

Technical Note Concerning Space Data System Standards

|  |
| --- |
| STATE MACHINES FOR SERVICE MANAGEMENT INFORMATION ENTITIES  |

Draft Working Group Technical Note

DOCUMENT CONTROL

|  |  |  |  |
| --- | --- | --- | --- |
| **Document** | **Title and Issue** | **Date** | **Status** |
| CSSA 902.42-TN-0.1 | State Machines for Service Management Information Entities | October 2018 | initial draft  |
|  |  |  |  |

CONTENTS

Section Page

[DOCUMENT CONTROL i](#_Toc525827160)

[CONTENTS ii](#_Toc525827161)

[1 Introduction 1-1](#_Toc525827162)

[1.1 Purpose of this Tech Note 1-1](#_Toc525827163)

[1.2 Background 1-1](#_Toc525827164)

[1.2.1 SCCS-SM Blue-1 (2009) 1-1](#_Toc525827165)

[1.3 Scope 1-8](#_Toc525827166)

[1.4 Document Organization 1-8](#_Toc525827167)

[1.5 References 1-8](#_Toc525827168)

[2 General Concepts for State Machines 2-1](#_Toc525827169)

[2.1 General 2-1](#_Toc525827170)

[2.2 Considerations regarding usage of State Machines 2-1](#_Toc525827171)

[2.3 Used Terminology and Conventions 2-2](#_Toc525827172)

[3 State Machines for Service Management Information Entities 3-4](#_Toc525827173)

[3.1 Service Package Request 3-4](#_Toc525827174)

[3.1.1 State Machine 3-4](#_Toc525827175)

[3.1.2 State Machine Behaviour 3-4](#_Toc525827176)

[3.1.3 Representation of State Machine within Information Entity 3-1](#_Toc525827177)

[3.2 Service Agreement 3-1](#_Toc525827178)

[3.2.1 State Machine 3-1](#_Toc525827179)

[3.2.2 State Machine Behaviour 3-2](#_Toc525827180)

[3.2.3 Representation of State Machine within Information Entity 3-2](#_Toc525827181)

[3.3 Configuration Profile 3-2](#_Toc525827182)

[3.3.1 State Machine 3-2](#_Toc525827183)

[3.3.2 State Machine Behaviour 3-3](#_Toc525827184)

[3.3.3 Representation of State Machine within Information Entity 3-3](#_Toc525827185)

[3.4 Event Sequence 3-3](#_Toc525827186)

[3.4.1 State Machine 3-4](#_Toc525827187)

[3.4.2 State Machine Behaviour 3-4](#_Toc525827188)

[3.4.3 Representation of State Machine within Information Entity 3-4](#_Toc525827189)

[3.5 Trajectory 3-5](#_Toc525827190)

[3.5.1 State Machine 3-5](#_Toc525827191)

[3.5.2 State Machine Behaviour 3-5](#_Toc525827192)

[3.5.3 Representation of State Machine within Information Entity 3-6](#_Toc525827193)

[3.6 Service Package 3-6](#_Toc525827194)

[3.6.1 State Machine 3-6](#_Toc525827195)

[3.6.2 State Machine Behaviour 3-7](#_Toc525827196)

[3.6.3 Representation of State Machine within Information Entity 3-1](#_Toc525827197)

[ANNEX A Reference Bookmarks (to be deleted) A-1](#_Toc515535804)

Figure Page

[Figure 1‑1 SM Blue-1 - Two Phase Operation State Machine 1-2](#_Toc526149684)

[Figure 1‑2 SM Blue-1 Three Phase Operation State Machine 1-2](#_Toc526149685)

[Figure 1‑3 SM Blue-1 Service Package State Machine 1-3](#_Toc526149686)

[Figure 1‑4 SM Blue-1 Service Package State Machine - Nested in Established State 1-3](#_Toc526149687)

[Figure 1‑5 SM Blue-1 Service Package State Machine - Nested in Planned State 1-4](#_Toc526149688)

[Figure 1‑6 SM Blue-1 Configuration Profile State Machine 1-5](#_Toc526149689)

[Figure 1‑7 SM Blue-1 Trajectory Prediction State Machine 1-6](#_Toc526149690)

[Figure 2‑1 State Machine Example 2-2](#_Toc526149691)

[Figure 3‑1 State Machine Diagram for Service Package Request 3-4](#_Toc526149692)

[Figure 3‑2 State Machine Diagram for Service Agreement 3-1](#_Toc526149693)

[Figure 3‑3 State Machine Diagram for Configuration Profile 3-1](#_Toc526149694)

[Figure 3‑4 State Machine Diagram for Event Sequence 3-1](#_Toc526149695)

[Figure 3‑5 State Machine Diagram for Trajectory Prediction 3-1](#_Toc526149696)

[Figure 3‑6 State Machine Diagram for Service Package 3-1](#_Toc526149697)

Table Page

[Table 1‑1 Service Management Bleu-1 Service Package State Transition Table (Part 1) 1-4](#_Toc526148934)

[Table 1‑2 Service Management Bleu-1 Service Package State Transition Table (Part 2) 1-5](#_Toc526148935)

[Table 2‑1 Operations and Notifications List 2-2](#_Toc526148936)

[Table 3‑1 State Machine Transition Table for Service Package Request 3-1](#_Toc526148937)

[Table 3‑2 State Machine Transitions for Service Package 3-1](#_Toc526148938)

# Introduction

The Cross Support Services Area (CSSA) is developing a series of Recommended Standards for Service Management. Some of these standards describe Information Entities, which shall be exchanged between User and Provider in context of cross support. Each of these Information Entities has its own lifecycle, which in turn imposes specific states. Some of the Information Entities (will) have states, which are explicitly provided and exchanged between User and Provider, whereas other states are rather implicit and not exposed to the interface. Therefore it can be assumed, that each of Information Entities – and especially its behavior – is controlled by a specific state machine.

## Purpose of this Tech Note

The purpose of this Tech Note is to collect information and ideas related to the state machines of Information Entities. This State Machines shall help then keeping track of interaction between Information Entities, and later on shall be used for definition of automation services and facilitate definition of operations and notifications related to Information Entities.

## Background

The new Service Management development is based on the idea to get released single Information Entity blue books in front, before actually developing the complex interaction patterns between them, and getting ultimately standardized automation services. The assumption (which in many aspects has proven correct) was that the Information Entity standards could be released faster and independent from each other, thus facilitating the implementation by Agencies.

While developing new Service Management standard series, and especially Service Management Utilization Request Formats (SMURF, reference [X]) together with Service Package Data Format (SPDF, reference [X]), it showed up relatively quickly, that it is very hard to develop the content of the Information Entity completely decoupled from its actual usage or better to say lifecycle. Eventually we started to make assumptions how the Request or Service Package actually will behave, which imposed specific class construction or definition of class parameters. This led finally to the decision, to get the actual State Machine for each Information Entity, and so get better overview of eventual dependencies in the behavior.

### SCCS-SM Blue-1 (2009)

The Space Communication Cross Support Service Management (SCCS-SM) Service Specification (reference [5]), published in 2009, included detailed description of lifecycle and operations performed on each Information Entity, which also included State Machines. Because the Standard included its own Document Exchange Protocol, also very simple communication stubs (like Invocation or Notification messages) got their State Machines.

It is to be noted, that the state machines used in Blue-1 show nesting of states in few cases (prominently for Service Package) thus making the transition rules respectively complex.

The below summary collects most of the state machine diagrams from SM-1, for reference and easier comparison with new state machines.



Figure 1‑1 SM Blue-1 - Two Phase Operation State Machine



Figure 1‑2 SM Blue-1 Three Phase Operation State Machine



Figure 1‑3 SM Blue-1 Service Package State Machine



Figure 1‑4 SM Blue-1 Service Package State Machine - Nested in Established State



Figure 1‑5 SM Blue-1 Service Package State Machine - Nested in Planned State

Table 1‑1 Service Management Bleu-1 Service Package State Transition Table (Part 1)



Table 1‑2 Service Management Bleu-1 Service Package State Transition Table (Part 2)





Figure 1‑6 SM Blue-1 Configuration Profile State Machine



Figure 1‑7 SM Blue-1 Trajectory Prediction State Machine

## Scope

*This Tech Note: identifies and documents the minimal set off information requirements needed to allow …(what should I write here actually?)*

## Document Organization

Section 2 describes the general concepts for State Machines in context of Cross Support Service Management

Section 3 Describes the State Machines for each Information Entity considered.

## References

The following documents are referenced in this Technical Note. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Technical Note are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS documents.

[1] *Service Management Utilization Request Formats* Recommendation for Space Data System Standards, CCSDS 902.1-W-0.4. White Book. Issue 0.4. Washington, D.C.: CCSDS, April 2017.

[2] *Space Communication Cross Support – Architecture Description Document*. Report Concerning Space Data System Standards, CCSDS 901.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, November 2013.

[3] *Extensible Space Communication Cross Support – Service Management - Concept*. Report Concerning Space Data System Standards, CCSDS 902.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, September 2014.

[4] *Cross Support Transfer Services – Monitored Data Service*. Recommendation for Space Data System Standards, CCSDS 922.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, (forthcoming).

[5] *Space Communication Cross Support - Service Management – Service Specification*. Recommendation for Space Data System Standards, CCSDS 910.11-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, August 2009.

[6] *Simple Schedule Formats*. Recommendation for Space Data System Standards, CCSDS 902.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, [forthcoming].

[7] Functional Resource Tech Note.

[8] IOAG Service Catalog #1.

[9] *Service Package Data Format*. Draft Recommendation for Space Data System Standards, CCSDS 9xx.x-W-0. White Book. Issue 0. CSSMWG Draft, May 2017.

[10] SANA Functional Resources Candidate Registry. <https://www.sanaregistry.org/r/functional_resources>. Updated 30 May 2017.

[11] *Space Communication Cross Support Service Management -* *Service Agreement and Configuration Profile*. Draft Recommendation for Space Data System Standards, CCSDS 902.x-W-0.3. White Book. Issue 0.3. CSSMWG Draft, November 2015.

[12] *Cross Support Reference Model – Part 1: Space Link Extension Services*. Recommendation for Space Data System Standards, CCSDS 910.4-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, October, 2005.

# General Concepts for State Machines

## General

State machine diagrams show the dynamic behavior of an entity based on its response to events, showing how the entity reacts to events depending on its state. They can specify the behavior of an interaction from the point of view of one partner in the interaction. From the general concept of SM-2, the state of each Information Entity is being kept principally on provider side. Also mainly the state of the Service Package is the one being explicitly exchanged (provided) over the SM interface, and being of importance for the user. Other Information Entities – at least as of writing the Tech Note – keep their lifecycle and respective states hidden to the user. It is still to be decided, if in future, depending on the range of the automation services, the states of other Information Entities be exposed to the user.

## Considerations regarding usage of State Machines

Various Information Entities go through various lifecycles and some require actual State Machines. Following list of IE’s has been considered:

* Service Package
* Service Package Request (New / Delete…)
* Report Request
* Info Request
* Submission Request
* Planning Info Request
* Configuration Profile
* Trajectory
* Event Sequence
* Service Agreement
* Planning Information
* Reports (i.e. Simple Schedule)

Analysis of above list resulted in following Information Entities considered as „one-way“ or „one-time-event“, thus having no actual lifecycle or state machine:

* Report Request
* Info Request
* Submission Request
* Planning Info Request
* Reports (i.e. Simple Schedule)
* Planning Information

## Used Terminology and Conventions

The following diagrams demonstrate the UML conventions used in the state machine diagrams.

Figure E-6 shows the same events have different effects depending on the state they arrive in, and possibly on ‘guard conditions’. For example, if the customer selects a drink before inserting enough money to pay for it, the machine transitions to the ‘Selected’ state, whereas if enough money has already been inserted the machine immediately starts dispensing. An event may cause a return to the same state, shown as an arrow returning to the state it starts from.



Figure 2‑1 State Machine Example

The following table tries to collect all possible operations and notifications that may have influence on Information Entities considered. The names of these operations and notifications are selected arbitrarily as of writing of this Tech Note, and will be updated respectively if needed.

Table 2‑1 Operations and Notifications List

|  |  |  |  |
| --- | --- | --- | --- |
| Acronym / Short Name | Name / Description | Type | Information Entity |
| CSP | Create Service Package | Operation | Service Package |
| SASP | Select Alternative Service Package | Operation | Service Package |
| RSP | Replace Service Package | Operation | Service Package |
| SPAR | Service Package Association Request | Operation | Service Package |
| NSPR | New Service Package Request | Operation | Service Package Request |
| RSPR | Replace Service Package Request | Operation | Service Package Request |
| DSP | Delete Service Package | Operation | Service Package |
| DSPR | Delete Service Package Request | Operation | Service Package Request |
| SPR-D | Service Package Request Deleted | Notification | Service Package Request |
| SPR-R | Service Package Request Rejected | Notification | Service Package Request |
| SPR-E | Service Package Request Ended (successfully) | Notification | Service Package Request |
| SP-SR | Service Package Successful Return | Notification | Service Package |
| SP-FR | Service Package Failed Return | Notification | Service Package |
| SP-C | Service Package Cancelled | Notification | Service Package |
| SP-R | Service Package Replaced | Notification | Service Package |
| SP-EA | Service Package Execution Aborted | Notification | Service Package |
| SP-EF | Service Package Execution Finished | Notification | Service Package |
| SP-ES | Service Package Execution Started | Notification | Service Package |
| SCP | Submit Configuration Profile | Operation | Configuration Profile |
| DCP | Delete Configuration Profile | Operation | Configuration Profile |
| ICP | Inactivate Configuration Profile | Operation | Configuration Profile |
| ACP | Activate Configuration Profile | Operation | Configuration Profile |
| SES | Submit Event Sequence | Operation | Event Sequence |
| DES | Delete Event Sequence | Operation | Event Sequence |
| IES | Inactivate Event Sequence | Operation | Event Sequence |
| AES | Activate Event Sequence | Operation | Event Sequence |
| STR | Submit Trajectory | Operation | Trajectory |
| DTR | Delete Trajectory | Operation | Trajectory |

# State Machines for Service Management Information Entities

## Service Package Request

Service Package Request (SPR) is an Information Entity specified in SMURF (reference [X]). In terms of online and offline SPR, the state machine stays the same, thus the considerations below are valid for both cases.

The lifecycle (and thus also the state machine) of the SPR is currently considered to be hidden to the user (requestor). Therefore, there is no state representation in the SPR File Format. The lifecycle/state machine presented here is rather meant to support the upkeep of the request information by the provider.

The state machine assumes that the Replacement of the Service Package Request (RSPR operation) actually replaces/updates the content of the SPR (unlike the Service Package, where replacement causes actual deletion of current SP and creation of new SP with new ID). This assumption simplifies the state machine.

### State Machine

The state machine diagram is provided below. The entry to the state machine is triggered by the New Online (or Offline) Service Package Request (an entity from SMURF Book), here shortly defined as NSPR. The request is being checked formally (mainly conformance to the XSD, authorization and if it belongs to the valid Service Agreement). In case this check is successful, the Service Package Request (SPR) is actively present in provider system (is being *Executed*).

The SPR in its Executing state generates CSP (Create Service Package) operations. Independently on the result of these operations (Successful or Failed Return) the state of SPR does not change.

In case of the submission of the Replace Service Package Request (RSPR) (also called Replace “Service Package Request” Request) the state of the SPR changes to *Rep\_Submitted* to check if new SPR included in RSPR is valid (similarly to the initial check). In case it is not, the state gets back to the Executing and the processing continues. In case the new SPR is valid, it changes the state again to *Rep\_Accepted*. This state is used to hold the processing of old/previous SPR and in this time delete all remaining Service Packages (remaining means ones, which are not yet executed). As soon all of the remaining Service Packages belonging to previous SPR are deleted, the information from new SPR is being activated and the state gets back to *Executing*. The new SPR contents are now used to provide/generate SP’s.

The lifecycle of the SPR ends within two possible situations. Either it is actively deleted by the user (via Delete “Service Package Request” Request, DSPR) or all of the requested Service Packages have been executed (which in that terms is a kind of timeout).



Figure 3‑1 State Machine Diagram for Service Package Request

### State Machine Behavior

The state transition table represents the state machine behavior.

Table 3‑1 State Machine Transition Table for Service Package Request

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| StateEvent | Initial State | R\_Submitted | R\_Executing | R\_Rep\_Submitted | R\_Rep\_Accepted | R\_Rejected | R\_Delted | R\_Archived |
| NSPR issued | 🡪 SUBMITTED | x | X | X | X | X | X | x |
| NSPR rejected | X | 🡪 REJECTED | X | X | X | X | X | X |
| NSPR successfully processed | X | 🡪 EXECUTING | X | X | X | X | X | X |
| CSP successful | X | X | Service Package created🡪 no change | X | X | X | X | X |
| CSP failed | X | X | Service Package not created🡪 no change | X | X | X | X | X |
| DSPR | X | X | All related, remaining (not executed yet) Service Packages need to be deleted by DSP operation.🡪 DELETED | X | X | X | X | X |
| Last SP from the SP Request range executed, cancelled or deleted | X | X | No remaining Service Packages left🡪 ARCHIVED | X | X | X | X | X |
| RSPR issued | X | X | 🡪 REP\_SUBMITTED | X | X | X | X | X |
| RSPR rejected | X | X | X | Replacement request rejected, get back to execution of previous one🡪 EXECUTING | X | X | X | X |
| RSPR accepted | X | X | X | Replacement is accepted,🡪 REP\_ACCEPTED | X | X | X | X |
| Deletion of all SP’s related to previous Request (via DSP) finished | X | X | X | X | 🡪 EXECUTING | X | X | X |
| SP Request deleted (notification) (SPR-D) | X | X | X | X | X | X | 🡪 Final | X |
| SP Request rejected (notification SPR-R) | X | X | X | X | X | 🡪 Final | X | X |
| SP Request finished operation (SPR-E) | X | X | X | X | X | X | X | 🡪 Final |

### Representation of State Machine within Information Entity

As of writing of this Tech Note, the state machine has no representation within the Information Entity format (XML).

## Service Agreement

Service Agreement (SA) is an Information Entity specified in Configuration Profile and Service Agreement book (reference [X]).

The state of the SA is potentially important to both user and provider, as it constitutes the basis for all other processes and interaction. The SA format is not yet developed, thus there is no explicit representation. This shall be considered when producing the SA format.

### State Machine

Operations or notifications related to SA are not yet defined, and not anticipated as of writing of this Tech Note.

The initial state of the SA would be *Draft*, and shall be implied as soon provider is generating the initial version/instance of SA. The SA stays in that state as long user and provider exchange the SA itself, and fills with information. First, when both parties agree, that the SA is complete, its status moves to *Agreed*. As each SA shall have, some validity period defined, first when this time period is reached, the status changes again into *Active*. Similarly, when the agreement period end time is reached, the status changes to *Finished*. One of the parties may cancel the SA from all states (*Draft*, *Agreed* or *Active*), which would change the state to *Cancelled*.



Figure 3‑2 State Machine Diagram for Service Agreement

### State Machine Behaviour

The state transition table represents the state machine behavior.

[TBD]

### Representation of State Machine within Information Entity

As of writing of this Tech Note, the state machine has no representation within the Information Entity format (XML). Actually, the Information Entity definition is not yet available.

## Configuration Profile

Configuration Profile (CP) is an Information Entity specified in Configuration Profile and Service Agreement book (reference [X]).

The lifecycle (and thus also the state machine) of the CP is currently considered to be hidden to the user (requestor). According to the below state machine, it may be considered to be helpful to yet provide the status between user and provider. In such a case the respective representation in CP file format and notifications shall be introduced.

### State Machine

The lifecycle of Configuration Profile begins with its submission by user to the provider system. The CP gets state *Submitted* and is being checked/validated. The CP is checked against XSD, validity in terms of Service Agreement and eventually technical plausibility (existence of respective Functional Resources). The CP shall also be complete at that time point (all obligatory parameters need to be set). If the validation fails, the CP is being *Rejected*. If the validation is correct, the CP gets state *Active* (and so available to the Mission for referencing in Service Packages).

As shown on the diagram below, the possibility to deactivate and activate back of the CP, as well as deletion, is provided. It is here not defined, if the activation/deactivation or deletion is triggered by user or provider or both.



Figure 3‑3 State Machine Diagram for Configuration Profile

### State Machine Behaviour

The state transition table represents the state machine behavior.

[TBD]

### Representation of State Machine within Information Entity

As of writing of this Tech Note, the state machine has no representation within the Information Entity format (XML). Actually, the Information Entity definition is not yet available.

## Event Sequence

Event Sequence (ES) is an Information Entity specified in Space Link Event Sequence Data Format book (reference [X]).

The lifecycle (and thus also the state machine) of the ES is currently considered to be hidden to the user (requestor). According to the below state machine, it may be considered to be helpful to yet provide the status between user and provider. In such a case the respective representation in ES file format and notifications shall be introduced.

### State Machine

The lifecycle of Event Sequence starts with its initial submission by the user to the provider system. The ES is checked against XSD, validity in terms of Service Agreement. In case of failed validation, the ES gets *Rejected*. In case the ES is complete (includes all information) it reaches the state of *Active*. In case the ES needs to be iterated between user and provider, it may stay in *Incomplete* state. The ES shall also be complete at that time point (all obligatory parameters need to be set).

As shown on the diagram below, the possibility to deactivate and activate back of the ES, as well as deletion, is provided. It is here not defined, if the activation/deactivation or deletion is triggered by user or provider or both.

Due to the fact, that the ES entity operates on time related information, there might be cases, where some included time absolute values expire. In that case the ES reaches the state of *Outdated* and can’t be used anymore in operation.



Figure 3‑4 State Machine Diagram for Event Sequence

### State Machine Behaviour

The state transition table represents the state machine behavior.

[TBD]

### Representation of State Machine within Information Entity

As of writing of this Tech Note, the state machine has no representation within the Information Entity format (XML).

## Trajectory

The Trajectory is an information Entity that is being provided by external format (not a part of SM). Therefore, it is not possible to keep the track of the status of state machine internally within the format. Generally, the state machine of the trajectory is considered relatively simple. As the trajectory predictions typically have limited usability (especially the TLE files, which loose on precision in time).

The lifecycle (and thus also the state machine) of the TR is currently considered to be hidden to the user (submitter).

### State Machine

The lifecycle of Trajectory begins with its submission by user to the provider system. The TR gets state *Submitted* and is being checked/validated (*here is to clarify how far we actually can do that – keeping in mind we actually take Trajectory Prediction as a text block, and not really being SM Format).* Whatever the check is, the Trajectory may get *Rejected* or *Active* (*if we are not going to check anything on Trajectory, than it would get Active always. Than it would be a question, what happens if for example the TLE provided couldn’t be actually ingested into the ACU of the antenna shortly before execution of a pass?*).

The trajectory prediction as such does not expire, however it may get unprecise in time. This can’t be checked by the provider however. Therefore, the only way to remove the older TR is an action executed by the user (Delete Trajectory Prediction, DTR).



Figure 3‑5 State Machine Diagram for Trajectory Prediction

### State Machine Behaviour

The state transition table represents the state machine behavior.

[TBD]

### Representation of State Machine within Information Entity

As of writing of this Tech Note, the state machine has no representation within the Information Entity format (XML).

## Service Package

Service Package (SP) is an Information Entity specified in Service Package Data Format book (reference [X]).

The lifecycle (and thus also the state machine) of the SP is currently considered as the main state which is being exchanged between user and provider side during Service Agreement period.

The state machine assumes that the Replacement of the Service Package Request (RSPR operation) actually replaces/updates the content of the SPR. This in turn results in deletion of the Service Packages associated to the old Request, and generation of new Service Packages associated to new Request. From the Service Package point of view, any replacement of the associated SP Request means end of life for the SP itself.

The specialty of the SP is that there may be multiple SP’s associated to each other via scenarios context. This may result in multiple SP’s being in alternative to the one being actually scheduled or executing.

The presented state machine does not differentiate between online and offline Service Package, however it is to be noted, that the offline SP won’t use *Alternative* state. Also the use of Scheduled state may be limited, and so the offline SP would change more or less directly from *Created* to *Executing*. This needs to be fleshed in detail later on.

### State Machine

The state machine of SP begins with Create Service Package operation (which is being issued by the mechanisms of active – *Executing* Service Package Request). The SP is being checked according to XSD, plausibility, completeness etc… In case of failure, the SP Failed Return with a given reason is being generated, and the SP is set to Rejected.

Due to the fact, that SP may not actually get scheduled for some time, and may stay in Created state, the Replace Service Package or Delete Service Package (invoked by user) or Service Package Replaced (invoked by provider) may have an influence on the SP. In case of replacement, the state does not change, in case of deletion, the state changes to *Cancelled*.

The Service Package Association Request (SPAR) can also trigger change of the state of SP from either *Created* or *Scheduled* into *Alternative* (Alternate?). In addition, the Select Alternative Service Package (SASP) operation can change the state of Scheduled or Executing SP’s into *Alternative*, and by picking the new scenario also moves one respectively into *Scheduled* or *Executing* state (*It is to clarify here if, especially in* Executing *state, it is possible to pick alternative Scenario and set it to* Executing*, without any additional check. It has to be guaranteed, that the alternative scenario SP is actually plausible and for example the start and stop times do not violate the global schedule*).

Replacement of the Service Package (via RSP operation) is assumed not to violate general boundaries of the SP (i.e. timing is the same). Therefore, it allows after successful submission, to get back to the original state (*Created* or *Scheduled* respectively). The same is the case of the changes performed by the provider in frame of flexibility given by the user within the SPR. In such case, the Service Package Replaced notification is generated, and the actual state of the SP stays. Please note, that RSP and SP-R are not allowed when SP is *Alternative* or *Executing*.

As soon the SP finishes its operation, the SP Execution Finished (SP-EF) notification is given, and the state changes to *Archived*. In case of some problems, the provider may abort execution, which leads to SP-EA (Execution Aborted) notification, and status change to *Aborted* (*it is to clarify, if the user may also perform some “abort” operation, and what would be the actual use case for that*).



Figure 3‑6 State Machine Diagram for Service Package

### State Machine Behaviour

 The state transition table represents the state machine behavior.

Table 3‑2 State Machine Transitions for Service Package

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| StateEvent | Initial State | Created | Scheduled | Alternative | Executing | Rejected | Cancelled | Aborted | Archived |
| CSP | 🡪 CREATED | X | X | X | X | X | X | X | x |
| DSP | X | 🡪 CANCELLED | 🡪 CANCELLED | 🡪 CANCELLED | X | X | X | X | X |
| SASP | X | X | 🡪 ALTERNATIVE | If not yet executing (time)🡪 SCHEDULEDIf already executing (time)🡪 EXECUTING | 🡪 ALTERNATIVE | X | X | X | X |
| CSP successfulSP-SR notification | X | If only one or main scenario🡪 SCHEDULEDIf alternative scenario🡪 ALTERNATIVE | X | X | X | X | X | X | X |
| CSP failedSP-FR notification | X | 🡪 REJECTED | X | X | X | X | X | X | X |
| SP-C notification | X | X | 🡪 CANCELLED | 🡪 CANCELLED | X | X | X | X | X |
| SP-ES notification, timeout | X | X | 🡪 EXECUTING | X | X | X | X | X | X |
| SP-EA execution aborted | X | X | X | X | 🡪 ABORTED | X | X | X | X |
| SP-EF execution finished, timeout | X | X | X | X | 🡪 ARCHIVED | X | X | X | X |
| Notification delivered | X | X | X | X | X | 🡪 Final | 🡪 Final | 🡪 Final | 🡪 Final |

### Representation of State Machine within Information Entity

The current version of the SPDF Book provides two classes, which to some extent carry the information on the status. This, however, is still based on concepts from SM-1, thus not reflecting the actual new state machine of the Service Package.

Chapter 3.1.5 of SPDF (reference X) describes ServicePkgResultDetails class. This class includes two parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| servicePackageResultStatus | Optional parameter.This is used to indicate the status of the service package in the service package state machine. Possible values are:-‘TENTATIVELY SCHEDULED’-‘PARTIALLY SCHEDULED’-‘FULLY SCHEDULED’ | Enum | n/a |
| servicePackageResultType | Optional parameter.This is used to provide context as to the reason why the result is was generated:-‘NEW’: used for results generated by a new service package request-‘UPDATE’: used for results generated by an update service package request-‘ALTERNATE’: used for results generated by a select alternate service package request-‘FLEXIBILITY CHANGE’: used for results generated by a service package change initiated by the service provider within flexibility of the request | Enum | n/a |

Additionally, there is a class ServicePkgReadinessStatus (Chapter 3.1.6):

| **Parameter** | **Description** | **Data Type** | **Data Units** |
| --- | --- | --- | --- |
| servicePackageReadyThresholdTime | Mandatory parameter.The latest time (in UTC) at which all elements required for the execution of the SLS Service Package can be defined or redefined. | CCSDS ASCII Time Code B (reference [7]) | UTC  |
| servicePackageReadinessStatus | Mandatory parameter.Summary status of the readiness of the Service Package for execution. The values are:– ‘READY’: All required Service Package items are currently defined;– ‘NOT READY’: One or more required Service Package items must still be defined. | Enum | n/a |

Both are considered for major rework.

One suggestion would be to use only one parameter called *servicePackageStatus* (service Package Result was a result of respective name for the Information Entity. In the current SM we do not have Service Package Result as such, but we have just Service Package). The *servicePackageStatus* could take the states as defined in previous sections.

Due to the fact, that in current SM-2 the assumption is to have Service Package always complete, there would be actually no need for readiness status at all. Therefore this class could actually be removed completely from SPDF.