but not start, a service order for the overall request. The target OS will also create, but not start, RFSs orders for the CFSs associated with the product. It should be emphasized that only the "overall" service order (but not the RFS orders) can be retrieved over the SAI.
Pre-conditions	The CFSs supporting the product instance either do not yet exist or are in the FeasibilityChecked or Designed state.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the Designed state. In case of failure:

	• In the case of an atomic request, all CFS(s) have been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	For the definitions of productInfo, subscriberList, userList and sapList see Table **3-3**. soAwareParamsReq (optional) – the following parameters are sent in the service order aware version of the interface: • requestedCompletionDate – the date and time at which the requesting OS has requested that execution of the service order be complete. The target OS will take this as input to its scheduling process. The target OS will provide the requesting OS with an expectedCompletionDate which indicates when the target OS thinks it can complete the execution of the service order. The target OS is allowed to complete execution of the service order before the requestedCompletionDate. This parameter is only used in the order aware version of the interface. • priority – this attribute allows the request to specify a priority for execution of the service request. This attribute is a positive integer. • purchaseOrder – identifies the associated customer order. • validFor – the date and time for which this request is valid. If the request is not fulfilled by the provided date and time, the service order process will automatically be aborted by the target OS.
Response	serviceRequestRef – this is the name that the target OS assigns to the request. As noted previous and not to be repeat again, in the IIS, this is handled in the message header. accept – this is an indication (“yes” or “no”) that the target OS has accepted the request and has put the request in a queue (awaiting processing) cfsName – the name of the CFS to which the progress report applies. cfsOldState – the state from which the CFS has transitioned. cfsNewState – the current state of the CFS to which the progress report applies. last – this Boolean indicates whether or not the target OS has sent the last event in response to the request. soAwareParamsRes (optional) – the following parameters are returned in the service order aware version of the interface: • serviceOrderRef – this is the name that the target OS assigns to the service order associated with the request. This parameter is only used when the target OS supports the “order aware” version of the SAI. • expectedCompletionDate – the date and time at which the target OS expects to complete execution of the service order. The target OS will update this parameter as the situation changes (such changes will be communicated to the requesting OS via AVC notifications on the given

	service order).
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).

R_TMF518_SA_2_II_0032	The **reserve** request shall convey the information and support the behavior described in Table **3-5**.
Source	TMF518_SA_2, Version 1.0

Table 3-5. Operation Signature for Reserve

Operation Signature	Request: ***reserve*** (productInfo, subscriberList, userList, sapList, expiringTime, productAvailabilitySchedule, commit, desiredState, soAwareParamsReq) ***amend*** (serviceRequestRef) – this operation is used by the requesting OS to confirm that it wants to go ahead with the reservation. It is only used when "commit" is set to false in the initial reserve request. If a commit is not made in either the initial reserve request or a subsequent amend request before the expiringTime, the target OS will cancel the reserve request. Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. 2. ***AmendRes*** – this is a positive or negative acknowledgment with no parameters. In the case, of a negative acknowledgment, a reason is also given, e.g., "expiringTime has passed". Events (these only apply to the reserve request): 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsCreationEvent*** (serviceRequestRef, List Of Cfs, last) – this event is an indication that one or more CFSs (supporting the requested product) has been created and is in the Feasibility Checked, Designed or Reserved state. • In the cases of multiple CFSs corresponding to a single product instance, it may be necessary to send several cfsCreationEvents. Also, it is possible to report on several CFSs in one event. • For CFSs that are created in the Reserved state, no further events will follow with respect to this request. 3. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CFSNewState), last) – this event is an indication that the target OS has progressed one or more CFSs to the Reserved state. In the service order aware version of the interface, a Service Order will be

	assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance. Note that the "last" parameter is set to true in the event that reports on the last CFS being transitioned to the Reserved state. This could be either a cfsCreationEvent or a cfsStateChangeEvent, but not both for the same request.
Behavior	This operation reserves the CFSs in support of a product instance. If feasibility check and design have not yet been done for the CFSs supporting the given product instance, both will be done as part of this operation. The target OS will not go forward with the reservation unless and until the requesting OS commits. If the commit is not indicated in the initial request, the requesting OS will need to commit via the amend operation before the expiring time. When this operation is complete, the requesting OS confirmation of their requested activation time and/or termination time. Alternately, if the target OS cannot meet the requested time(s), it will reject the request. The possibility of negotiation concerning activation/termination times is for further study.
Pre-conditions	The CFSs supporting the product instance either do not yet exist or are in the FeasibilityChecked, Designed or Reserved state.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the Reserved state. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	For the definitions of productInfo, subscriberList, userList and sapList see Table **3-3**. The soAwareParamsReq are defined in Table **3-4**. However, requestedCompletionDate is not used here. expiringTime (optional) – the date and time by which an uncommitted reservation will be cancelled. A committed reservation request always contains an empty expiringTime value. productAvailabilitySchedule – this is the schedule for when the product shall be made available. The target OS (SM&O) process needs to map this schedule into a transition schedule for one or more of the associated CFSs. See R_TMF518_SA_2_II_0013, R_TMF518_SA_2_II_0014, R_TMF518_SA_2_II_0015 and R_TMF518_SA_2_II_0016 for more details. commit – this parameter indicates whether or not the requesting OS has committed to the reservation. It is a Boolean. If the requesting OS does not commit in the initial request, it must commit to the reservation using the amend operation

	before the expiringTime; otherwise, the target OS will cancel the request. desiredState – the desired initial transition when the reservation period is over. There are two choices, i.e., Provisioned_Inactive or Provisioned_Active.
Response	For the definition of serviceRequestRef, accept, cfsName, cfsOldState, cfsNewState, soAwareParamsRes and last see Table **3-4**. However, expectedCompletionDate is not used here.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064). All of the above exceptions apply to the reserve request but only the basicFailureEvent applies to the amend request.

R_TMF518_SA_2_II_0033	The **provision** request shall convey the information and support the behavior described in Table **3-6**.
Source	TMF518_SA_2, Version 1.0

Table 3-6. Operation Signature for Provision

Operation Signature	Request: ***provision*** (productInfo, subscriberList, userList, sapList, soAwareParamsReq) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsCreationEvent*** (serviceRequestRef, List Of Cfs, last) – this event is an indication that one or more CFSs (supporting the requested product) has been created and is in the Feasibility Checked, Designed, Reserved or Provisioned_Inactive state. • In the cases of multiple CFSs corresponding to a single product instance, it may be necessary to send several cfsCreationEvents. Also, it is possible to report on several CFSs in one event. • For CFSs that are created in the Provisioned_Inactive state, no further events will follow with respect to this request. 3. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CFSNewState), last) – this event is an indication that the target OS has progressed one or more CFSs to the Provisioned_Inactive state. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance. Note that the "last" parameter is set to true in the event that reports on the last CFS being transitioned to the Provisioned_Inactive state. This could be either a cfsCreationEvent or a cfsStateChangeEvent, but not both for the same request.
Behavior	This operation provisions the CFSs in support of a product instance. If feasibility check and design have not yet been done for the CFSs supporting the given product instance, both will be done as part of this operation. When this request is complete, each CFS (associated with the product instance) shall be in the Provisioned_Inactive state with all associated RFSs (and subtending resources) configured and allocated to the appropriate CFS. At this point, the CFSs have not yet been activated for use by the customer.
Pre-conditions	The CFSs supporting the product instance do not yet exist or are in the FeasibilityChecked, Designed, Reserved (either Reserved-Committed or Reserved-Uncommitted) or Provisioned_Inactive state.

Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the Provisioned_Inactive state. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	For the definitions of productSpecificationName, productBundleName, subscriberList, userList, sapList and productCharacteristics see Table **3-3**. The soAwareParamsReq are defined in Table **3-4**.
Response	For the definition of serviceRequestRef, accept, cfsName, cfsOldState, cfsNewState, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).

R_TMF518_SA_2_II_0034	The **activate** request shall convey the information and support the behavior described in **Table 3-7**
Source	TMF518_SA_2, Version 1.0

Table 3-7. Operation Signature for Activate

Operation Signature	Request: ***activate*** (productInfo, subscriberList, userList, sapList, cfsCurrentState, soAwareParamsReq) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If “yes”, the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsCreationEvent*** (serviceRequestRef, List Of Cfs, last) – this event is an indication that one or more CFSs (supporting the requested product) has been created and is in the Feasibility Checked, Designed, Reserved, Provisioned_Inactive or Provisioned_Active state.

	• In the cases of multiple CFSs corresponding to a single product instance, it may be necessary to send several cfsCreationEvents. Also, it is possible to report on several CFSs in one event. • For CFSs that are created in the Provisioned_Active state, no further events will follow with respect to this request. 3. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CFSNewState), last) – this event is an indication that the target OS has progressed one or more CFSs to the Provisioned_Active state. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance. Note that the "last" parameter is set to true in the event that reports on the last CFS being transitioned to the Provisioned_Active state. This could be either a cfsCreationEvent or a cfsStateChangeEvent, but not both for the same request.
Behavior	This operation activates the CFSs in support of a product instance. If feasibility check, design and provisioning have not yet been done for the CFSs supporting the given product instance, all will be done as part of this operation. When this request is complete, each CFS (associated with the product instance) shall be in the Provisioned_Active state with all associated RFSs (and subtending resources) configured and allocated to the appropriate CFS. At this point, the CFSs have been activated for use by the customer.
Pre-conditions	The CFSs supporting the product instance either do not yet exist or are in the FeasibilityChecked, Designed, Reserved (either Reserved-Committed or Reserved-Uncommitted), Provisioned_Inactive, Provisioned_Active state or Terminated.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the Provisioned_Active state. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	For the definitions of productInfo, subscriberList, userList and sapList see Table **3-3**. The soAwareParamsReq are defined in Table **3-4**. cfsCurrentState –the target OS should check that each CFS is in the CFS state indicated in this parameter before performing the request. If not, an exception should be raised. This parameter can be used (for example) when the requesting OS only wants a transition to Provisioned_Active given the current state is Provisioned_Inactive (basically a resume operation).
Response	For the definition of serviceRequestRef, accept, cfsName, cfsOldState,

	cfsNewState, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).

R_TMF518_SA_2_II_0035	The **deactivate** request shall convey the information and support the behavior described in Table **3-8**.
Source	TMF518_SA_2, Version 1.0

Table 3-8. Operation Signature for Deactivate

Operation Signature	Request: ***deactivate*** (soAwareParamsReq) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CFSNewState), last) – this event is an indication that the target OS has progressed one or more CFSs to the Provisioned_Inactive state. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance.
Behavior	This operation deactivates the CFSs in support of a product instance. It is does the opposite of the activate operation. When this request is complete, each CFS (associated with the product instance) shall be in the Provisioned_Inactive state with all associated RFSs (and subtending resources) configured and allocated to the appropriate CFS. At this point, the CFSs are not activated for use by the customer.
Pre-conditions	The CFSs supporting the product instance should be in the Provisioned_Active state.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the Provisioned_Inactive state. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to

	their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	The soAwareParamsReq are defined in Table **3-4**.
Response	For the definition of serviceRequestRef, accept, cfsName, cfsOldState, cfsNewState, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062).

R_TMF518_SA_2_II_0036	The **terminate** request shall convey the information and support the behavior described in Table **3-9**.
Source	TMF518_SA_2, Version 1.0

Table 3-9. Operation Signature for Terminate

Operation Signature	Request: ***terminate*** (soAwareParamsReq) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CFSNewState), last) – this event is an indication that the target OS has progressed one or more CFSs to the Terminated state. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance.
Behavior	This operation terminates the CFSs in support of a product instance. When this request is complete, each CFS (associated with the product instance) shall be in the Terminated state. The RFSs (that once supported the CFSs) are disassociated from the CFSs and may be made available for use in other situations.
Pre-conditions	The CFSs supporting the product instance should be in the Provisioned_Inactive state.

Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the Terminated state. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	The soAwareParamsReq are defined in Table **3-4**.
Response	For the definition of serviceRequestRef, accept, cfsName, cfsOldState, cfsNewState, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062).

R_TMF518_SA_2_II_0037	The **retire** request shall convey the information and support the behavior described in Table **3-10**.
Source	TMF518_SA_2, Version 1.0

Table 3-10. Operation Signature for Retire

Operation Signature	Request: ***retire*** (soAwareParamsReq) Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsDeletionEvent*** (serviceRequestRef, List Of CfsName, last) – this event is an indication that the target OS has deleted one or more CFSs. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance.

Behavior	This operation deletes the CFS objects related to a product instance. When this request is complete, all the CFS objects that where once associated with the product instance are deleted from the target OS.
Pre-conditions	The CFSs supporting the product instance should be in the Terminated state.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all deleted. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	The soAwareParamsReq are defined in Table **3-4**.
Response	For the definition of serviceRequestRef, accept, cfsName, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceDeletionFailureEvent (see R_TMF518_SA_2_III_0065).

R_TMF518_SA_2_II_0038	The **cancel** request shall convey the information and support the behavior described in Table **3-11**.
Source	TMF518_SA_2, Version 1.0

Table 3-11. Operation Signature for Cancel

Operation Signature	Request: ***cancel*** (soAwareParamsReq) – this operation is very similar to *retire*, except cancel is used when no RFSs have been allocated to the CFSs in support of a product instance. Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsDeletionEvent*** (serviceRequestRef, List Of CfsName, last) – this event is an indication that the target OS has deleted one or more CFSs. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance.
Behavior	This operation cancels the CFS objects related to a product instance. When this request is complete, all the CFS objects that where once associated with the product instance are deleted from the target OS.
Pre-conditions	The CFSs supporting the product instance should be in the Feasibility Checked, Designed or Reserved state.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all deleted. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	The soAwareParamsReq are defined in Table **3-4**.
Response	For the definition of serviceRequestRef, accept, cfsName, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceDeletionFailureEvent (see R_TMF518_SA_2_III_0065).

The various modify requirements in Section 3.3.1.1 (i.e., R_TMF518_SA_2_II_0017, R_TMF518_SA_2_II_0018, R_TMF518_SA_2_II_0019, R_TMF518_SA_2_II_0020,

R_TMF518_SA_2_II_0021, R_TMF518_SA_2_II_0022, R_TMF518_SA_2_II_0025, R_TMF518_SA_2_II_0024 and R_TMF518_SA_2_II_0025) are covered by a single modify request.

It should be mentioned that "modify" could be accomplished by a delete or cancel followed by (for example) activate or provision. The problem with this approach is that the target OS may very well release all the entities (e.g., CFSs, RFSs, and resources) that were supporting the product instance while the requesting OS may only want to change the existing CFSs in support of a give product instance. So, the delete and recreate approach is not efficient. Thus, a modify operation is defined. This operation tells the target OS to move the CFS in support of a given product instance from one "state" to another where "state" refers to not only the CFS state but also product specification characteristics, product characteristic values, SAPs, users and subscribers. Further, note that the product specification characteristics and product characteristic values give rise to associated service specification characteristics, service characteristic values and CFSs (this is done via a predetermined mapping done outside of the SAI).

R_TMF518_SA_2_II_0039	The **modify** request shall convey the information and support the behavior described in Table **3-1**.
Source	TMF518_SA_2, Version 1.0

Table 3-12. Operation Signature for Modify

Operation Signature	Request: ***modify*** (productInfo, subscriberList, userList, sapList, targetState, soAwareParamsReq) – this operation takes the CFSs, subscriber, users and SAPs in support of a given (existing) product instance and makes modifications to support the new productInfo, subscribers, users and SAPs listed in the modify request. Response: 1. ***Response*** (serviceRequestRef, accept) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue. Events: 1. ***beginProcessingEvent*** – see description in Table **3-4**. 2. ***cfsCreationEvent*** (serviceRequestRef, List Of Cfs, last) – this event is an indication that one or morenew CFSs (supporting the requested modification) has been created placed in the given targetState (either Provisioned_Inactive or Provisioned_Active). • In the cases of multiple CFSs corresponding to a single product instance, it may be necessary to send several cfsCreationEvents. Also, it is possible to report on several CFSs in one event. • For CFSs that are created in the Provisioned_Active state, no further events will follow with respect to this request. 3. ***cfsDeletionEvent*** (serviceRequestRef, List Of CfsName, last) – this event is an indication that the target OS has deleted one or more CFSs. In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by

	the same Service Order instance. 4. ***cfsStateChangeEvent*** (serviceRequestRef, List Of (CfsName, CfsOldState, CFSNewState), last) – this event is an indication that the target OS has progressed one or more of the existing CFSs to the target State (either Provisioned_Inactive or Provisioned_Active). In the service order aware version of the interface, a Service Order will be assembled. For each CFS, a service order item will be created and associated with the service order. It is assumed all the CFSs are driven by the same Service Order instance. 3. ***cfsAvcEvent*** (serviceRequestRef, List Of (Attribute, AttributeValue, last) – this event is an indication a set of attribute values have been changed, added or deleted for a give set of CFSs. Note that the "last" parameter is set to true in the event that reports on the last CFS being transitioned to the targetState (either Provisioned_Inactive or Provisioned_Active). This could be either a cfsCreationEvent or a cfsStateChangeEvent, but not both for the same request.
Behavior	This operation modifies the CFSs, subscribers, users and SAPs in support of a product instance. If feasibility check or design have not yet been done for a CFSs supporting the given product instance, this will be done as part of this operation. When this request is complete, each CFS (associated with the product instance) shall be in the targetState (either Provisioned_Inactive or Provisioned_Active) with all associated RFSs (and subtending resources) configured and allocated to the appropriate CFS.
Pre-conditions	The CFSs supporting the product instance either do not yet exist or are in the FeasibilityChecked, Designed, Reserved (either Reserved-Committed or Reserved-Uncommitted), Provisioned_Inactive, Provisioned_Active state or Terminated.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are all in the targetState (either Provisioned_Inactive or Provisioned_Active). Subscribers, users and SAPs will be added or deleted as indicated in the modify request. In case of failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. Also, no subscriber, users or SAPs will be added or deleted. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Arguments	
Request	For the definitions of productInfo, subscriberList, userList and sapList see Table **3-3**. The soAwareParamsReq are defined in Table **3-4**. targetState – this is the state to which the target OS wants the CFSs transitioned when the request is complete (the value is either Provisioned_Inactive or Provisioned_Active).

Response	For the definition of serviceRequestRef, accept, cfsName, cfsOldState, cfsNewState, soAwareParamsRes and last see Table **3-4**.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceAvcFailureEvent (see R_TMF518_SA_2_III_0063) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064) • serviceDeletionFailureEvent (see R_TMF518_SA_2_III_0065).

R_TMF518_SA_2_II_0040	The **test** request shall allow the requesting OS to ask whether the CFSs and SAPs for a given product instance are able to provide service to the customer.
Source	TMF518_SA_2, Version 1.0

Table 3-13. Operation Signature for Test

Operation Signature	Request: ***test*** (productName) – this operation tells whether the CFSs and SAPs supporting a given product instance are able to provide service to a customer. Response: 1. ***Response*** (serviceRequestRef, testResult) – this provides an indication of whether or not the target OS has accepted the request. If "yes", the request is placed in a queue.
Behavior	For an existing product instance, the target OS determines whether the CFSs and SAPs are currently able to support service. For this version of the operation, the response will be very simple (just success or failure). In future releases, the operation may be defined to provide more detailed responses that indicate what CFSs and/or SAPs have failed and the reason for failure.
Pre-conditions	The CFSs supporting the product instance are either in the Provisioned_Inactive or Provisioned_Active state.
Post-conditions	In case of success: • The requesting OS has received an indication as to whether the test was successful or not. It is important to note that "success" means the test was performed whether or not the CFSs and SAPs are found to be capable of supporting service. In case of failure: • The target OS was not able to perform and test, and an exception is returned to the requesting OS. "Failure" means that the target OS could not perform the test.

Arguments	
Request	productName – identifies the product instance to be tested.
Response	testResult – this is a Boolean that tells whether the test was successful or not.
Exceptions	The following exceptions are allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061)

R_TMF518_SA_2_II_0041	The retrieveServiceStates request shall convey the information and support the behavior described in Table **3-14**.
Source	TMF518_SA_2, Version 1.0

Table 3-14. Operation Signature for retrieveServiceStates

Operation Signature	Request: **retrieveServiceStates** (productName) – this operation returns the state for each of the CFSs associated with a given product instance. Events: 1. ***retrieveServiceStatesResponse*** (List Of (cfsName, cfsState)) – this is a list of (cfsName, cfsState) pairs.
Behavior	This operation returns the name and state for each CFS associated with a given product instance. This would typically be used in lieu of CFS events for creation, state change and deletion.
Pre-conditions	The product instance (identified by productName) must already exist.
Post-conditions	In case of success: • The CFS(s) associated with the product instance are unchanged. • The requesting OS has the information that it requested. In case of failure: • The CFS(s) associated with the product instance are unchanged. • The requesting OS does not have the information that it requested.
Arguments	
Request	productName – the unique name of the product instance for which the requesting OS wants the states of the associated CFSs.
Response	ListOf (cfsName, cfsState) – the name and current state for each CFS associated with the given product instance.
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

3.3.2 Service Ordering

In addition to the service activation capabilities described in Section 3.3.1, it is possible to define a service order for each activation request. The service order provides additional control and information to the requesting OS with respect to a given service activation request.

3.3.2.1 General Behavior and State Machine

R_TMF518_SA_2_II_0042	The Interface shall support the creation of a service order in association with a service activation request. It should be emphasized that the service activation request is related to a given product instance. Although several CFSs are likely to be required for a given product instance, there is only one service order created as a result of service activation request over the Interface.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0043	A service order shall be in one of the following states for a given point in time:
	CLOSED
	• COMPLETED
	• ABORTED
	  ○ ABORTED_BYCLIENT
	  ○ ABORTED_BYSERVER
	OPEN
	• NOT_RUNNING
	  ○ SUSPENDED
	  ▪ SUSPENDED_BYSERVER
	  • AWAITING_INPUT – this state is used when the target OS (i.e., the OS that is attempting to fulfill the service order) cannot proceed with further external input. This state is also used when the target OS is only able to partially complete the order and needs external assistance to complete the order.
	  • AWAITING_VALIDATION
	  ▪ SUSPENDED_BYCLIENT

	○ NOT_STARTED • RUNNING
Source	TMF518_SA_2, Version 1.0

The service order states may also be depicted as shown in Figure **3-4**.

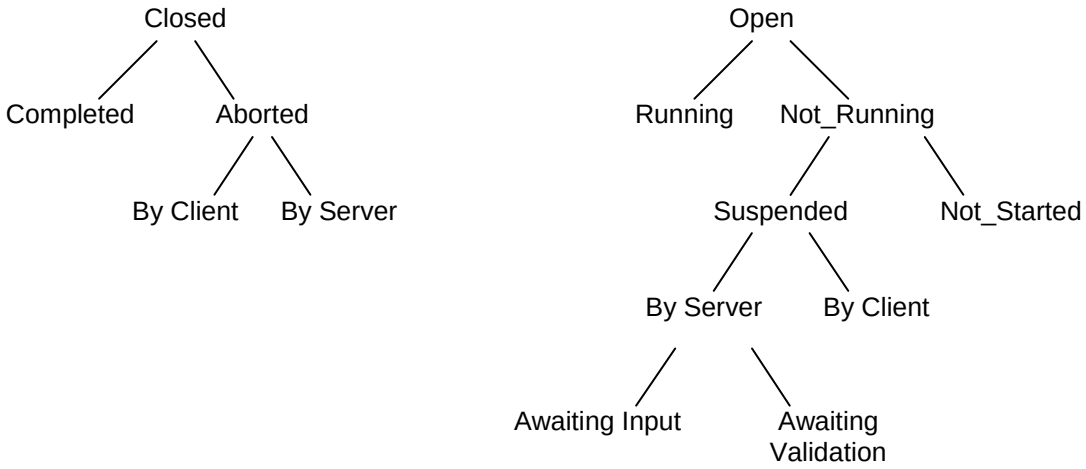

Figure 3-4. Service Order States

It should be noted that the states listed in Figure **3-4** are not completely aligned with the OSS/J Order Management (OM) API defined in JSR 264. In particular, JSR 264 has Awaiting Input and Awaiting Validation directly under Suspended, and does not have an explicit separation of the Suspended – By Server and Suspended – By Client substates. In order to make use of the OSS/J OM API the following mapping is suggested:

Table 3-15. Mapping between SAI and OSS/J OM Order States

SAI Order States	**Mapping to OSS/J OM Order State**
Suspended – By Server	Not allowed unless the substate is specified, i.e., Awaiting Input or Awaiting Validation
Suspended – By Client	Suspended
Suspended – By Server – Awaiting Input	Suspended – Awaiting Input
Suspended – By Server – Awaiting Validation	Suspended – Awaiting Validation

The OSS/J OM approach is awkward (if not incorrect) since Suspended implies Suspended – By Client but the substates of Suspended relate to suspension by the server.

R_TMF518_SA_2_II_0044	In conjunction with service activation requests, a service order will also be created by the target OS and be made accessible to the requesting OS (for reading and writing is some cases). The service order tracks both the functional and non-functional

	characteristics of the service activation request. There are three basic kinds of service orders: 1. **new** – a new service order is created when a product design, reservation, provisioning or activation request is received by the target OS. 2. **modification** – a modification service order is created when service modification request is received by the target OS. 3. **removal** – a removal service order is created when service deletion, termination or deactivation request is received by the target OS.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0045	The following requests may be made with respect to a given service order: 1. ***stop*** – stop from running and then rollback an open service order. The service order must be in the OPEN-RUNNING state when this operation is invoked. 2. ***cancel*** – rollback an open service order that is in the OPEN-NOT_RUNNING 3. ***finalize*** an order – this essentially puts the order in a completed state such that no further changes can be made to the order. At this point, the customer is typically informed of the order completion and billing for the service may begin. In particular, this operation takes the service order from the OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION state to the CLOSED-COMPLETED. Further, OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION is the only state value that can accept a finalize order request. An exception shall be returned if “finalize” is invoked when the service order is in any other state. 4. ***suspend*** – this operation is used to put the service order in the SUSPENDED-BYCLIENT state. The service order can either be in OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state when the operation is invoked. 5. ***resume*** – this operation is used to move the service from the SUSPENDED-BYCLIENT state to the OPEN-RUNNING state. 6. ***modify*** – this is used to modify either the functional or non-functional characteristics of an order. The service

	order must be in the OPEN state when this operation is invoked. • Some suppliers may restrict the substates of OPEN for which they support this function, e.g., support only when the service order is in the OPEN-NOT_RUNNING-NOT_STARTED state. Such restrictions will need to be stated in the supplier's implementation statement for the interface. • For the first release of this interface, this operation will only modify a subset of the non-functional characteristics.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0046	It shall be possible (for the OS managing a given service order) to generate a ***notification*** whenever a functional or non-functional characteristic of that service order has changed.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0047	Table **3-16** summarizes the allowable transitions between the service order states. The initial (or "from") states are listed in the first column and the final (or "to") states are listed in the top row. Note that Not Currently Allowed (NCA) is distinguished from Not Allowed (NA) in that NA is not possible by definition but NCA is possible but no requirements for the transition have been discovered yet.
Source	TMF518_SA_2, Version 1.0

	CLOSED-COMPLETED	CLOSED-ABORTED_BY CLIENT	CLOSED-ABORTED_BY SERVER	OPEN-NOT_RUNNING-SUSPENDED-BYCLIENT	OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_INPUT	OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_ VALIDATION	OPEN-NOT_RUNNING-NOT_STARTED	OPEN-RUNNING
CLOSED-COMPLETED	Not Allowed (NA)	NA	NA	NA	NA	NA	NA	NA
CLOSED-ABORTED_BYCLIENT	NA	NA	NA	NA	NA	NA	NA	NA
CLOSED-ABORTED_BYSERVER	NA	NA	NA	NA	NA	NA	NA	NA
OPEN-NOT_RUNNING-SUSPENDED-BYCLIENT	- server can decide to make this transition (typically happens as the result of outside intervention)	- client invokes CANCEL operation - the server may also make this transition after it has completed a roll back related to a STOP request	- the server may choose to abort the order while in any of the OPEN substates	Not Currently Allowed (NCA)	- server can decide to make this transition	- server can decide to make this transition	NA	- client invokes RESUME operation - order is resumed via some action outside of the interface
OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_INPUT	- server can decide to make this transition (typically happens as the result of outside intervention)	- client invokes CANCEL operation	- the server may choose to abort the order while in any of the OPEN substates	NCA	NCA	NCA	NA	- server will continue processing when it receives the required input
OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_ VALIDATION	- client invokes FINALIZE operation	- client invokes CANCEL operation	- the server may choose to abort the order while in any of the OPEN substates	NCA	NCA	NCA	NA	- the server may decide to return the order to this state

	CLOSED-COMPLETED	CLOSED-ABORTED_BY CLIENT	CLOSED-ABORTED_BY SERVER	OPEN-NOT_RUNNING-SUSPENDED-BYCLIENT	OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_INPUT	OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_ VALIDATION	OPEN-NOT_RUNNING-NOT_STARTED	OPEN-RUNNING
OPEN-NOT_RUNNING-NOT_STARTED	NA	- client invokes CANCEL operation	- the server may choose to abort the order while in any of the OPEN substates	- client invokes SUSPEND operation - server can decide to make this transition also	- server can decide to make this transition	- server can decide to make this transition	NA	- server can decide to make this transition
OPEN-RUNNING	- server determines that order is complete and client validation is not required	- NA	- the server may choose to abort the order while in any of the OPEN substates	- client invokes SUSPEND or STOP operation - server can decide to make this transition also	- server can decide to make this transition also	- server can decide to make this transition also	NA	This is possible but a specific transition has not yet been identified

Table 3-16. Service Order State Transitions

R_TMF518_SA_2_II_0048	Table **3-17** provides a mapping between the service order states and the states of the services (CFSs) associated with a service order for a given product instance. It is important to note that several CFS instances can be associated with a given service order.
Source	TMF518_SA_2, Version 1.0

Table 3-17. Relationship between Service States and Service Order States

Service Order State	Service States
CLOSED-COMPLETED	All of the services associated with the order should be in one of the allowable states with respect to the particular service request associated with the order. For example, if the service order is associated with an activation request, then all the services associated with the given product should be in the Provisioned_Active state. If the service order is associated with a modify request, then all the services most be in the same state as before the request was made.
CLOSED-ABORTED-ABORTED_BYCLIENT	At the time the service order is aborted, the associated services may be in any state. Typically, the services will eventually be terminated and then deleted. In the case of atomic request service request, the associated services must be returned to the same state as before the request.
CLOSED-ABORTED-ABORTED_BYSERVER	At the time the service order is aborted, the associated services may be in any state. Typically, the services will eventually be retired. In the case of atomic request service request, the associated services must be returned to the same state as before the request.
OPEN-NOT_RUNNING-SUSPENDED-AWAITING_INPUT	The services can be any state with the possible exception of Terminated.
OPEN-NOT_RUNNING-SUSPENDED-AWAITING_VALIDATION	All of the services associated with the order should be in one of the allowable states with respect to the particular service request associated with the order.
OPEN-NOT_RUNNING-NOT_STARTED	The services can be any state with the possible exception of Terminated.
OPEN_RUNNING	The services can be any state with the possible exception of Terminated.

One point of complication here is the relationship between the service order and the CFSs. The point is that there could be more than one service order executed in going from Start (where the CFS does not yet exist) to a point where all the CFSs are in the Provisioned_Active state. For example, the first service order may require that the CFSs be reserved and then transitioned to Provisioned_Inactive at a given time. A second service order may be executed when the CFSs are transitioned from Reserved to Provisioned_Active. In order to clarify this complication, the following requirement is stipulated.

R_TMF518_SA_2_II_0049	Each service order request over the Interface shall give rise to at most one service order execution in the target OS. The lifetime of the service order and associated service request shall coincide.
Source	TMF518_SA_2, Version 1.0

Requirement R_TMF518_SA_2_II_0049 only states that at most one service order is executed to fulfill a given service activation request. In some cases, however, it may not be necessary to expose the associated service object over the interface (see R_TMF518_SA_2_II_0050).

R_TMF518_SA_2_II_0050	In the order aware version of the Interface, a service order object is exposed over Interface based on the conditions stated in Table **3-18**.
Source	TMF518_SA_2, Version 1.0

Table 3-18. Conditions under which a service order object is exposed over the Interface.

Operation	**Conditions under which a service order object is created and accessible across the interface**
feasibilityCheck	A service order is not executed and consequently, a service order object can not be created in this case.
design	A service order object is created. However, it never started at this point. The service order should indicate that the CFSs are to be transitioned to either the Provisioned_Inactive or Provisioned_Active state.
reserve	A service order object is created.
provision	A service order object is created.
activate	A service order object is created except if the transition is from Provisioned_Inactive to Provisioned_Active in which case no service order is executed and consequently, a service order object can not be created.
deactivate	A service order is not executed and consequently, a service order object can not be created in this case.
terminate	A service order object is created.
retire	A service order object may be created, but it is not necessary.
cancel	A service order object may be created, but it is not necessary.
modify	A service order object is created.

automatic time-scheduled service transitions	A service order object is created when going from Provisioned_Active to Reserved. When going from Provisioned_Active to Provisioned_Inactive a service order is not executed and consequently, a service order object can not be created in this case.

R_TMF518_SA_2_II_0051	The Interface shall allow an OS the ability to retrieve all or a filter subset of the service orders known to the target OS. In particular, this concerns retrieval of service orders previously created over the SAI.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_II_0052	The Interface shall allow an OS to retrieve the current state of a given service order.
Source	TMF518_SA_2, Version 1.0

3.3.2.2 Detailed Behavior

This section provides detailed operations signatures for the various SAI operations. The general pattern is a single request, followed by an initial response and then a series of events that report on the progress of the request. It should be noted that these definitions are not intended to imply a specific Message Exchange Pattern (MEP). The term “event” is used in a very general sense here (just a progress report on the request) and it does not imply a notification in the MTOSI sense. Further, the events defined below may also contain exceptions.

R_TMF518_SA_2_II_0053	The **stop order** request shall convey the information and support the behavior described in Table **3-19**.
Source	TMF518_SA_2, Version 1.0

Table 3-19. Operation Signature for Stop Order

Operation Signature	Request: ***stopOrder*** (Service Order Name) Events: • ***stopOrderInitialEvent*** (Service Order Name, InitialResp) • ***stopOrderCompletionEvent*** (FinalResp)
Behavior	This operation stops and rollbacks an order in the OPEN-RUNNING state. The target OS (i.e., the OS receiving the request) returns all entities involved in the order back to the state just before the order was started.
Pre-conditions	1. The referenced order exists and is in the OPEN-RUNNING state and the target OS supports the “stop order” capability.

	2. The initial request for the associated services (CFSs) was made with the understanding that a rollback could be requested.
Post-conditions	In case of success: 1. The target OS has put the order in the CLOSED-ABORTED_BY CLIENT state. 2. The services related to the requested order have been rolled back. In case of failure: 1. The target OS cannot completely rollback the order and has informed the requesting OS. (In this case, manual measures will typically be needed to complete the rollback.) 2. Not all of the services related to the requested order have been rolled back.
Arguments	
Request	Service Order Name – identifies the order to be stopped
Response	InitialResp – this is an indication of whether or not the target OS can perform the request. There are two possible value *willDo* (1) and *cannotDo* (2). In the latter case, the target OS will supply an exception. FinalResp – this is an indication of whether or not the rollback associated with the stop request was successful. There are three possible values *successful* (1), *partiallySuccessful* (2) and *unsuccessful* (3). In the latter two cases, manual intervention will be needed to completely rollback the order.
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061)

R_TMF518_SA_2_II_0054	The **cancel order** request shall convey the information and support the behavior described in Table **3-20**.
Source	TMF518_SA_2, Version 1.0

Table 3-20. Operation Signature for Cancel Order

Operation Signature	Request: ***cancelOrder*** (Service Order Name) Events: • ***cancelOrderInitialEvent*** (Service Order Name, InitialResp) • ***cancelOrderCompletionEvent*** (FinalResp)
Behavior	This operation cancels and rollbacks an order in the OPEN-NOT_RUNNING state. The target OS (i.e., the OS receiving the request) deallocates and returns all entities involved in the order back to the state just before the order was started.
Pre-conditions	1. The referenced order exists and is in the OPEN-NOT_RUNNING state and the target OS supports the “cancel order” capability.

	2. The initial request for the associated service was made with the understanding that a rollback could be requested.
Post-conditions	In case of success: 1. The target OS has put the order in the CLOSED-ABORTED_BY CLIENT state. 2. The services related to the requested order have been rolled back. In case of failure: 1. The target OS cannot completely rollback the order and has informed the requesting OS. (In this case, manual measures will typically be needed to complete the rollback.) 2. Not all of the services related to the requested order have been rolled back.
Arguments	
Request	Service Order Name – identifies the order to be cancelled
Response	InitialResp – this is an indication of whether or not the target OS can perform the request. There are two possible value *willDo* (1) and *cannotDo* (2). In the latter case, the target OS will supply an exception. FinalResp – this is an indication of whether or not the rollback associated with the stop request was successful. There are three possible values *successful* (1), *partiallySuccessful* (2) and *unsuccessful* (3). In the latter two cases, manual intervention will be needed to completely rollback the order.
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

R_TMF518_SA_2_II_0055	The **finalize order** request shall convey the information and support the behavior described in Table **3-21**.
Source	TMF518_SA_2, Version 1.0

Table 3-21. Operation Signature for Finalize Order

Operation Signature	Request: ***finalizeOrder*** (Service Order Name) Response: ***finalizeOrderEvent*** ( )
Behavior	This operation puts an order in a completed state such that no further changes can be made to the order. At this point, the customer is typically informed of the order completion and billing for the service may begin (or end in the case of a termination).
Pre-conditions	The order is in the OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION state

Post-conditions	In case of success: 1. The target OS informs the requesting OS that order has been finalized. 2. The order is in the CLOSED-COMPLETED state. In case of failure: 1. The target OS informs the requesting OS that it was not able to finalize the order. 2. The order remains in the OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION state.
Arguments	
Request	Service Order Name – identifies the order to be finalized
Response	None
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

R_TMF518_SA_2_II_0056	The **suspend order** request shall convey the information and support the behavior described in Table **3-22**.
Source	TMF518_SA_2, Version 1.0

Table 3-22. Operation Signature for Suspend Order

Operation Signature	Request: ***suspendOrder*** (Service Order Name) Response: ***suspendOrderEvent*** ( )
Behavior	This operation is used to put the service order in the SUSPENDED-BYCLIENT state. The service order can either be in OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state when the operation is invoked. No further action is taken on the order until and unless a subsequent resume order request is made.
Pre-conditions	The service order must be in either the OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state
Post-conditions	In case of success: 1. The target OS informs the requesting OS that the order has been suspended. 2. The order is in the SUSPENDED-BYCLIENT state In case of failure: 1. The target OS informs the requesting OS that it was not able to suspend the order.

	2. The order remains in either the OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state.
Arguments	
Request	Service Order Name – identifies the order to be suspended
Response	None
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

R_TMF518_SA_2_II_0057	The **resume order** request shall convey the information and support the behavior described in Table **3-23**.
Source	TMF518_SA_2, Version 1.0

Table 3-23. Operation Signature for Resume Order

Operation Signature	Request: ***resumeOrder*** (Service Order Name) Response: ***resumeOrderEvent*** ( )
Behavior	This operation is used to move the service from the SUSPENDED-BYCLIENT state to the OPEN-RUNNING state.
Pre-conditions	1. The service order must be in the SUSPENDED-BYCLIENT state.
Post-conditions	In case of success: 1. The target OS informs the requesting OS that the order has been resumed. 2. The order is in the OPEN-RUNNING state. In case of failure: 1. The target OS informs the requesting OS that it was not able to resume the order. 2. The order remains in the SUSPENDED-BYCLIENT state.
Arguments	
Request	Service Order Name – identifies the order to be resumed
Response	None
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

R_TMF518_SA_2_II_0058	The **modify order** request shall convey the information and support the behavior described in Table **3-24**.
Source	TMF518_SA_2, Version 1.0

Table 3-24. Operation Signature for Modify Order

Operation Signature	Request: ***modifyOrder*** (Service Order Name, Attribute List) Response: ***modifyOrderEvent*** (Attribute List)
Behavior	This operation entails the modification of the following non-functional attributes: • requestedCompletionTime – the date and time that the requesting OS has specified for completion of the service order • priority – this attribute allows the request to specify a priority for execution of the service request. This attribute is a positive integer • orderOwner – this is the person or entity (e.g., work center) that has been assigned to track the order. A modification of all or any subset of the above attributes may be made in a single request.
Pre-conditions	1. The service order must be in the OPEN state when this operation is invoked. Some suppliers may restrict the substates of OPEN for which they support this function, e.g., support only when the service order is in the OPEN-NOT_RUNNING-NOT_STARTED state. Such restrictions will need to be stated in the supplier's implementation statement for the interface.
Post-conditions	In case of success: 1. The target OS informs the requesting OS that it has made some or all of the requested modifications. An indication is given for the failed changes. 2. The order is in the same state as before the request, and some or all of the requested attributes have been modified. In case of failure: 1. The target OS informs the requesting OS that it could not make any of the requested changes. 2. The order is in the same state as before the request and none of the requested attributes have been modified.
Arguments	
Request	Service Order Name – identifies the order to be resumed Attribute List – identifies the list of attributes to be changed, removed, or added. If an entry is to be removed, "-" shall be specified as a value. If an attribute is specified that is currently not included for the order, then the attribute is added (by the target) with the specified value.
Response	Attribute List – this indicates the attributes and associated values that have been

	set as a result of the operation. Removed attributes are shown with their value set to "-".
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

R_TMF518_SA_2_II_0059	The retrieveServiceOrders request shall convey the information and support the behavior described in Table **3-25**.
Source	TMF518_SA_2, Version 1.0

Table 3-25. Operation Signature for retrieveServiceOrders

Operation Signature	Request: ***retrieveServiceOrders*** (Filter) Response: ***retrieveServiceOrderResponse*** (Service Order List)
Behavior	This operation returns a filtered subset of the service orders and service order items known to the target OS. The following attribute matching evaluations shall be supported: Presence, Equality, Comparison, Substrings, SetOf (which can be qualified with Superset, Subset or Non-Null Intersection). These logical expressions are conformant to the requirements R_TMF518_FMW_II_0022 and R_TMF518_FMW_II_0023 in TMF518_FMW.
Pre-conditions	None
Post-conditions	In case of success: 1. No change is made to the state of the service orders or associated service order items as a result of this request. 2. The requesting OS has received the requested information concerning the service order and service order items. In case of failure: 1. No change is made to the state of the service orders or associated service order items as a result of this request. 2. The requesting OS has not received the requested information concerning the service order and service order items.
Arguments	
Request	Filter – identifies subset of the service order and service order items to be retrieved.
Response	Service Order List – this is a listing of requested service orders and associated service order items. The structure is List Of { Service Order Item, List Of {Service Order Items} }
Exceptions	The following exception is allowed:

	• basicFailureEvent (see R_TMF518_SA_2_III_0061).

R_TMF518_SA_2_II_0060	The retrieveOrderState request shall convey the information and support the behavior described in Table **3-26**.
Source	TMF518_SA_2, Version 1.0

Table 3-26. Operation Signature for retrieveOrderState

Operation Signature	Request: ***retrieveOrderState*** (Service Order Name, ProductName) Response: ***retrieveOrderStateResponse*** (ServiceOrderState)
Behavior	This operation returns the state of an identified service order. The requesting OS may either supply the name of the service order itself or the name of the associated product instance.
Pre-conditions	1. The service order and product instance must exist.
Post-conditions	In case of success: 1. No change is made to the state of the service order. 2. The requesting OS has been informed about the state of the given service order. In case of failure: 1. No change is made to the state of the service order. 2. The requesting OS has not been informed about the state of the given service order.
Arguments	
Request	Service Order Name – identifies the service order whose state is being requested. ProductName – the state of the service order associated with given ProductName is to be returned to the requesting OS. Only one of the above two attributes needs to be supplied in the request.
Response	ServiceOrderState – the state of the identified service order.
Exceptions	The following exception is allowed: • basicFailureEvent (see R_TMF518_SA_2_III_0061).

3.4 Category III: Abnormal or Exception Conditions, Dynamic Requirements

3.4.1 Exceptions concerning Service Activation Requests

R_TMF518_SA_2_III_0061	The ***basicFailureEvent*** is an indication that the target OS was not able to accomplish the requested task. The following parameters are supported: • exceptionType – the following basis exceptions are supported: ○ NOT_IMPLEMENTED – raised when the target OS does not support this operation. ○ INTERNAL_ERROR - raised when the target OS has encountered an internal error. ○ INVALID_INPUT – raised if any input parameter is syntactically incorrect. ○ UNABLE_TO_COMPLY – raised when the target OS cannot fulfill the request. This can be used in either the initial or final response. ○ COMM_FAILURE – raised when the target OS is not able to communicate with one or more entities and as a consequence could not complete the rollback. • errorReason – a free format string that further explains the exception
Source	TMF518_SA_2, Version 1.0

The events in R_TMF518_SA_2_III_0062, R_TMF518_SA_2_III_0063, R_TMF518_SA_2_III_0064 and R_TMF518_SA_2_III_0065 are essentially partially failure responses but are not exceptions in the sense that the entire request has failed, and as such, the events are treated a replies to the initial request. Based on Section 4.1.5 of SD2-2, the correlation between the request and theses multiple responses (failure events) is done via the CorrelationID in the response Header

R_TMF518_SA_2_III_0062	The ***serviceStateTransitionFailureEvent*** is an indication that the target OS is reporting a CFS state transition failure from the current state to the next one in the CFS activation lifecycle. The following parameters are supported: • serviceRequestRef – identifies the initial request • productName – identifies the associated product instance • cfsName – identifies the CFS to which the exception

	applies • currentState – identifies the current state of the CFS • failedState – identifies the state into which the target OS attempted (and failed) to transition the CFS.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_III_0063	The ***serviceAvcFailureEvent*** is an indication that the target OS could not make the serviceCharacteristicValue changes necessary to support a request to change a productCharacteristicValue. Multiple such failures can be reported in a single serviceAvcFailureEvent. The following parameters are supported: • serviceRequestRef – identifies the initial request • productName – identifies the associated product instance • cfsName – identifies the CFS to which the exception applies • List of ○ productSpecCharacteristicRef – the ID for the productSpecCharacteristic associated with the list of one or more serviceSpecCharacteristics that could not be changed. ○ List of serviceSpecCharacteristicRef – a list of the serviceSpecCharacteristics that could not be changed.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_III_0064	The ***serviceCreationFailureEvent*** is an indication that the target OS is reporting a CFS could not be created. The following parameters are supported: • serviceRequestRef – identifies the initial request • productName – identifies the associated product instance • cfsName – identifies the CFS to which the exception applies.
Source	TMF518_SA_2, Version 1.0

R_TMF518_SA_2_III_0065	The ***serviceDeletionFailureEvent*** is an indication that the target OS is reporting that a CFS could not be deleted. The following parameters are supported:

	• serviceRequestRef – identifies the initial request • productName – identifies the associated product instance • cfsName – identifies the CFS to which the exception applies • currentState – identifies the current state of the CFS.
Source	TMF518_SA_2, Version 1.0

3.5 Category IV: Expectations and Non-Functional Requirements

None have been identified thus far

3.6 Category V: System Administration Requirements

None have been identified thus far.

4 Use Cases

4.1 Service Activation

Figure **4-1** below shows some typical flows concerning the order of execution for the requests defined in requirement R_TMF518_SA_2_II_0008. This figure is only meant as a high-level summary, the use cases that follow will provide more details concerning the various process flows.

1. It is likely to alternate between invocations of Feasibility Check and Design as a service is being prepared for a customer. Once the customer agrees to go forward, a Reserve request would typically be executed.
2. It is possible to go directly from non-existence (the solid circle) to the execution of a Design, Reserve, Provision or Activate. This is not shown in the figure as it would become too cluttered.
3. It is possible to alternate between Activation and Deactivation of a service. For example this may be necessary if the customer is not paying their bill on time. Also, for some services, it may be necessary to first deactivate the service before doing certain modifications.
4. It is possible to activate a service via several (progressive) requests. For example, a service could first be designed and then, via a second request, be activated. It should be noted that the second request needs to reference the productId from the initial request.

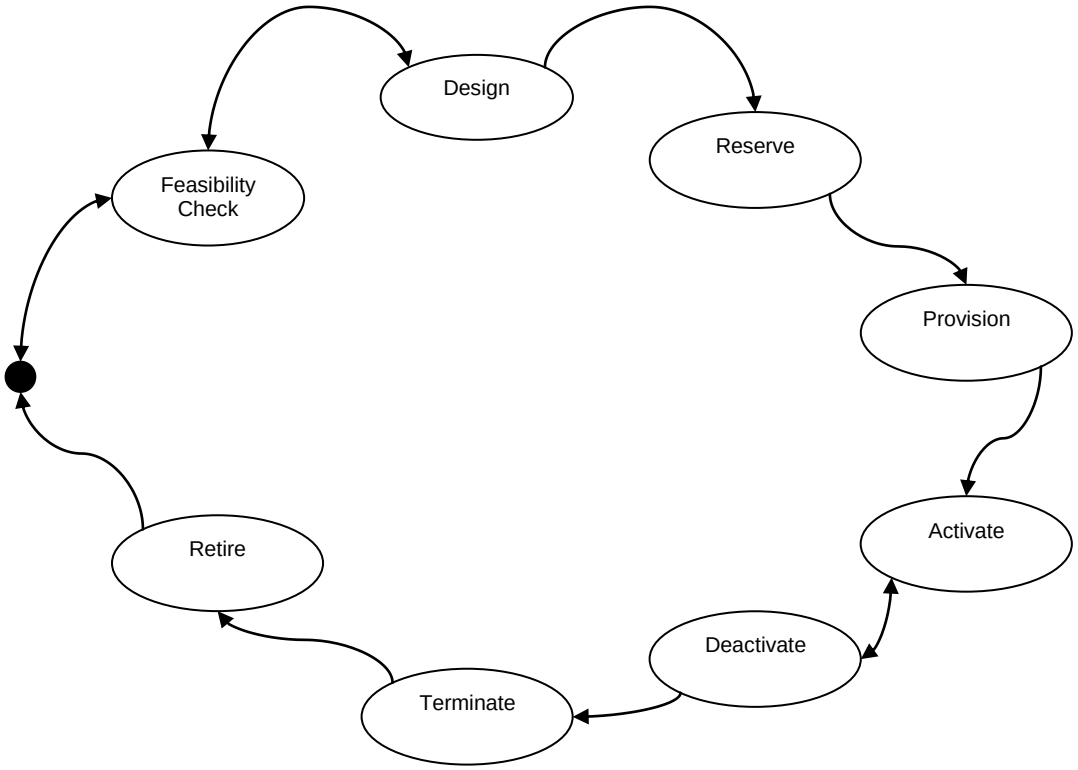

Figure 4-1. Example flows among the service activation functions

4.1.1 Feasibility Check for Services in Support of a Product Instance

Use Case Id	UC_TMF518_SA_2_0001
Use Case Name	Feasibility Check for Services in Support of a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) check the feasibility of provisioning a set of CFSs in support of a product instance. This is only a feasibility check and nothing is actually provisioned. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case which is defined in TMF518_FMW. 2. The CFSs supporting the product instance do not yet exist. **Assumptions**: 1. If a composite product is being managed by the Client application, it's assumed the Client application is responsible for decomposing the product and submitting an independent service request for each product.

	2. The issue of dependency between different products is dealt with in the bundle concept. 3. This Feasibility Check operation is restricted to the checking of the basic conditions for the initiation of the activation process. If any other feasibility checks are implemented during other states, this is out of scope in this document.
Begins When	The use case begins when the requesting OS sends a service request to the target OS.
Description	1. The requesting OS sends a feasibility check request to the target OS concerning a given product instance. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will determine if CFSs in support of the product instance ***can be*** provisioned and then activated. • If so, the target OS will inform the requesting OS. Optionally, an offered activation time can also be provided. The offered activation time is an estimate of when the target OS can activate the CFSs associated with the product instance assuming the requesting OS wants to go forward with the activation. The associated CFSs will be first be created and put in the FeasibilityChecked state once feasibility has been checked. • If not, the target OS will inform the requesting OS that the product is not feasible.
Ends When	Success: the requesting OS has been informed that product is feasible the associated CFSs are in the FeasibilityChecked state. Failure: the target OS has been informed that the product is not feasible.
Post-Conditions	Success: The associated CFSs are in the FeasibilityChecked state. Failure: No product, service or resource state changes have occurred as a result of the feasibility check.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064)
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0030, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0064

4.1.2 Design Services in Support of a Product Instance

Use Case Id	UC_TMF518_SA_2_0002
Use Case Name	Design Services in Support of a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) design a set of CFSs in support of a product instance. This is purely a design activity – nothing is provisioned as a result of this operation. This also entails a feasibility check. The concept of design goes beyond the feasibility check in that a service order is also assembled. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance either do not yet exist or are in the FeasibilityChecked or Designed state. **Assumptions**: 1. The target OS is aware of the state of any existing CFS that is being used to support the requested product instance. 2. If a composite product is being managed by the Client application, it's assumed the Client application is responsible for decomposing the product and submitting an independent service request for each product. 3. The issue of dependency between different products is dealt with in the bundle concept. In case of such dependency, the target OS is responsible for taking in consideration the activation dependencies between the CFSs of the Products in the bundle. **Notes** • It may be possible for this use case to work when a CFS is in a state other than non- existent, FeasibilityChecked or Designed (e.g., Reserved, Provisioned_Inactive or Provisioned_Active). However, this appears to involve more complicated behavior and it is recommended that such cases be left for consideration in a future release of the interface.
Begins When	The use case begins when the requesting OS sends a service design request to the target OS.
Description	1. The requesting OS sends a service design request to the target OS concerning a given product instance. If some work has been

	done on some or all of the associated services (i.e., some services have been designed), then this needs to be discernable by target OS, as noted in the assumption above. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions to design each CFS, e.g., creating RFS orders (these are not seen over the SAI, however) and assemble the associated service order. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g., CFS creation, CFS state transition events and service order creation events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception.
Ends When	Success: the requesting OS has been informed that the service design request has been successfully completed. Failure: the target OS has been informed that the service design request cannot be fulfilled.
Post-Conditions	Success: The CFS(s) associated with the product instance have all been designed and may be retrieved over the Interface. In the case of the order aware version of the Interface, the associated service order can also be retrieved at this point. Failure: 1. In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. 2. In the case of best effort, the CFSs and associated resources remain in their current state. No rollback is attempted in this case.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009,

	R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0031, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064

4.1.3 Reserve Services in Support of a Product Instance

Use Case Id	UC_TMF518_SA_2_0003
Use Case Name	Reserve Services in Support of a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) reserve a set of CFSs in support of a product instance. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName. This use case only covers the case of an initial activation or termination of the CFSs associated with a product. An example scenario concerning time-based products (with associated availability schedule) is provided in UC_TMF518_SA_2_0010.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance either do not yet exist, or are in any of the following states: (in the case of initial activation) FeasibilityChecked or Designed or Reserved and (in the case of termination) Provisioned_Active or Provisioned_Inactive. **Assumptions**: 1. There is no assumption that the requested services have already been checked for feasibility or designed. 2. The target OS is aware of the state of any existing CFS that is being used to support the requested product instance. 3. If a composite product is being managed by the Client application, it's assumed the Client application is responsible for decomposing the product and submitting an independent service request for each product. 4. The issue of dependency between different products is dealt with in the bundle concept. **Notes** • It may be possible for this use case to work when a CFS is in a state other the ones mentioned above. However, this appears to involve more complicated behavior and it is recommended that such cases be left for consideration in a future release of the interface.

Begins When	The use case begins when the requesting OS sends a service reservation request to the target OS.
Description	1. The requesting OS sends a service reservation request to the target OS concerning a given product instance. If some work has been done on some or all of the associated services (i.e., some services have been designed, checked for feasibility or reserved), then this needs to be discernable by target OS, as noted in the assumption above. As part of the request, the requesting OS needs to provide a product availability schedule (see R_TMF518_SA_2_II_0013, R_TMF518_SA_2_II_0014, R_TMF518_SA_2_II_0015, and R_TMF518_SA_2_II_0016 for more details on the schedule). As noted in the summary of this use case, involves a simple schedule, i.e., only an initial activation or a termination. The requesting OS also needs to indicate whether or not the reservation has been committed or not. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value “waiting_in_queue”. 3. The requesting OS must commit to the reservation (via the Amend operation) by the expiringTime; otherwise, the target OS will cancel the request. 4. At some point, the target OS takes the service request message from the queue and proceeds to process it (assuming the requesting OS has committed to the reservation). In cases where the interface is “order aware”, the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. • In the case of initial activation, the target OS will perform the necessary actions to move each CFS into the Reserved state. • In the case of termination, the CFSs will be left in their current state until the transition time. 5. The target OS sends progress updates to the requesting OS (e.g., CFS creation and CFS state transition events). In the case of initial activation, the CFSs will be moved to the Provisioned_Inactive or Provisioned_Active state (depending on the preference of the requesting OS) at the initial transition time specified by the productAvailabilitySchedule. In the case of termination, the CFSs will be moved to the Terminated state at the specified transition time. 6. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception.

Ends When	Success: the requesting OS has been informed that the service request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: The CFS(s) associated with the product instance are in the Provisioned_Inactive, Provisioned_Active or Terminated state depending on what is stipulated in the request message. Failure: 1. In the case of an atomic request, all CFS(s) and associated resources have all been returned to their original state before the service request. 2. In the case of best effort, the CFSs and associated resources remain in their current state. No rollback is attempted in this case.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0032, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064

4.1.4 Provision Services in Support of a Product Instance

Use Case Id	UC_TMF518_SA_2_0004
Use Case Name	Provision Services in Support of a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) provision a set of CFSs in support of a product instance. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance either do not yet exist, or are in any of the following states: FeasibilityChecked, Designed, Reserved or Provisioned_Inactive. **Assumptions**:

	1. There is no assumption that the requested services have already been checked for feasibility, designed or reserved. 2. The target OS is aware of the state of any existing CFS that is being used to support the requested product instance. 3. If a composite product is being managed by the Client application, it is assumed the Client application is responsible for decomposing the product and submitting an independent service request for each product. 4. The issue of dependency between different products is dealt with in the bundle concept.
Begins When	The use case begins when the requesting OS sends a service provisioning request to the target OS.
Description	1. The requesting OS sends a service provisioning request to the target OS concerning a given product instance. If some work has been done on some or all of the associated services (i.e., some services have been checked for feasibility, designed, reserved or even provisioned), then this needs to be discernable by target OS, as noted in the assumption above. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions to move each CFS into the Provisioned_Inactive state. Effectively, this activity entails the completion of the service order for each associated RFS. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g., CFS creation and CFS state transition events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception. When this request is successfully completed, each CFS (associated with the product instance) shall be in the Provisioned_Inactive state with all associated RFS (and subtending resources) configured and allocated to the CFS. At this point, the CFSs have not yet been activated for use by the customer.
Ends When	Success: the requesting OS has been informed that the service provisioning request has been successfully completed.

	Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: The CFS(s) associated with the product instance are all in the Provisioned_Inactive state. At this point, the CFSs have not yet been activated for use by the customer. Failure: 1. In the case of an atomic request, all CFS(s) and associated resources have all been returned to their original state before the service request. 2. In the case of best effort, the CFSs and associated resources remain in their current state. No rollback is attempted in this case.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0027, R_TMF518_SA_2_II_0033, R_TMF518_SA_2_II_0041, R_TMF518_SA_2_II_0052, R_TMF518_SA_2_II_0060, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064

4.1.5 Activation of Services in Support of a Product Instance

Use Case Id	UC_TMF518_SA_2_0005
Use Case Name	Activation of Services in Support of a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) activate a set of CFSs in support of a product instance. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance either do not yet exist, or are in any of the following states: FeasibilityChecked, Designed, Reserved, Provisioned_Inactive, Provisioned_Active or Terminated. **Assumptions**:

	• There is no assumption that the requested services have already been checked for feasibility, designed, reserved or provisioned. • The target OS is aware of the state of any existing CFS that is being used to support the requested product instance. • If a composite product is being managed by the Client application, it's assumed the Client application is responsible for decomposing the product and submitting an independent service request for each product. • The issue of dependency between different products is dealt with in the bundle concept.
Begins When	The use case begins when the requesting OS sends a service activation request to the target OS.
Description	1. The requesting OS sends a service activation request to the target OS concerning a given product instance. If some work has been done on some or all of the associated services (i.e., some services have been checked for feasibility, designed, reserved, provisioned or even activated), then this needs to be discernable by target OS, as noted in the assumption above. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions to move each CFS into the Provisioned_Active state. This entails making the CFSs available for use by the customer. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g., CFS creation and CFS state transition events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception. When this request is successfully completed, each CFS (associated with the product instance) shall be in the Provisioned_Active state and made available for use by the customer. The requesting OS determines that the service request is complete via one of the methods mentioned above concerning progress reports.
Ends When	Success: the requesting OS has been informed that the service activation request has been successfully completed.

	Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: The CFS(s) associated with the product instance are all in the Provisioned_Active state and available for use by the customer. Failure: 1. In the case of an atomic request, all CFS(s) and associated resources have all been returned to their original state before the service request. 2. In the case of best effort, the CFSs and associated resources remain in their current state. No rollback is attempted in this case.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0034, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064

4.1.6 Deactivation of Services Related to a Product Instance

The “tear-down” steps related to a product instance are not symmetric with the establishment steps. For example, there is no equivalent to “feasibility check” and “provision” during the tear-down phase since one does not un-check feasibility nor is there a need to un-provision the attributes associated with the CFSs supporting the product instance.

Use Case Id	UC_TMF518_SA_2_0006
Use Case Name	Deactivation of Services Related to a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) deactivate a set of CFSs related to a product instance. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance are in the Provisioned_Active state.
Begins When	The use case begins when the requesting OS sends a service deactivation request to the target OS.

Description	1. The requesting OS sends a service deactivation request to the target OS concerning a given product instance. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions to move each CFS into the Provisioned_Inactive state. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g., CFS state transition events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception. In the case of success, all the CFSs are in the Provisioned_Inactive state, but they cannot be used by the customer. It should be emphasized, however, that the RFSs (and associated resources) remain allocated to the CFSs.
Ends When	Success: the requesting OS has been informed that the service activation request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: The CFS(s) associated with the product instance are all in the Provisioned_Inactive state. Failure: 1. In the case of an atomic request, all CFS(s) and associated resources have all been returned to their original state before the service request. 2. In the case of best effort, the CFSs and associated resources remain in their current state. No rollback is attempted in this case.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062).
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0035, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0062

4.1.7 Termination of Services Related to a Product Instance

It should be noted that this use case is a little different from the others in this section in that it entails several state transitions for the CFSs associated with a product instance. There are two basic outcomes:

1. In one case, the intention is to remove the product. In this case, the associated CFS objects should be in the Terminated state at the end of this use case.
2. In the other case, the intention is to only disassociate the RFSs from the CFSs and to leave open the possibility of associating the RFSs at some later point (possibly different instances). In this case, the associated CFSs should be in the Designed state at the end of this use case.

Design Decision: It was considered putting the CFSs in a Deallocated state and then terminating the CFSs in another step, but it is not clear what happens (if anything) in going from Deallocated to Terminated. Further, it was decided not to have a Deallocated state as the behavior matches the Designed state (especially if the Deallocated state does not necessarily imply that the CFS will next be put in the Terminated state).

Use Case Id	UC_TMF518_SA_2_0007
Use Case Name	Termination of Services Related to a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) terminate all the CFSs in support of a given product instance. When this request is complete, the RFSs (and associated resources) related to the CFSs supporting the product are all disassociated from the CFSs and made available for use in other situations. If a CFS is in the Provisioned_Active state, it will first be deactivated (put in the Provisioned_Inactive state) and then terminated. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance are in the Reserved, Provisioned_Active or Provisioned_Inactive state.
Begins When	The use case begins when the requesting OS sends a service request to the target OS.
Description	1. The requesting OS sends a service termination request to the target OS concerning a given product instance. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will

	return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions deallocate all RFSs from each CFS and to place each CFS into the Terminated state. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g., CFS state transition events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception. 6. Once all the CFSs are in the Terminated state, the requesting OS will typically do one of the following. • (Option #1) request that the CFSs associated with the product instance be retired (i.e., deleted). This is done when the requesting OS has no intention of reactivating the CFSs associated with the product. • (Option #2) request that the CFSs associated with the product be put in the Designed state. This may be done if the customer is given a period of time to request reactivation of a product. 7. Alternately, based on a previously agreed policy, the target OS could automatically retire (i.e., delete) the CFSs after a given time period.
Ends When	Success: the requesting OS has been informed that the service request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: 1. All the RFSs associated with all CFSs related to the product instance have been deallocated. 2. All the CFSs are in the Terminated state (Option #1) or all are in the Designed state (Option #2). Failure: 1. Some CFSs have not been terminated and as such are not in the Terminated or Designed state. It is recommended that this use case be done on a best effort basis, i.e., deallocate as many CFSs as possible.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062).

Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0036, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0062

4.1.8 Retiring of Services Related to a Product Instance

Use Case Id	UC_TMF518_SA_2_0008
Use Case Name	Retiring of Services Related to a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) retire the CFSs (i.e., delete the objects themselves) related to a given product instance. When this use case is complete, there will be no remaining record (in the target OS) of the CFSs that once supported the product instance. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance are in the Terminated state.
Begins When	The use case begins when the requesting OS sends a service retire request to the target OS.
Description	1. The requesting OS sends a service retire request to the target OS concerning the CFS associated with a given product instance. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value “waiting_in_queue”. 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions delete the CFS objects. In cases where the interface is “order aware”, the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g.,

	CFS deletion events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception.
Ends When	Success: the requesting OS has been informed that the service request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: All the CFS objects related to the product instance have been deleted. Failure: Some CFSs objects related to the product instance have not been deleted. It is recommended that this use case be done on a best effort basis, i.e., delete as many CFSs as possible.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceDeletionFailureEvent (see R_TMF518_SA_2_III_0065)
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0037, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0065

4.1.9 Cancellation of Services Related to a Product Instance

Use Case Id	UC_TMF518_SA_2_0009
Use Case Name	Cancellation of Services Related to a Product Instance
Summary	An OS (called the requesting OS) requests that another OS (called the target OS) cancel the CFSs (i.e., delete the objects themselves) related to a given product instance. When this use case is complete, there will be no remaining record (in the target OS) of the CFSs that once supported the product instance. The request is made using the language of the CRM layer, e.g., productSpecificationName and productBundleName.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance are in the FeasibilityChecked, Designed or Reserved state.

Begins When	The use case begins when the requesting OS sends a service cancel request to the target OS.
Description	1. The requesting OS sends a service cancel request to the target OS concerning the CFS associated with a given product instance. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions delete the CFS objects. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (e.g., CFS deletion events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception.
Ends When	Success: the requesting OS has been informed that the service request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: All the CFS objects related to the product instance have been deleted. Failure: Some CFSs objects related to the product instance have not been deleted. It is recommended that this use case be done on a best effort basis, i.e., delete as many CFSs as possible.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceDeletionFailureEvent (see R_TMF518_SA_2_III_0065)
Traceability	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0038, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0065

4.1.10 Activation of Services Related to a Time-Based Product Instance

Use Case Id	UC_TMF518_SA_2_0010
Use Case Name	Activation of Services Related to a Time-Based Product Instance
Summary	In this use case, the services supporting a product instance are activated. However, the activation of the product is only for certain time periods. For example, it may be a time of day transport product that is only active from 12AM to 6AM each night. Non-periodic schedules are also supported. It is important to note that the time schedule passed over the interface pertains to the product and the target OS (SM&O process) needs to map the schedule to one or more of the CFSs associated with the product (not necessarily all the CFSs associated with the product).
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notifications. 3. The CFSs supporting the product instance either do not yet exist, or are in any of the following states: FeasibilityChecked, Designed, Reserved, Provisioned_Inactive, Provisioned_Active or Terminated. **Assumptions**: • There is no assumption that the requested services have already been checked for feasibility, designed, reserved or provisioned. • The target OS is aware of state of any existing CFS that is being used to support the requested product instance. • If a composite product is being managed by the Client application, it's assumed the Client application is responsible for decomposing the product and submitting an independent service request for each product. • The issue of dependency between different products is dealt with in the bundle concept.
Begins When	The use case begins when the requesting OS sends a service activation request to the target OS.
Description	1. The requesting OS sends a service activation request to the target OS concerning a given product instance. This request will contain specific information concerning the activation schedule (if needed) for the time-based product. If some work has been done on some or all of the associated services (i.e., some services have been checked for feasibility, designed, reserved, provisioned or even activated), then this needs to be discernable by target OS, as noted in the assumption above.

	2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS (SM&O process) needs to map the product schedule to one or more of the associated CFSs. 5. The target OS will perform the necessary actions to move each CFS into the Reserved state. The target OS sends progress updates to the requesting OS (e.g., CFS creation and CFS state transition events). Once the CFSs are moved to the Reserved state, this initial service order is closed. If the target OS cannot fulfill the schedule for the given initial start date/time, it should raise an exception and inform the requesting OS at which date/time the request can be fulfilled. 6. When the time comes for the initial activation, the target OS will transition each CFS associated with the product to the Provisioned_Active state. A service order may be executed in association with this transition. 7. Subsequent to the initial activation, some or all of the CFSs in support of the product instance will be transitioned (by the target OS) between Provisioned_Active and Provisioned_Inactive or Reserved based on the activation schedule or based on network events (such as protection switches). A service order is executed for the transitions from Provisioned_Active to Reserved but not for the transitions from Provisioned_Active to Provisioned_Inactive (see R_TMF518_SA_2_II_0050).
Ends When	Success: the requesting OS has been informed that the service activation request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: The CFS(s) associated with the product instance are all in the Provisioned_Active, Provisioned_Inactive or Reserved state (depending on the activation schedule or other criteria). Failure: • In the case of an atomic request, all CFS(s) and associated resources have all been returned to their original state before the service request.

	• In the case of best effort, the CFSs and associated resources remain in their current state. No rollback is attempted in this case.
Exceptions	• basicFailureEvent (see R_TMF518_SA_2_III_0061) • serviceStateTransitionFailureEvent (see R_TMF518_SA_2_III_0062) • serviceCreationFailureEvent (see R_TMF518_SA_2_III_0064).
Traceability	R_TMF518_SA_2_II_0013, R_TMF518_SA_2_II_0014, R_TMF518_SA_2_II_0015, R_TMF518_SA_2_II_0016 and R_TMF518_SA_2_II_0050

4.2 Service Modification

4.2.1 Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance

Use Case Id	UC_TMF518_SA_2_0011
Use Case Name	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance
Summary	This operation modifies the CFSs, subscribers, users and SAPs in support of a product instance. If feasibility check or design have not yet been done for a CFSs supporting the given product instance, this will be done as part of this operation. When this request is complete, each CFS (associated with the product instance) shall be in the given targetState (either Provisioned_Inactive or Provisioned_Active) with all associated RFSs (and subtending resources) configured and allocated to the appropriate CFS.
Actor(s)	Requesting OS
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case. 2. In the case notifications are used to inform the requesting OS about the progress of the service request, the requesting OS must first subscribe to the appropriate notification topic. 3. The CFSs supporting the product instance either do not yet exist or are in the FeasibilityChecked, Designed, Reserved (either Reserved-Committed or Reserved-Uncommitted), Provisioned_Inactive, Provisioned_Active state or Terminated.
Begins When	The use case begins when the requesting OS sends a service request to the target OS.
Description	1. The requesting OS sends a service request to the target OS concerning an existing product instance. The requesting OS provides a desired list of product specification characteristics and

	product characteristic values, list of SAPs, list of users and/or list of subscribers. 2. The target OS analyzes and validates the request message, checks its priority, and puts it in a corresponding queue for processing. • If the message validation process fails, an appropriate exception will be returned to the requesting OS. • If the message validation succeeds, the target OS will return a response with the request status parameter value "waiting_in_queue". 3. At some point, the target OS takes the service request message from the queue and proceeds to process it. The target OS will perform the necessary actions modify the entities associated with the existing product instance (e.g., CFSs) in support of the modification request. **Typically, the target OS will attempt to make efficient re-use of the entities already allocated to the product instance if possible.** In cases where the interface is "order aware", the target OS will also return the Service Order ID to the requesting OS. The requesting OS may retrieve the associated service order and service order items. 4. The target OS sends progress updates to the requesting OS (i.e., CFS creation, CFS deletion, CFS state transition and CFS attribute value change events). 5. The final response from the target OS will indicate that request has been successfully completely; otherwise, the target OS will send an exception.
Ends When	Success: the requesting OS has been informed that the service request has been successfully completed. Failure: the target OS has been informed that the service request cannot be fulfilled.
Post-Conditions	Success: • The CFS(s) associated with the product instance are all in the targetState (either Provisioned_Inactive or Provisioned_Active). Subscribers, users and SAPs will be added or deleted as indicated in the modify request. Failure: • In the case of an atomic request, all CFS(s) have all been returned to their original state before the service request. Also, no subscriber, users or SAPs will be added or deleted. • In the case of best effort, the CFSs associated with the product instance remain in their current state. No rollback is attempted in this case.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax)

	3. Unable to comply: The requested operation could not be performed. 4. Communications failure: The target OS was not able to communicate with one or more entities and as a consequence could not complete the rollback.
Traceability	R_TMF518_SA_2_II_0017, R_TMF518_SA_2_II_0018, R_TMF518_SA_2_II_0019, R_TMF518_SA_2_II_0020, R_TMF518_SA_2_II_0021, R_TMF518_SA_2_II_0022, R_TMF518_SA_2_II_0023, R_TMF518_SA_2_II_0024, R_TMF518_SA_2_II_0025, R_TMF518_SA_2_II_0039

4.3 Service Ordering

4.3.1 Stop an Order

Use Case Id	UC_TMF518_SA_2_0012
Use Case Name	Stop an Order
Summary	The *requesting OS* sends a request to stop and rollback an order in the OPEN-RUNNING state. The target OS (i.e., the OS receiving the request) returns all entities involved in the order back to the state just before the order was started. This use case entails a multiple response communication pattern. The first response is essentially an acknowledgement or a rejection (based on whether the target OS supports the operation and whether the operation itself is well-formed). Subsequently, the target OS sends a second response that is either an indication that the rollback is complete or an exception (indicating that the rollback could not be completed).
Actor(s)	*Requesting OS*
Pre-Conditions	1. The referenced order exists and is in the OPEN-RUNNING state and the target OS supports the “stop order” capability. 2. The initial request for the associated service was made with the understanding that a rollback could be requested. 3. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to stop an order.
Description	1. The *requesting OS* sends a request to the *target OS* to stop an order. 2. If the order is not in the OPEN-RUNNING state, the target OS will reject the order. 3. If the order is in the OPEN-RUNNING state, the target OS puts

	the order in the OPEN-NOT_RUNNING-SUSPENDED-BYCLIENT state. 4. The target OS starts the rollback and returns all entities involved in the order back to the state just before the order was started. 5. Assuming inventory notifications are supported by the target OS and notifications have been enabled, the target OS will publish notifications concerning the various CFS state changes. 6. If the target OS cannot completely rollback the order, the target OS • leaves the order in the OPEN-NOT_RUNNING-SUSPENDED-BYCLIENT state • sends an exception to the requesting OS (this will be either a Processing Failure or Communication Failure) 7. If and when the order is completely rolled back, the target OS • puts the order in the CLOSED-ABORTED_BY CLIENT state • sends a stop order completion indication back to the requesting OS
Ends When	In case of success: The target OS has put the order in the CLOSED-ABORTED_BY CLIENT state. In case of failure: The target OS cannot completely rollback the order and has informed the requesting OS. (In this case, manual measures will typically be needed to complete the rollback.)
Post-Conditions	In case of success: The services related to the requested order have been rolled back. In case of failure: Not all of the services related to the requested order have been rolled back.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed. 4. Communications failure: The target OS was not able to communicate with one or more entities and as a consequence could not complete the rollback.
Traceability	R_TMF518_SA_2_II_0053

4.3.2 Cancel an Order

This is very similar to the “stop order” use case except the order is in an OPEN-NOT_RUNNING state.

Use Case Id	UC_TMF518_SA_2_0013
Use Case Name	Cancel an Order
Summary	The *requesting OS* sends a request to cancel and rollback an order in the OPEN-NOT_RUNNING state. The target OS (i.e., the OS receiving the request) returns all entities involved in the order back to the state just before the order was started. This use case entails a multiple response communication pattern. The first response is essentially an acknowledgement or a rejection (based on whether the target OS supports the operation and whether the operation itself is well-formed). Subsequently, the target OS sends a second response that is either an indication that the rollback is complete or an exception (indicating that the rollback could not be completed).
Actor(s)	*Requesting OS*
Pre-Conditions	1. The referenced order exists and is in the OPEN-NOT_RUNNING state and the target OS supports the “cancel order” capability. 2. The initial request for the associated service was made with the understanding that a rollback could be requested. 3. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to cancel an order.
Description	1. The *requesting OS* sends a request to the *target OS* to cancel an order. 2. If the order is not in the OPEN-NOT_RUNNING state, the target OS will reject the order. 3. If the order is in one of the OPEN-NOT_RUNNING states, the target OS starts the rollback and returns all entities involved in the order back to the state just before the order was started 4. Assuming inventory notifications are supported by the target OS and notifications have been enabled, the target OS will publish notifications concerning the various service deactivations, deallocations and deletions noted in the item above. 5. If the target OS cannot completely rollback the order, the target OS • leaves the order in the current OPEN-NOT_RUNNING state, e.g., OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_INPUT

	• sends an exception to the requesting OS (this will be either a Processing Failure or Communication Failure) 6. If and when the order is completely rolled back, the target OS • puts the order in the CLOSED-ABORTED_BY CLIENT state • sends a cancel order completion indication back to the requesting OS
Ends When	In case of success: The target OS has put the order in the CLOSED-ABORTED_BY CLIENT state. In case of failure: The target OS cannot completely rollback the order and has informed the requesting OS. (In this case, manual measures will typically be needed to complete the rollback.)
Post-Conditions	In case of success: The services related to the requested order have been rolled back. In case of failure: Not all of the services related to the requested order have been rolled back.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed. 4. Communications failure: The target OS was not able to communicate with one or more entities and as a consequence could not complete the rollback.
Traceability	R_TMF518_SA_2_II_0054

4.3.3 Finalize an Order

Use Case Id	UC_TMF518_SA_2_0014
Use Case Name	Finalize an Order
Summary	The requesting OS sends a request to have an order put in the CLOSED-COMPLETED state such that no further changes can be made to the order. At this point, the customer is typically informed of the order completion and billing for the service may begin.

Actor(s)	*Requesting OS*
Pre-Conditions	1. The order is in the OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION state 2. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to finalize an order.
Description	1. The requesting OS sends a finalize order request to the target OS 2. The target OS will reject the request if any of the following is true • the order is not in the OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION state or • the request is ill-formed. 3. Assuming the request is valid (i.e., passes the test stated in the item above), the target OS will finalize the order and thereby allowing no further modifications to the order. 4. The order is placed in the CLOSED-COMPLETED state and an appropriate state change notification is emitted, assuming the target OS supports notifications and notifications have been enabled. 5. The target OS responds to the requesting OS with an indication that the order has been finalized.
Ends When	In case of success: The target OS informs the requesting OS that order has been finalized. In case of failure: The target OS informs the requesting OS that it was not able to finalize the order.
Post-Conditions	In case of success: The order is in the CLOSED-COMPLETED state. In case of failure: The order remains in the OPEN-NOT_RUNNING-SUSPENDED-BYSERVER-AWAITING_VALIDATION state.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed.

Traceability	R_TMF518_SA_2_II_0055

4.3.4 Suspend an Order

Use Case Id	UC_TMF518_SA_2_0015
Use Case Name	Suspend an Order
Summary	The requesting OS sends a request to have an order suspended. No further action is taken on the order until and unless a subsequent resume order request is made.
Actor(s)	*Requesting OS*
Pre-Conditions	1. The service order must be in either the OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state 2. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to suspend an order.
Description	1. The requesting OS sends a suspend order request to the target OS. 2. The target OS will reject the request if any of the following is true • The order is not in either the OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state. • The request is ill-formed. 3. Assuming the request is valid (i.e., passes the test stated in the item above), the target OS will suspend the order and allow no further progress on the order until a resume request is made. 4. The order is placed in the SUSPENDED-BYCLIENT state and an appropriate state change notification is emitted, assuming the target OS supports notifications and notifications have been enabled. 5. The target OS responds to the requesting OS with an indication that the order has been suspended.
Ends When	In case of success: The target OS informs the requesting OS that the order has been suspended. In case of failure: The target OS informs the requesting OS that it was not able to suspend the order.
Post-Conditions	In case of success: The order is in the SUSPENDED-BYCLIENT state.

	In case of failure: The order remains in either the OPEN-RUNNING or OPEN-NOT_RUNNING-NOT_STARTED state.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed.
Traceability	R_TMF518_SA_2_II_0056

4.3.5 Resume an Order

Use Case Id	UC_TMF518_SA_2_0016
Use Case Name	Resume an Order
Summary	The requesting OS sends a request to have a suspended order resumed and continue processing of the order.
Actor(s)	*Requesting OS*
Pre-Conditions	1. The service order must be in the SUSPENDED-BYCLIENT state 2. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to resume an order.
Description	1. The requesting OS sends a resume order request to the target OS. 2. The target OS will reject the request if any of the following is true • The order is not in the SUSPENDED-BYCLIENT state. • The request is ill-formed. 3. Assuming the request is valid (i.e., passes the test stated in the item above), the target OS will resume the order, i.e., resume processing of the order. 4. The order is placed in the OPEN_RUNNING state and an appropriate state change notification is emitted, assuming the target OS supports notifications and notifications have been enabled. 5. The target OS responds to the requesting OS with an indication that the order has been resumed.
Ends When	In case of success:

	The target OS informs the requesting OS that the order has been resumed. In case of failure: The target OS informs the requesting OS that it was not able to resume the order.
Post-Conditions	In case of success: The order is in the OPEN-RUNNING state. In case of failure: The order remains in the SUSPENDED-BYCLIENT state.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed.
Traceability	R_TMF518_SA_2_II_0057

4.3.6 Modify an Order

Use Case Id	UC_TMF518_SA_2_0017
Use Case Name	Modify an Order
Summary	For this release of the interface, the use case only entails the modification of the following non-functional attributes: • requestedCompletionTime – the date and time that the client OS has requested for completion of the service order. • priority – this attribute allows the request to specify a priority for execution of the service request. This attribute is a positive integer. • orderOwner – this is the person or entity (e.g., work center) that has been assigned to track the order. A modification of all or any subset of the above attributes may be made in a single request. Future releases may entail the modification of additional non-functional attributes as well as functional items such as services.
Actor(s)	*Requesting OS*
Pre-Conditions	1. The service order must be in the OPEN state when this operation is invoked. Some suppliers may restrict the substates of OPEN for which they support this function, e.g., support only when the service order is in the OPEN-NOT_RUNNING-NOT_STARTED state. Such restrictions

	will need to be stated in the supplier's implementation statement for the interface. 2. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to modify an order.
Description	1. The *requesting OS* sends a request to the *target OS* to modify an order. 2. The target OS will reject the request if following is true • the order is not in an OPEN state • the request entails something other than the allowable modifications listed in the summary above • the request is ill-formed. 3. Assuming the request is valid (i.e., passes the test stated in the item above), the target OS will make the requested changes. If for some reason not all the changes can be made, the target OS informs the requesting OS of the changes that could be made.
Ends When	In case of success: The target OS informs the requesting OS that it has made some or all of the requested modifications. An indication is given for the failed changes. In case of failure: The target OS informs the requesting OS that it could not make any of the requested changes.
Post-Conditions	In case of success: The order is in the same state as before the request, and some or all of the requested attributes have been modified. In case of failure: The order is in the same state as before the request and none of the requested attributes have been modified.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed.
Traceability	R_TMF518_SA_2_II_0058

4.3.7 Service Order Retrieval

Use Case Id	UC_TMF518_SA_2_0018
Use Case Name	Service Order Retrieval
Summary	The requesting OS retrieves the service orders, service order items and associated services from a target OS. The requesting OS may request either all or some of the service orders. This is done by adding a filter to the request.
Actor(s)	*Requesting OS*
Pre-Conditions	1. All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The *requesting OS* sends a request to the *target OS* to retrieve service order information.
Description	1. The requesting OS requests some or all of the service orders, service order items and associated services (i.e., CFSs) from the target OS. A filter (contained in the request) identifies a subset of the total set of service orders. 2. The target OS collects and packages the requests information (service orders, service order items and associated services) and sends the information to the requesting OS.
Ends When	In case of success: The target OS has sent all the requested information to the requesting OS. In case of failure: The target OS informs the requesting OS that it could not fulfill its request.
Post-Conditions	In case of success: The requesting OS has all the information that it requested. The state of the service orders should not have changed as a result of the request. In case of failure: The requesting OS does not have the information that it requested. The state of the service orders should not have changed as a result of the request.
Exceptions	1. Not implemented: The target OS does not support this service. 2. Invalid input: An input parameter is invalid (e.g., invalid parameter syntax) 3. Unable to comply: The requested operation could not be performed.

Traceability	R_TMF518_SA_2_II_0051

5 Traceability Matrices

Table 5-1. Use Cases – Requirements Traceability Matrix

Use Case Id	Use Case Name	Requirements
UC_TMF518_SA_2_0001	Feasibility Check for Services in Support of a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0030, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0064
UC_TMF518_SA_2_0002	Design Services in Support of a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0031, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064
UC_TMF518_SA_2_0003	Reserve Services in Support of a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0032, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064
UC_TMF518_SA_2_0004	Provision Services in Support of a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0027, R_TMF518_SA_2_II_0033, R_TMF518_SA_2_II_0041, R_TMF518_SA_2_II_0052, R_TMF518_SA_2_II_0060, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064
UC_TMF518_SA_2_0005	Activation of Services in Support of a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0034, R_TMF518_SA_2_III_0061, R_TMF518_SA_2_III_0062 and R_TMF518_SA_2_III_0064

UC_TMF518_SA_2_0006	Deactivation of Services Related to a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0035, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0062
UC_TMF518_SA_2_0007	Termination of Services Related to a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0036, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0062
UC_TMF518_SA_2_0008	Retiring of Services Related to a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0037, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0065
UC_TMF518_SA_2_0009	Cancellation of Services Related to a Product Instance	R_TMF518_SA_2_II_0008, R_TMF518_SA_2_II_0009, R_TMF518_SA_2_II_0010, R_TMF518_SA_2_II_0038, R_TMF518_SA_2_III_0061 and R_TMF518_SA_2_III_0065
UC_TMF518_SA_2_0010	Activation of Services Related to a Time-Based Product Instance	R_TMF518_SA_2_II_0013, R_TMF518_SA_2_II_0014, R_TMF518_SA_2_II_0015, R_TMF518_SA_2_II_0016 and R_TMF518_SA_2_II_0050
UC_TMF518_SA_2_0011	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	R_TMF518_SA_2_II_0017, R_TMF518_SA_2_II_0018, R_TMF518_SA_2_II_0019, R_TMF518_SA_2_II_0020, R_TMF518_SA_2_II_0021, R_TMF518_SA_2_II_0022, R_TMF518_SA_2_II_0023, R_TMF518_SA_2_II_0024, R_TMF518_SA_2_II_0025, R_TMF518_SA_2_II_0039
UC_TMF518_SA_2_0012	Stop an Order	R_TMF518_SA_2_II_0053
UC_TMF518_SA_2_0013	Cancel an Order	R_TMF518_SA_2_II_0054
UC_TMF518_SA_2_0014	Finalize an Order	R_TMF518_SA_2_II_0055
UC_TMF518_SA_2_0015	Suspend an Order	R_TMF518_SA_2_II_0056
UC_TMF518_SA_2_0016	Resume an Order	R_TMF518_SA_2_II_0057
UC_TMF518_SA_2_0017	Modify an Order	R_TMF518_SA_2_II_0058
UC_TMF518_SA_2_0018	Service Order Retrieval	R_TMF518_SA_2_II_0051

Table 5-2. Requirements – Use Cases Traceability Matrix

Requirement Id	Use Case Name	Use Case Id
R_TMF518_SA_2_BR_0001		
R_TMF518_SA_2_BR_0002		
R_TMF518_SA_2_BR_0003		
R_TMF518_SA_2_BR_0004		
R_TMF518_SA_2_BR_0005		
R_TMF518_SA_2_BR_0006		
R_TMF518_SA_2_BR_0007		
R_TMF518_SA_2_II_0008	Cancellation of Services Related to a Product Instance Retiring of Services Related to a Product Instance Termination of Services Related to a Product Instance Deactivation of Services Related to a Product Instance Activation of Services in Support of a Product Instance Provision Services in Support of a Product Instance Reserve Services in Support of a Product Instance Design Services in Support of a Product Instance Feasibility Check for Services in Support of a Product Instance	UC_TMF518_SA_2_0009 UC_TMF518_SA_2_0008 UC_TMF518_SA_2_0007 UC_TMF518_SA_2_0006 UC_TMF518_SA_2_0005 UC_TMF518_SA_2_0004 UC_TMF518_SA_2_0003 UC_TMF518_SA_2_0002 UC_TMF518_SA_2_0001

R_TMF518_SA_2_II_0009	Cancellation of Services Related to a Product Instance Retiring of Services Related to a Product Instance Termination of Services Related to a Product Instance Deactivation of Services Related to a Product Instance Activation of Services in Support of a Product Instance Provision Services in Support of a Product Instance Reserve Services in Support of a Product Instance Design Services in Support of a Product Instance Feasibility Check for Services in Support of a Product Instance	UC_TMF518_SA_2_0009 UC_TMF518_SA_2_0008 UC_TMF518_SA_2_0007 UC_TMF518_SA_2_0006 UC_TMF518_SA_2_0005 UC_TMF518_SA_2_0004 UC_TMF518_SA_2_0003 UC_TMF518_SA_2_0002 UC_TMF518_SA_2_0001
R_TMF518_SA_2_II_0010	Cancellation of Services Related to a Product Instance Retiring of Services Related to a Product Instance Termination of Services Related to a Product Instance Deactivation of Services Related to a Product Instance Activation of Services in Support of a Product Instance Provision Services in Support of a Product Instance Reserve Services in Support of a Product Instance Design Services in Support of a Product Instance Feasibility Check for Services in Support of a Product Instance	UC_TMF518_SA_2_0009 UC_TMF518_SA_2_0008 UC_TMF518_SA_2_0007 UC_TMF518_SA_2_0006 UC_TMF518_SA_2_0005 UC_TMF518_SA_2_0004 UC_TMF518_SA_2_0003 UC_TMF518_SA_2_0002 UC_TMF518_SA_2_0001
R_TMF518_SA_2_II_0011		
R_TMF518_SA_2_II_0012		
R_TMF518_SA_2_II_0013	Activation of Services Related to a Time-Based Product Instance	UC_TMF518_SA_2_0010
R_TMF518_SA_2_II_0014	Activation of Services Related to a Time-Based Product Instance	UC_TMF518_SA_2_0010

R_TMF518_SA_2_II_0015	Activation of Services Related to a Time-Based Product Instance	UC_TMF518_SA_2_0010
R_TMF518_SA_2_II_0016	Activation of Services Related to a Time-Based Product Instance	UC_TMF518_SA_2_0010
R_TMF518_SA_2_II_0017	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0018	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0019	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0020	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0021	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0022	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0023	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0024	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0025	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0026		
R_TMF518_SA_2_II_0027	Provision Services in Support of a Product Instance	UC_TMF518_SA_2_0004
R_TMF518_SA_2_II_0028		
R_TMF518_SA_2_II_0029		
R_TMF518_SA_2_II_0030	Feasibility Check for Services in Support of a Product Instance	UC_TMF518_SA_2_0001
R_TMF518_SA_2_II_0031	Design Services in Support of a Product Instance	UC_TMF518_SA_2_0002
R_TMF518_SA_2_II_0032	Reserve Services in Support of a Product Instance	UC_TMF518_SA_2_0003

R_TMF518_SA_2_II_0033	Provision Services in Support of a Product Instance	UC_TMF518_SA_2_0004
R_TMF518_SA_2_II_0034	Activation of Services in Support of a Product Instance	UC_TMF518_SA_2_0005
R_TMF518_SA_2_II_0035	Deactivation of Services Related to a Product Instance	UC_TMF518_SA_2_0006
R_TMF518_SA_2_II_0036	Termination of Services Related to a Product Instance	UC_TMF518_SA_2_0007
R_TMF518_SA_2_II_0037	Retiring of Services Related to a Product Instance	UC_TMF518_SA_2_0008
R_TMF518_SA_2_II_0038	Cancellation of Services Related to a Product Instance	UC_TMF518_SA_2_0009
R_TMF518_SA_2_II_0039	Modify CFSs, Subscribers, Users and SAPs Associated with a Product Instance	UC_TMF518_SA_2_0011
R_TMF518_SA_2_II_0040		
R_TMF518_SA_2_II_0041	Provision Services in Support of a Product Instance	UC_TMF518_SA_2_0004
R_TMF518_SA_2_II_0042		
R_TMF518_SA_2_II_0043		
R_TMF518_SA_2_II_0044		
R_TMF518_SA_2_II_0045		
R_TMF518_SA_2_II_0046		
R_TMF518_SA_2_II_0047		
R_TMF518_SA_2_II_0048		
R_TMF518_SA_2_II_0049		
R_TMF518_SA_2_II_0050	Activation of Services Related to a Time-Based Product Instance	UC_TMF518_SA_2_0010
R_TMF518_SA_2_II_0051	Service Order Retrieval	UC_TMF518_SA_2_0018
R_TMF518_SA_2_II_0052	Provision Services in Support of a Product Instance	UC_TMF518_SA_2_0004
R_TMF518_SA_2_II_0053	Stop an Order	UC_TMF518_SA_2_0012
R_TMF518_SA_2_II_0054	Cancel an Order	UC_TMF518_SA_2_0013
R_TMF518_SA_2_II_0055	Finalize an Order	UC_TMF518_SA_2_0014
R_TMF518_SA_2_II_0056	Suspend an Order	UC_TMF518_SA_2_0015
R_TMF518_SA_2_II_0057	Resume an Order	UC_TMF518_SA_2_0016
R_TMF518_SA_2_II_0058	Modify an Order	UC_TMF518_SA_2_0017
R_TMF518_SA_2_II_0059		

R_TMF518_SA_2_II_0060	Provision Services in Support of a Product Instance	UC_TMF518_SA_2_0004
R_TMF518_SA_2_III_0061	Cancellation of Services Related to a Product Instance Retiring of Services Related to a Product Instance Termination of Services Related to a Product Instance Deactivation of Services Related to a Product Instance Activation of Services in Support of a Product Instance Provision Services in Support of a Product Instance Reserve Services in Support of a Product Instance Design Services in Support of a Product Instance Feasibility Check for Services in Support of a Product Instance	UC_TMF518_SA_2_0009 UC_TMF518_SA_2_0008 UC_TMF518_SA_2_0007 UC_TMF518_SA_2_0006 UC_TMF518_SA_2_0005 UC_TMF518_SA_2_0004 UC_TMF518_SA_2_0003 UC_TMF518_SA_2_0002 UC_TMF518_SA_2_0001
R_TMF518_SA_2_III_0062	Termination of Services Related to a Product Instance Deactivation of Services Related to a Product Instance Activation of Services in Support of a Product Instance Provision Services in Support of a Product Instance Reserve Services in Support of a Product Instance Design Services in Support of a Product Instance	UC_TMF518_SA_2_0007 UC_TMF518_SA_2_0006 UC_TMF518_SA_2_0005 UC_TMF518_SA_2_0004 UC_TMF518_SA_2_0003 UC_TMF518_SA_2_0002
R_TMF518_SA_2_III_0063		

R_TMF518_SA_2_III_0064	Activation of Services in Support of a Product Instance Provision Services in Support of a Product Instance Reserve Services in Support of a Product Instance Design Services in Support of a Product Instance Feasibility Check for Services in Support of a Product Instance	UC_TMF518_SA_2_0005 UC_TMF518_SA_2_0004 UC_TMF518_SA_2_0003 UC_TMF518_SA_2_0002 UC_TMF518_SA_2_0001
R_TMF518_SA_2_III_0065	Cancellation of Services Related to a Product Instance Retiring of Services Related to a Product Instance	UC_TMF518_SA_2_0009 UC_TMF518_SA_2_0008

6 Future Directions

6.1 Open Issues

Open Issue 1 We may move Sections 3.3.1.2 and 3.3.2.2 to the IA at some point.

6.2 Items for Future Study

6.2.1 User and Subscriber Management

The Interface does not currently support user and subscriber management.

The SAI conveys lists of User, Subscriber and SAP identities for a given product as part of a service activation request. The SAI does not convey information on User, Subscriber or SAP characteristics, or the relationship between Users and Subscribers. Rather, that information is passed over one or more other CRM-SM&O interfaces, which are currently outside the scope of the SAI. It would be the responsibility of the SM&O layer to internally make the appropriate association between service activation requests and the user/subscriber/SAP information, and any necessary association of users/subscribers/SAPs to specific Customer Facing Services (CFSs) for any given product.

There is an assumption that subscribers are associated with the identified product. The users are assumed to be associated with all the CFSs related to the given product. It is possible, however, to have a situation where only a subset of the users is associated with some or all of the CFSs. In such cases, the relationships between the users and CFSs needs to be handle outside of the Interface.

6.2.2 Product Bundles

In cases where the requesting OS wants to activate all the CFSs in support of a given product bundle, the requesting OS needs to send a separate service activation request for each product instance that comprises the given product bundle.

Further, Version 1.0 of the SAI only supports requests for one product at a time. Requests concerning multiple products (not necessarily part of a bundle) could be included in a future version

7 References

7.1 References

[1] TMF518_NRA, Network Resource Assurance BA.

[2] TMF513, Multi-Technology Network Management (MTNM) Business Agreement, Version 3.1, March 2007

[3] TMF517, Multi-Technology Operations System Interface (MTOSI) Business Agreement, Version 1.2, December 2006

[4] TMF612_SA, Service Activation IA

[5] TMF864_SA, Service Activation IIS

[6] SD0-1, Dictionary

7.2 Source or use

The various sources for the requirements in this document are listed in the “Source” field of each requirement.

The following documents make use of this document:

- TMF612_SA, *Service Activation IA*
- TMF864_RPM, *Service Activation IIS*

7.3 IPR Releases and Patent Disclosure

There are no known IPR claims on the material in this document. As per the TM Forum bylaws, any TM Forum member company that has IPR claims on this or any TM Forum specification needs to make the claims known to the TM Forum membership immediately.

8 Administrative Appendix

This Appendix provides additional background material about the TM Forum and this document.

8.1 About this document

This document has been generated from the SD0-3_Template_BA.dot Word template.

8.2 Use and Extension of a TM Forum Business Agreement

This document defines the business problem and requirement model for the Service Activation Interface. The Business Agreement is used to gain consensus on the business requirements for exchanging information among processes and systems in order to solve a specific business problem. The Business Agreement should feed the development of Information Agreement(s), which is a technology-neutral model of one or more interfaces. While the Business Agreement contains sufficient information to be a "stand alone" document, it is better read together with the Information Agreement document TMF612_SA when the Information Agreement is available. Reviewing the two documents together helps in gaining a full understanding of how the technology neutral information model solution is defined for this requirement model. An initial Business Agreement may only deal with a subset of the requirements. It is acceptable for subsequent issues of the document to add additional requirements not addressed by earlier releases of the BA. Business Agreements are the basis for requirement traceability for information models.

It is expected that this document will be used:

- As the foundation for a TM Forum Information Agreement(s)
- To facilitate requirement agreement between Service Providers and vendors
- As input to a service Provider's Request for Information / Request for Proposal (RFI/RFP—RFX)
- As input for vendors developing COTS products
- As a source of requirements for other bodies working in this area

8.3 Document History

Version	Date Modified	Description of changes
1.0	September 2007	This is the first version of the document and as such, there are no changes to report.
1.1	May 2008	Consolidation based on review comments from the team

1.2	September 2011	Updated sections 1.1 and 2. Replaced mTOP by MTNM / MTOSI everywhere in the document ○ removed feasibilityFlag and offeredActivationTime from the cfsCreationEvent and the list of response parameters in Table 3-3 of TMF518_SA_2 ○ updated description for “list of Cfs” ○ removed “feasible” from the list of response parameters as it does not appear in the operation signatures in Table 3-3 of TMF518_SA_2 or in the associated XSD ○ removed productName from all the responses in TMF518_SA_2 ○ made a note (in two places) that serviceRequestRef is handled in the message header in the IIS. ○ updated the first sentence of the Behavior statement in Table 3-4 ○ Added a note in Section 3.4.1 that says the various failure events are not exceptions but rather partial failures. – Also, removed the reasonForException parameter in the four failure events since these are being treated as replies not exceptions (and it is the exceptions that have a “reason” field). – Removed serviceRequestRef and productName in both TMF518_SA_1 and SaExceptions.xsd since these are unnecessary. The correction is done via the message header as noted.

8.4 Company Contact Details

Company	Team Member Representative
Telcordia	*Name:* Stephen Fratini *Email:* sfratini@telcordia.com *Phone:* +1 732 699 2226

8.5 Acknowledgments

This document was prepared by the members of the TM Forum

- Stephen Fratini, Telcordia, **Editor** and co-leader of MTOSI SM team
- Michel Besson, Amdocs, co-leader of MTOSI SM team
- Shlomo Cwang, Amdocs, co-leader of MTOSI SM team

Additional input was provided by the following people:

- Nigel Davis, Ciena
- Jessie Jewitt, Ciena
- Gary Munson, AT&T
- Giuseppe Ricucci, Telecom Italia
- Jeff Wheeler, Cisco
- Wudy Wu, Chunghwa Telecom