

RESTRICTIONS ON USE, DUPLICATION OR DISCLOSURE OF   
PROPRIETARY INFORMATION

This document contains information proprietary to Sandvine Incorporated, to its affiliates or to a third party to which Sandvine Incorporated may have a legal obligation to protect such information from unauthorized disclosure, use or duplication. Any disclosure, use or duplication of this document or any of the information herein for other than the specific purpose for which it was disclosed by Sandvine Incorporated is expressly prohibited, except as Sandvine Incorporated may otherwise agree in writing. Recipient by accepting this document agrees to the above stated conditional use of this document and the information disclosed herein.

S9-SDD

Raghavendra Chikmagalur Eswarappa [reswarappa@sandvine.com](mailto:reswarappa@sandvine.com)

Viswanath Deepak [vdeepak@sandvine.com](mailto:vdeepak@sandvine.com)

Shailesh Devaraj [sdevaraj@sandvine.com](mailto:sdevaraj@sandvine.com)

Rajesh Kumar Ghildiyal [rghildiyal@sandvine.com](mailto:rghildiyal@sandvine.com)

Arjun Sachdeva asachdeva@sandvine.com

Richa Saxena [rsaxena@sandvine.com](mailto:rsaxena@sandvine.com)

Rajeshwari Malennanavar [rmalennanav@sandvine.com](mailto:rmalennanav@sandvine.com)

30-Jan-17

reqst00541896

**Table of Contents**

1 Introduction 6

1.1 Purpose 6

1.2 Acronyms and Terms 6

1.3 Applicable Documents and References 6

1.4 Revision History 6

1.5 Outstanding Items 7

2 System Overview 8

2.1 System Overview 8

2.2 S9 Message flows 13

2.3 System Requirements 17

2.4 Backward Compatibility 17

2.5 Design Constraints 17

3 High Level Design 18

3.1 System Interfaces 18

3.1.1 V-PCRF 18

3.1.1.1 System interaction 18

3.1.1.2 GxServer Subsystem 18

3.1.1.2.1.1 Modifications to existing Gx Workflows 20

3.1.1.2.1.2 Multi-Gx support 23

3.1.1.2.1.3 Overlapping IP (Missed logout) support 25

3.1.1.2.1.4 Deferred IP allocation support 25

3.1.1.3 S9ClientSubsystem 25

3.1.1.3.1 S9Client workflow to handle S9 CCR procedure 26

3.1.1.3.2 S9Client workflow to handle RAR procedure 28

3.1.1.3.3 QoS Authorization/Validation at S9Client 28

3.1.1.3.4 Gx Diameter Message to S9 Diameter Message 34

3.1.1.4 RxServer Subsystem 37

3.1.1.4.1 RxAFSessionEstablishment 37

3.1.1.4.2 RxAFSessionModification 40

3.1.1.4.3 RxAFSessionTermination 42

3.1.1.4.4 RxAFSessionReauthorization 43

3.1.1.4.5 RxAFSessionAbort 46

3.1.1.5 Handling Origin-State-ID change of AF 47

3.1.2 H-PCRF 49

3.1.2.1 System interaction 49

3.1.2.2 Service Designer 49

3.1.2.2.1 Installation of PCC Rules on S9 49

3.1.2.2.1.1 Design Option 1 – Mapping of S9Subsession to Gx access type 50

3.1.2.2.1.2 Design Option 2 – Reuse Gx Traffic & Session Rules 52

3.1.2.2.1.3 Design Option 3 – New S9 Traffic & Session Rules 54

3.1.2.3 GxServer Subsystem 56

3.1.2.4 S9ServerSubsystem 57

3.1.2.4.1 S9ServerSessionEstablishment 57

3.1.2.4.1.1 Overlapping IP (Missed logout) support 59

3.1.2.4.1.2 Deferred IP allocation support 60

3.1.2.4.2 S9ServerSessionModification 61

3.1.2.4.2.1 Overlapping IP (Missed logout) support 62

3.1.2.4.2.2 Deferred IP allocation support 62

3.1.2.4.3 S9ServerSessionTermination 63

3.1.2.4.4 S9ServerSubSessionWorkFlow 68

3.1.2.4.5 S9ServerSessionReauthorization 72

3.1.2.5 S9 Server Sanity Step 76

3.1.2.6 RxServer Subsystem 77

3.2 System CLI 78

3.3 System Alarms 87

3.4 System Configuration and Installation 91

3.4.1 Packaging 91

3.4.2 S9 Policy Packages 91

3.4.3 Configuration 93

3.5 Session Model 95

3.5.1 Design Considerations 96

3.5.2 Design Options. 96

3.5.2.1 Independent Subsession Model – Created Subsession ID 96

3.5.2.1.1 New configuration CLIs 98

3.5.2.1.2 Session CLI Modifications. 98

3.5.2.1.3 Policy Package. 101

3.5.2.2 Independent Subsession Model – Compound Key. 101

3.5.2.3 True Subsession Model 102

3.5.2.4 Other Session Models 106

3.6 Event Triggers 106

3.6.1 Use Cases 107

3.6.2 Design Considerations 107

3.6.3 Implementation 107

3.7 Logging and Debugging 108

3.8 Reports 115

Table of Figures

[Figure 1: Overview of SDE S9 solution 9](#_Toc473314845)

[Figure 2: S9 Stack Overview 9](#_Toc473314846)

[Figure 3: Network interface & entities in Roaming scenario 10](#_Toc473314847)

[Figure 4: SDE 8.10 System Architecture 12](#_Toc473314848)

[Figure 5: S9 & Rx Session with V-AF (Visited access) 15](#_Toc473314849)

[Figure 6: S9 & Rx Session with H-AF (Visited access) 15](#_Toc473314850)

[Figure 7: Rx Session with V-AF (Visited access) 16](#_Toc473314851)

[Figure 8: AcessSessionInfoS9ClientInterface 19](#_Toc473314852)

[Figure 9: GxIPCanSessionEstablishment 20](#_Toc473314853)

[Figure 10: GxIPCanSessionModification 21](#_Toc473314854)

[Figure 11: GxIPCanSessionTermination 22](#_Toc473314855)

[Figure 12: Session attribute & Config for EnforcementGroup 23](#_Toc473314856)

[Figure 13: Sample policy.conf for multi-Gx support 24](#_Toc473314857)

[Figure 14: S9ClientPcrfInternalTriggerProcessor creating S9Client Workflows 26](#_Toc473314858)

[Figure 15: S9Client Credit Control workflow 28](#_Toc473314859)

[Figure 16: S9Client Reauthorization workflow 28](#_Toc473314860)

[Figure 17: Acceptable QoS 31](#_Toc473314861)

[Figure 18: AppMsgCopy class to copy specific AVPs from Diameter Message 35](#_Toc473314862)

[Figure 19: AppendRawAvp method added in DiameterHeaderWriter class 35](#_Toc473314863)

[Figure 20: Example - Building S9 CCR-I 36](#_Toc473314864)

[Figure 21: RxSessionEstablishment Workflow state machine 40](#_Toc473314865)

[Figure 22: RxSessionModification Workflow state machine 42](#_Toc473314866)

[Figure 23: RxAFSessionTerminationWorkflow state machine 43](#_Toc473314867)

[Figure 24: RxAFSessionReauthorization Workflow state machine at 45](#_Toc473314868)

[Figure 25: RxAFSessionAbort Workflow state machine at 47](#_Toc473314869)

[Figure 26: Handling Origin-State-ID change of AF -1 48](#_Toc473314870)

[Figure 27: Handling Origin-State-ID change of AF - 2 49](#_Toc473314871)

[Figure 28: ServiceDesignerS9Home.default.xml 51](#_Toc473314872)

[Figure 29: New S9-Enforcement-Workflow-Processor 51](#_Toc473314873)

[Figure 30: Exmaple - S9HomeSubsession session type map to Gx access type 52](#_Toc473314874)

[Figure 31: New S9 Access Type 52](#_Toc473314875)

[Figure 32: S9 Access Type Class definition reusing Gx Rules 54](#_Toc473314876)

[Figure 33: S9 Access Type added in Rule-Schema-Manager 54](#_Toc473314877)

[Figure 34: S9 Access Type Class definition with new S9 Rules 56](#_Toc473314878)

[Figure 35: S9ServerSessionEstablishment workflow state machine 57](#_Toc473314879)

[Figure 36: LLD for S9ServerSessionEstablishment workflow 59](#_Toc473314880)

[Figure 37: S9ServerSessionModification workflow state machine 61](#_Toc473314881)

[Figure 38: S9ServerSessionTermination workflow state machine – P1 64](#_Toc473314882)

[Figure 39: S9ServerSessionTermination workflow state machine – P2 65](#_Toc473314883)

[Figure 40: LLD for S9ServerSessionTermination workflow 68](#_Toc473314884)

[Figure 41: S9ServerSubSession workflow state machine – P1 69](#_Toc473314885)

[Figure 42: S9ServerSubSession workflow state machine – P2 70](#_Toc473314886)

[Figure 43: LLD for S9ServerSubSession workflow state machine 72](#_Toc473314887)

[Figure 44: S9ServerEnforcement workflow processor class definition 72](#_Toc473314888)

[Figure 45: Processing of outgoing message received from business rules 74](#_Toc473314889)

[Figure 46: S9ServerEnforcement workflow state machine 75](#_Toc473314890)

[Figure 47: S9ServerEnforcement sub-workflow state machine 75](#_Toc473314891)

[Figure 48: Pseudo code for S9ServerEnforcement workflow 75](#_Toc473314892)

[Figure 49: Pseudo code for S9ServerEnforcement sub-workflow 76](#_Toc473314893)

[Figure 50: S9-Server PDB structure overview 78](#_Toc473314894)

[Figure 51: S9-Client PDB structure overview 78](#_Toc473314895)

[Figure 52: S9-Server config PDB structure 79](#_Toc473314896)

[Figure 53: S9-Client config PDB structure 79](#_Toc473314897)

[Figure 54: S9-Server & S9-Client overall messageRateCounters PDB structure 80](#_Toc473314898)

[Figure 55: S9-Server & S9-Client overall messageCounters PDB structure 81](#_Toc473314899)

[Figure 56: S9-Server & S9-Client overall errorRateCounters PDB structure 82](#_Toc473314900)

[Figure 57: S9-Server & S9-Client overall errorCounters PDB structure 84](#_Toc473314901)

[Figure 58: S9-Server & S9-Client peer messageCounters PDB structure 86](#_Toc473314902)

[Figure 59: S9-Server & S9-Client peer errorCounters PDB structure 87](#_Toc473314903)

# Introduction

## Purpose

The purpose of this document is to describe the design details for supporting 3GPP S9 functionality in Sandvine’s SDE 8.10 solution.

The design details of the following newly added subsystem are described:

* S9 Server (H-PCRF)
* S9 Client (V-PCRF)

Also the modified design details of the following existing subsystem are described (which shall be impacted by S9 functionality):

* Gx Server
* Subscriber
* Session
* Business Rules (Service Designer)
* Rx Server

## Acronyms and Terms

|  |
| --- |
| **Sandvine Wiki link** |
| <https://wiki.sandvine.com/display/SDE/Acronyms+and+terms> |
| <https://wiki.sandvine.com/display/SDE/Definitions> |

## Applicable Documents and References

|  |
| --- |
| **Document Reference** |
| SDE 8.10 S9 MRD - <http://intranet/vobs/Documents/Engineering/Projects/Logs/sde_8_10/Planning/S9-MRDs.xlsx> |
| SDE 8.10 S9 SRS - <http://intranet/vobs/Documents/Engineering/Projects/Logs/sde_8_10/Features/DiameterS9/Requirements/S9-SRS.docx> |
| SDE 8.10 S9 Architecture - <https://wiki.sandvine.com/display/SDE/S9> |
| 3GPP TS 29.215: PCC over S9 reference point (Release 12 v12.8.0)  <http://www.etsi.org/deliver/etsi_ts/129200_129299/129215/12.08.00_60/ts_129215v120800p.pdf> |
| 3GPP TS 23.203: PCC Architecture (Release 12 v.12.11.0)  <http://www.etsi.org/deliver/etsi_ts/123200_123299/123203/12.11.00_60/ts_123203v121100p.pdf> |
| 3GPP TS 29.212: "Policy and Charging Control (PCC); Reference points" (Release 12 v12.13.0)  <http://www.etsi.org/deliver/etsi_ts/129200_129299/129212/12.13.00_60/ts_129212v121300p.pdf> |
| 3GPP TS 29.213: "Policy and charging control signaling flows and Quality of Service (QoS) parameter mapping" (Release 12 v12.12.0)  <http://www.etsi.org/deliver/etsi_ts/129200_129299/129213/12.12.00_60/ts_129213v121200p.pdf> |
| 3GPP TS 29.214: "Policy and charging control over Rx reference point” (Release v12.12.0)  <http://www.etsi.org/deliver/etsi_ts/129200_129299/129214/12.12.00_60/ts_129214v121200p.pdf> |

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason** | **Version** |
| Raghavendra | 12-Dec-2016 | Initial draft | 0.1 |
| Shailesh | 13-Dec-2016 | Added VPCRF Workflow | 0.2 |
| Deepak | 23-Dec-2016 | Added HPCRF Workflow | 0.3 |
| Raghavendra | 30-Dec-2016 | Added Multi-Gx, Overlapping IP & Deferred IP | 0.4 |
| Rajesh | 2-Jan-2017 | Added Rx Workflow | 0.5 |
| Arjun | 2-Jan-2017 | Added Session Model | 0.6 |
| Raghavendra | 4-Jan-2017 | Update for fixing review defects | 0.7 |
| Arjun | 9-Jan-2017 | Updated the session Model | 0.8 |
| Raghavendra | 11-Jan-2017 | Added Installation of PCC Rules for H-PCRF | 0.9 |
| Arjun | 17-Jan-2017 | Updated Event trigger subscription (26th Upload) | 1.0 |
| Raghavendra | 18-Jan-2017 | Updated for H-PCRF RAR Workflow | 1.1 |
| Deepak/ Richa | 18-Jan-2017 | Update H-PCRF CCR-I/ CCR-T Workflow | 1.2 |
| Deepak/Shailesh/  Rajeshwari | 20-Jan-2017 | Updated Gx and S9 client workflows, added S9 diameter dictionary, updated for review comments | 1.3 |

## Outstanding Items

|  |  |  |
| --- | --- | --- |
| **Name** | **Date** | **Outstanding Items** |
| SCTP | March 2017 | S9 Interface is dependent on SCTP to make transport layer connection to peer.  SCTP feature is being developed under SDE 8.00 feature and stated to complete by March 2017 |

# System Overview

## System Overview

The S9 reference point is used to exchange rules information between H-PCRF and V-PCRF. The S9 reference point allows the H-PCRF to control policies of its subscribers when in roaming (home subscribers). The V-PCRF obtains the charging rules associated with the roaming subscribers (visited subscribers) that will be evaluated and enforced on V-PCEF.

In absence of S9 interface, the visited network’s V-PCRF shall use the dynamic/pre-defined policies based on roaming agreement with H-PLMN operator. Note that these policies are not controlled by H-PLMN.

On introduction of S9 reference point in PCRF, the H-PCRF installs the dynamic/pre-defined PCC rules in the V-PCRF, when the subscriber is roaming.

The two main roaming scenarios:

* **Home routed Access**: Where the PCEF is located in the H-PLMN. UE obtains the access to the IP cloud through the PCEF in the H-PLMN
* **Visited Access (Local breakout)**: Where the PCEF is located in the V-PLMN. UE obtains the access to the IP cloud through the PCEF in the V-PLMN

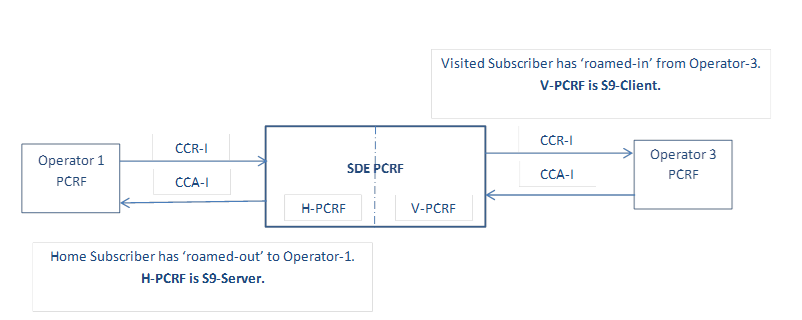
The SDE S9 solution shall have:

* **S9-Server** Subsystem as H-PCRF
* **S9-Client** Subsystem as V-PCRF

SDE can act as both

* S9-Server/S9-Home (for home subscribers in roaming) &
* S9-Client/S9-Visited (for visited subscribers in roaming).

Refer following diagram for clarity:



SDE PCRF is deployed with roaming agreement with Operator-1 and Operator-3 networks.

SDE PCRF is H-PCRF for home subscriber roamed-out to Operator-1 and V-PCRF for visited subscriber roamed-in from Operator-3.

Figure : Overview of SDE S9 solution

The S9 reference point is a **Diameter interface** resides between PCRF in the H-PLMN (H-PCRF) and PCRF in V-PLMN (V-PCRF).

* H-PCRF acts as a diameter server
* V-PCRF acts as the diameter client

S9 uses SCTP/TCP for messaging between V-PCRF and H-PCRF.

Diameter Base

TCP/ SCTP

IP

Ethernet

Physical Layer

S9 App

H-PCRF

Diameter Base

TCP/ SCTP

IP

Ethernet

Physical Layer

S9 App

V-PCRF

**S9**

Figure : S9 Stack Overview

S9 is used to:

1. Install PCC or QOS rules generated in the H-PLMN into the V-PLMN
2. Transport the events that may occur in the VPLMN to H-PLMN
3. Install the ADC rules generated in the H-PLMN into the V-PLMN (Not scope in SDE 8.10)
4. Transport the ADC information from the V-PLMN to H-PLMN, when the H-PCRF and V-PCRF both support the ADC feature (Not scope in SDE 8.10)

V-PCEF

H-PCRF

H-PCEF

SGW

V-PCRF

Gx for IP1

S8 for IP2

Gx for IP2

S5 for IP1

S9 for IP1

V-PLMN

H-PLMN

H-AF

V-AF

Rx

Rx

Rx Messages over Rx Interface

Gx Messages over Gx Interface

S5/S8 Messages over S5/S8 Interface

S9 Messages over S9 Interface

Subscriber IP1 explains visited access roaming scenario, where IP1 access IP cloud via V-PCEF in visited network. While on S9 interface the V-PCRF obtains PCC rules for roaming subscriber from H-PCRF to apply on V-PCEF. The Gx & S5 interfaces are also visited network.

Subscriber IP2 explains home routed access roaming scenario, where IP2 access IP cloud via H-PCEF in home network. The Gx interface is on home network, while on S8 the SGW in visited network & H-PCEF communicate.

Figure 3: Network interface & entities in Roaming scenario

The S9 reference point, involves the following network entities:

* H-PLMN and V-PLMN in Visited access

(PCEF in V-PLMN and AF in V-PLMN OR H-PLMN)

* H-PLMN in Home-routed access

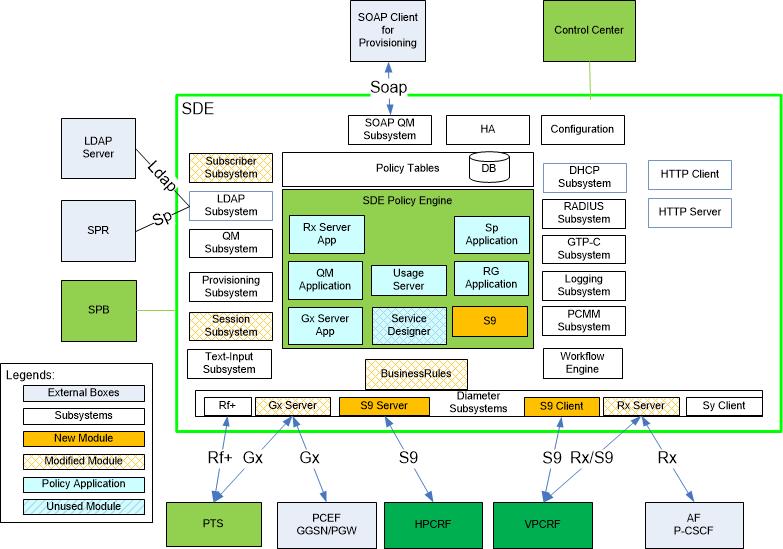
(PCEF in H-PLMN and AF in the H-PLMN)

If AF is present in V-PLMN, Rx protocol is used over S9 reference point to exchange service session information from V-PCRF to H-PCRF.

**S9 Messages** - 3GPP S9 Application (Application & Auth-Application-Id: **16777267**)

| **Message** | **Description** | **Code** | **Direction** | **Application** |
| --- | --- | --- | --- | --- |
| CCR | Credit Control Request | 272 | V-PCRF -> H-PCRF | Request PCC/QoS rules  Indicate PCC/QoS rule related events |
| CCA | Credit Control Answer | 272 | V-PCRF <- H-PCRF | Response to CCR  To provision PCC/QoS rules & event triggers for sub-session/session |
| RAR | Re-Auth Request | 258 | V-PCRF <- H-PCRF | To provision PCC/QoS rules & event triggers for sub-session/session |
| RAA | Re-Auth Answer | 258 | V-PCRF -> H-PCRF | Response to RAR |
| TER | Trigger Establishment Request | 8388656 | V-PCRF <- H-PCRF | To trigger S9 Session establishment |
| TEA | Trigger Establishment Answer | 8388656 | V-PCRF -> H-PCRF | Response to TER |

**Note** – Greyed colored text is not scoped in SDE 8.10.



Rx/S9 implies Rx messages on S9 interface in case of visited access roaming scenarios. If the AF is located in V-PLMN, then the Rx messages are exchanged between V-PCRF and H-PCRF via S9 interface only.

Figure : SDE 8.10 System Architecture

## S9 Message flows

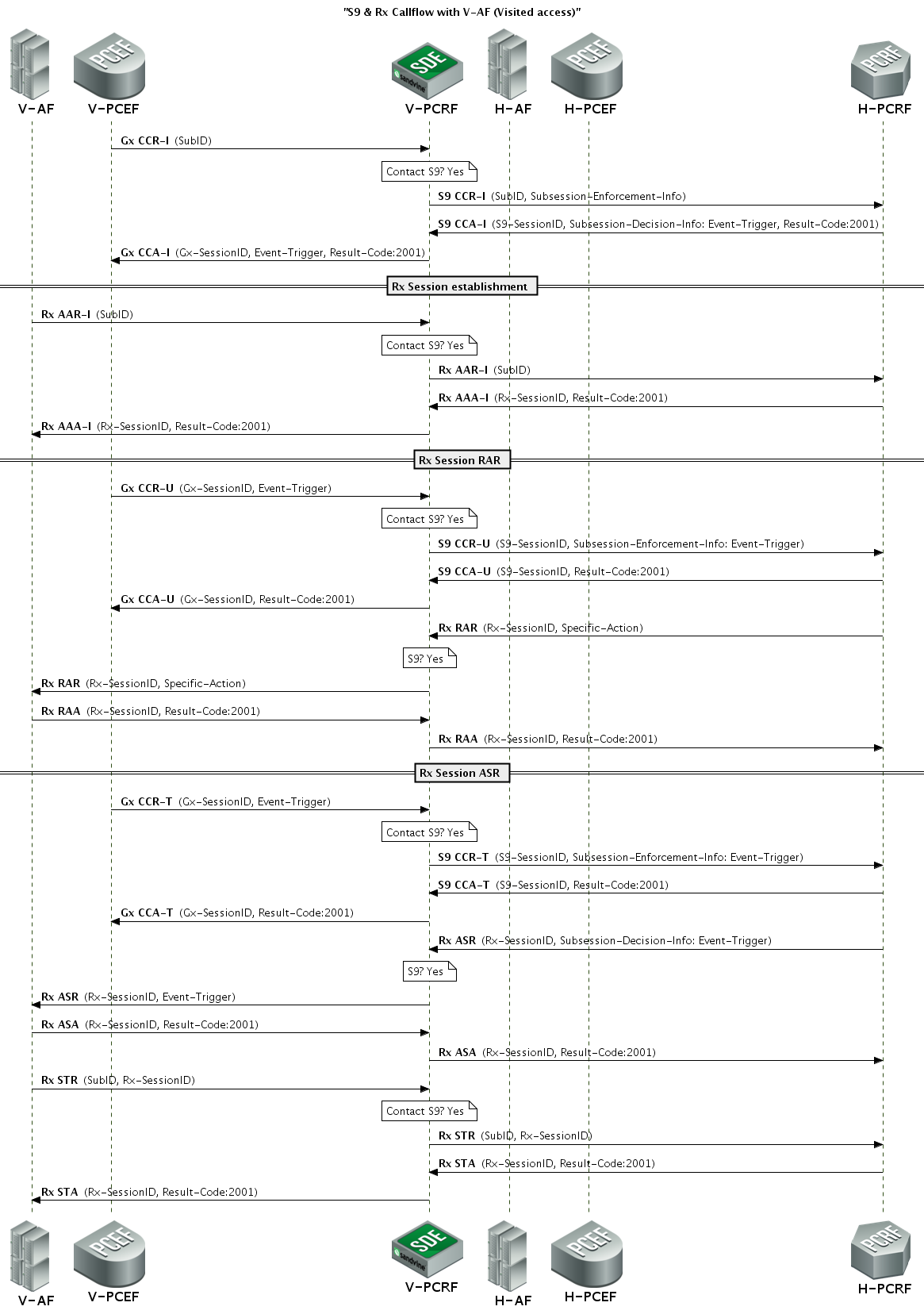


Figure : S9 & Rx Session with V-AF (Visited access)

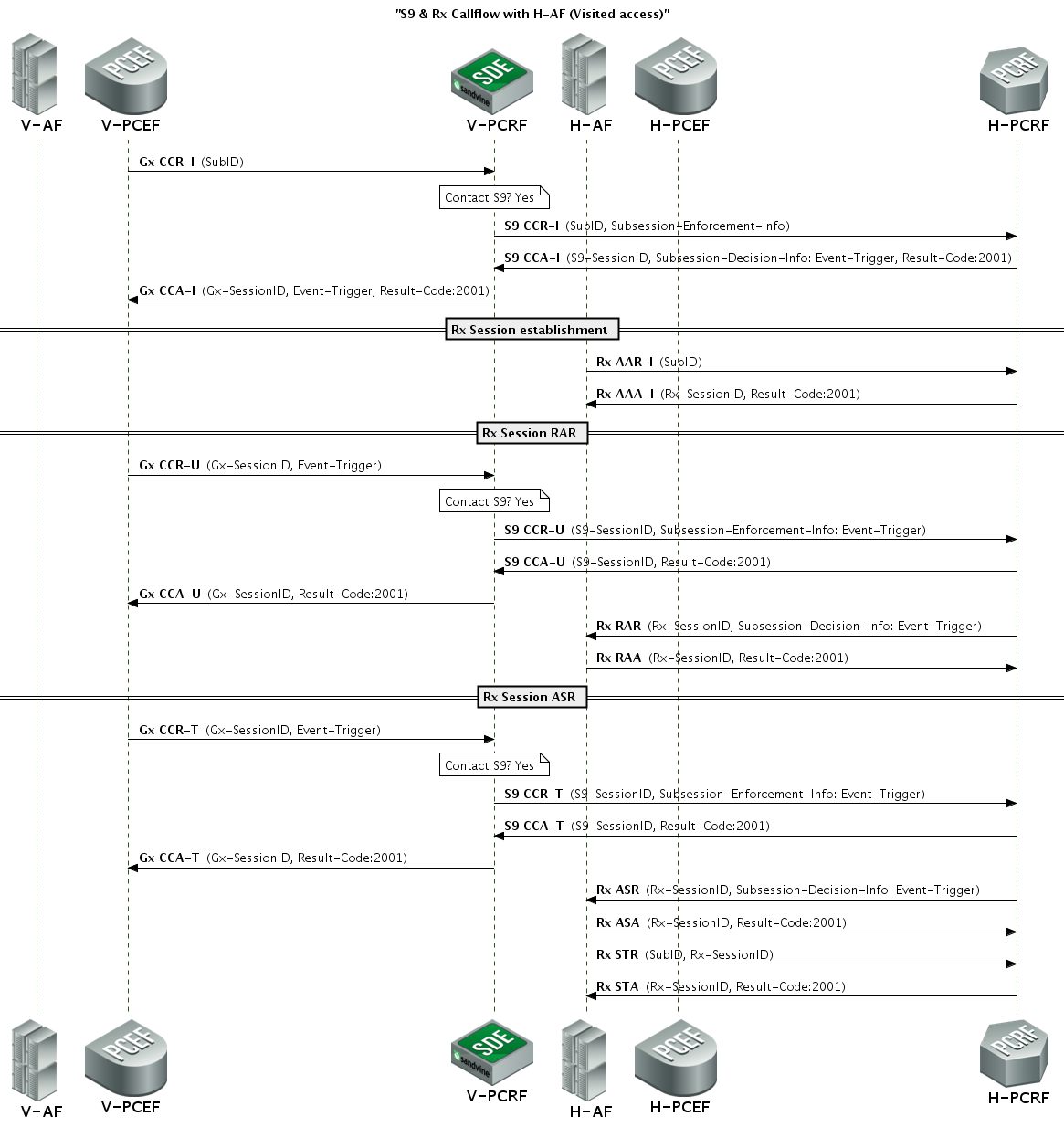


Figure : S9 & Rx Session with H-AF (Visited access)

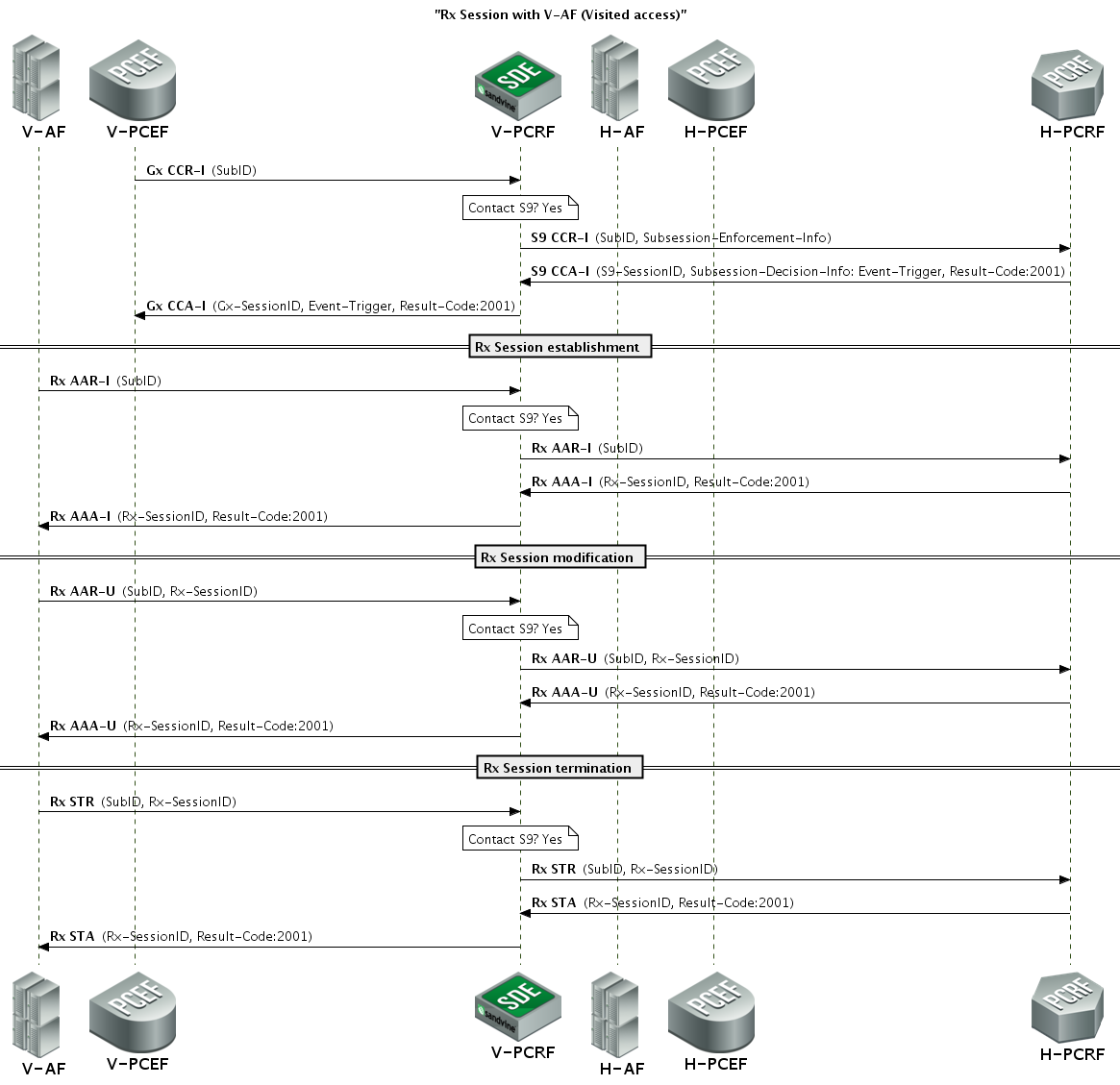


Figure : Rx Session with V-AF (Visited access)

## System Requirements

SDE 8.10 S9 SRS - 6

## Backward Compatibility

The following existing subsystems shall be impacted by addition of S9 in SDE 8.10:

* Gx Server
  + Will be updated to check if the subscriber is a roaming subscriber or not for every incoming Gx CCR-I request
  + Gx Sanity code will be refactored to use in S9 server.
  + Gx work flows will be updated to handle requests for roaming subscribers.
* Business Rules
  + Will be updated to install rules for roaming subscriber over S9 interface.
* Rx Server
  + Updated to check if the request is for a roaming subscriber or not.
  + SDE acting as S9 client (V-PCRF), if Rx request is received for a roaming subscriber, it will be forwarded to H-PCRF.
  + SDE acting as S9 server (H-PCRF), if Rx request is received, it will be rejected only if there is no Gx and as well as S9 session.

The functionalities, SandScript, CLI, Configuration, Reports & Audit logs of the existing subsystems MUST be backward compatible and unaffected with addition of S9.

## Design Constraints

1. In case PCRF receives Gx CCR for the existing subscriber with changed origin-state-id, on receiving this message PCRF deletes the existing session. The same behavior will continue for S9 CCR message.
2. As SDE is single thread application, on receiving Gx CCR-I one after the other for the same subscriber, there will be single instance of SubSessionEnforcmentInfo in S9 CCR-I message. On receiving Gx CCR-U the multiple instances of SubSessionEnforcementInfo AVPs is possible.

# High Level Design

## System Interfaces

### V-PCRF

#### System interaction

#### GxServer Subsystem

The GxServer needs to interact with the S9-Client Subsystem to relay enforcement request (i.e. Gx CCR) and enforcement result (i.e. Gx RAA). The idea here is that the diameter avps received on the Gx interface (from the PGW), if needed (subscriber is roaming and authorized by H-PLMN) needs to be relayed as S9 message over the S9 interface.

Based on the avps that are current parsed/encoded by the GxServer, here is the class definition for the object exchanged between GxServer and S9Client.

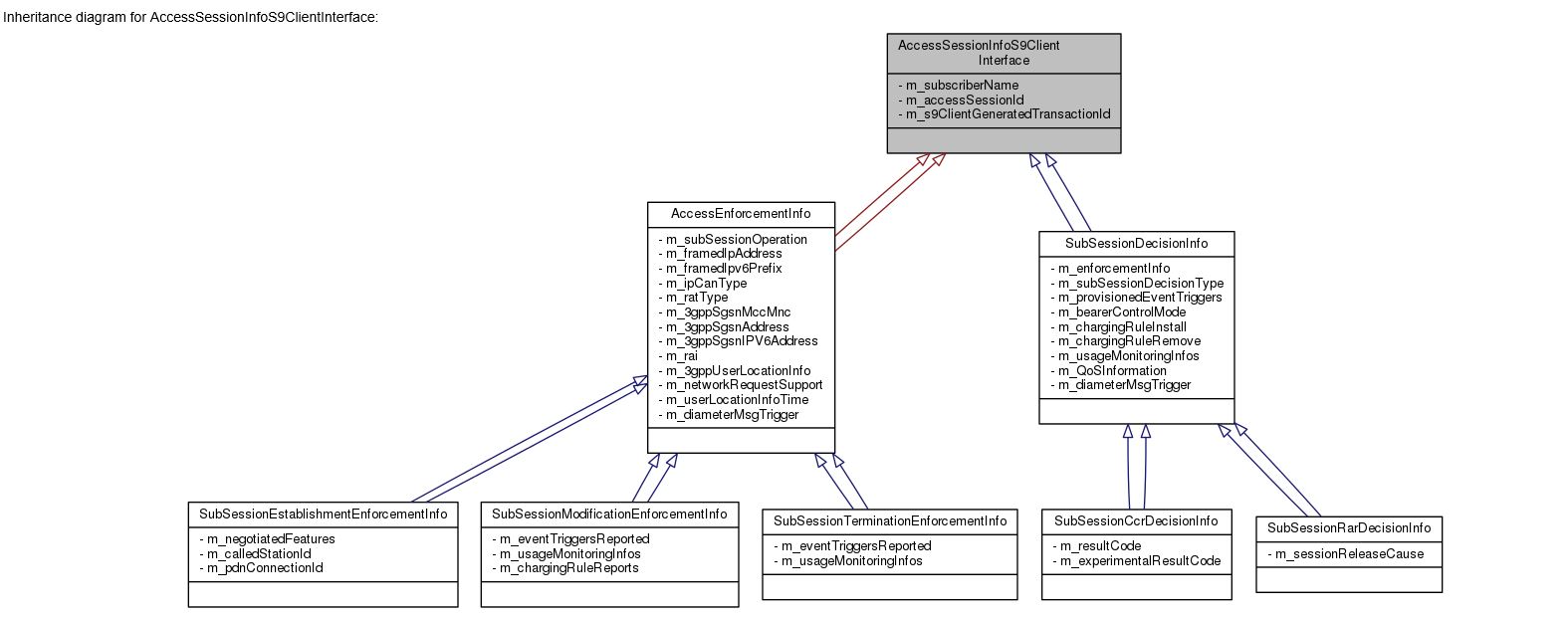


Figure : AcessSessionInfoS9ClientInterface

The class diagram is generated using Doxygen.

AccessEnforcementInfo instances are a set of TinyDiameter::AvpStringType, i.e. Raw bytes of avp values (includes only the avp value and not the avp header). This info shall be constructed by the GxServer on receiving CCR/RAA from the PGW.

Similarly, SubSessionDecisionInfo instances are a set of Raw bytes of values received over the S9 interface. More importantly DecisionInfo shall contain raw bytes for the final decision (decision after QoS Validation). For example m\_chargingRuleInstall shall contain vector of TinyDiameter::AvpStringType for rules which has passed QoS Validation.

###### Modifications to existing Gx Workflows

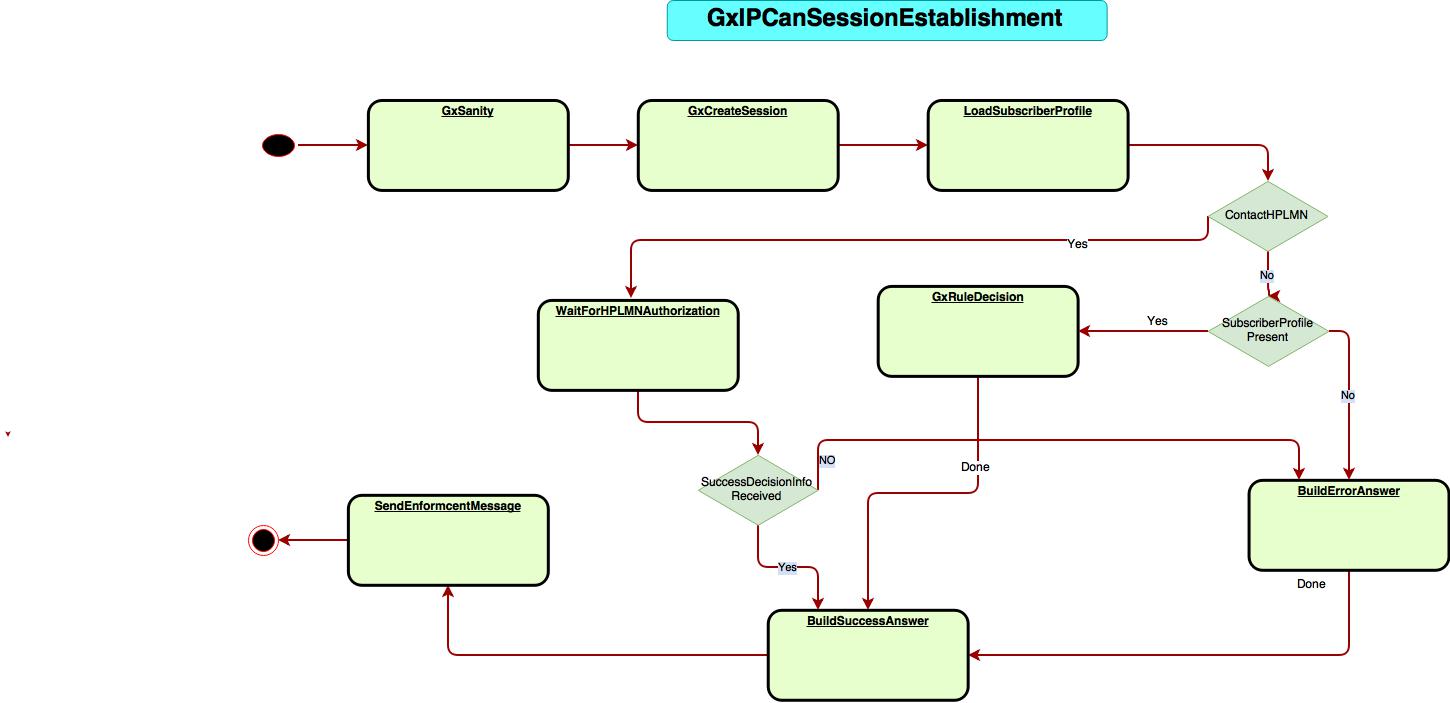


Figure : GxIPCanSessionEstablishment

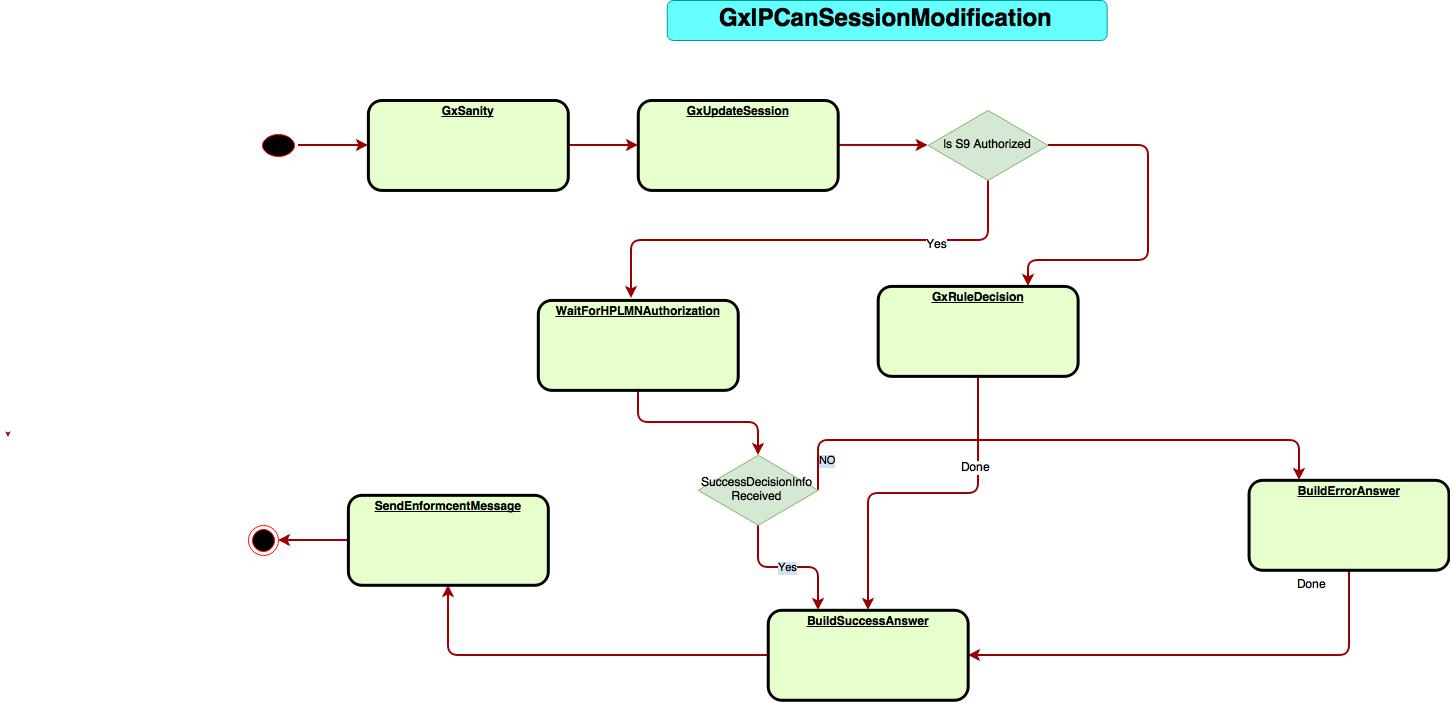


Figure : GxIPCanSessionModification

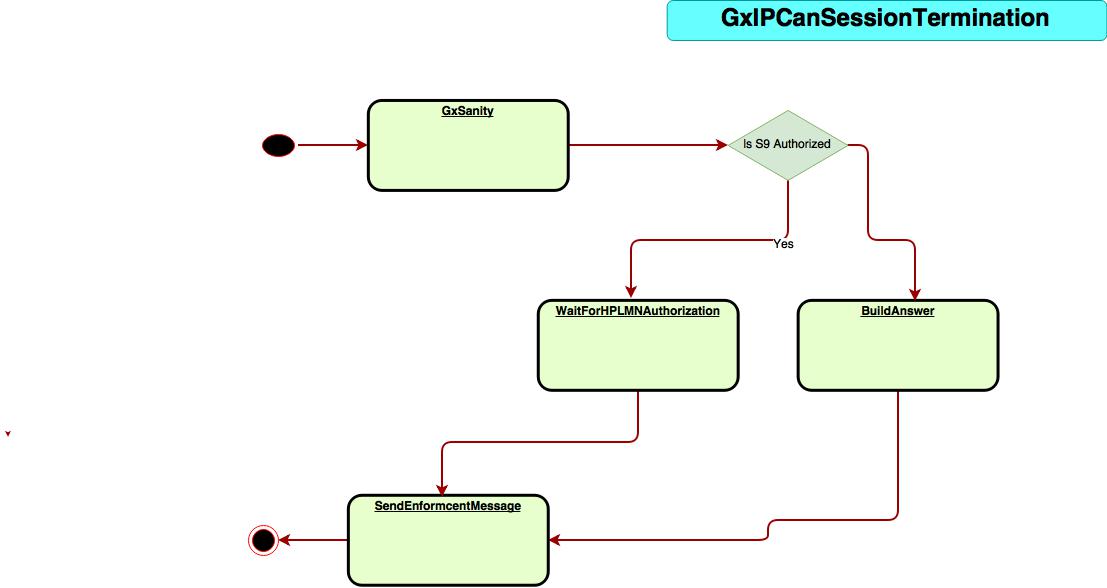


Figure : GxIPCanSessionTermination

###### Multi-Gx support

For a subscriber with a PDN Connection ID within the PDN-Connection-ID AVP, the user identity within the Subscription-Id AVP, the PDN ID within the Called-Station-Id AVP a unique Gx session exists between PGW and PCRF. In the roaming scenario, each Gx session (PGW – PCRF) for a subscriber results in a S9 session/subsession.

In case of multi-Gx support multiple Gx sessions (PGW - PCRF and PTS - PCRF) exists for a subscriber - with same PDN Connection ID within the PDN-Connection-ID AVP, the user identity within the Subscription-Id AVP, the PDN ID within the Called-Station-Id AVP. In the roaming scenario, for multiple Gx sessions for a subscriber a single S9 session/subsession MUST be created. This achieved by using the existing session attribute EnforcementGroup. The policy author shall create S9 session for one of the EnforcementGroup in Gx session.

If multi-Gx is supported the following Session attributes and configuration for EnforcementGroup is applicable:

|  |
| --- |
| **/usr/local/sandvine/etc/session/dictionary/GxServer.default.xml**  <Attribute>  <Name>EnforcementGroup</Name>  <Type>string</Type>  <Writable>false</Writable>  </Attribute>  **/usr/local/sandvine/etc/session/enforcementgroup/EnforcementGroup.xml**  <Host-Type>  <Name>ShapingGroup</Name>  <Host>pts.sandvine.com</Host>  <Realm>sandvine.com</Realm>  </Host-Type>  <Host-Type>  <Name>MonitoringGroup</Name>  <Host>pgw.sandvine.com</Host>  </Host-Type>  Note - The <EnforcementGroup> tag is applicable for Rules |

Figure : Session attribute & Config for EnforcementGroup

In the above configuration:

* if the Origin-Host is pts.sandvine.com and the Origin-Realm is sandvine.com then the EnforcementGroup is set to ShapingGroup
* if the Origin-Host is pgw.sandvine.com then the EnforcementGroup is set to MonitoringGroup

During Gx IP-CAN Session establishment (GxIPCANSessionEstablishmentWorkflow), based on EnforcementGroup, the policy author shall trigger S9 session establishment.

|  |
| --- |
| PolicyGroup expr (Workflow.GxIPCANSessionEstablishment.LoadSubscriberProfile.Pre)  {  if (**Session[Workflow.IPCANSessionEstablishment.LoadSubscriberProfile.Pre.Gx.'CC-Request'.'Session-Id'].AttributeView.Gx.EnforcementGroup = "MonitoringGroup"**) then \  {  if expr(Workflow.GxIPCANSessionEstablishment.LoadSubscriberProfile.Pre.Subscriber.Name = <a roaming subscriber>) then \  {  set Workflow.GxIPCANSessionEstablishment.LoadSubscriberProfile.Pre.ContactHplm = true  }  }  } |

Figure : Sample policy.conf for multi-Gx support

###### Overlapping IP (Missed logout) support

On receiving Gx-CCR-I with an overlapping IP-Address & Site-Qualifier (If a Gx session already exists with same IP-Address & Site-Qualifier for different subscriber), the GxServer shall

* Initiate termination of existing Gx session by sending Gx-RAR-T to PCEF
* On successful termination of Gx session (on receiving Gx-RAA-T, followed by exchange of Gx-CCR-T & Gx-CCA-T), if corresponding S9 subsession exists (roaming subscriber), terminate the existing S9 subsession (exchange S9-CCR-U/T and S9-CCA-U/T messages with H-PCRF)
* Initiate new Gx session and thus S9 subsession with the same IP-Address

**Note** – If Gx session exists for the same subscriber and origin-state-id is changed, overwrite the existing Gx session with the new Gx session with the latest Gx-CCR-I message.

Note – If Gx session exists for the same subscriber (belongs to a different PCEF), send Gx-RAR-T for the first session and once the first session is terminated gracefully, create the new Gx session.

**Note** – This should be handled in GxIPCANSessionEstablishmentWorkflow at GxServer. CQ reqst00544734 shall address the changes in GxServer subsystem.

###### Deferred IP allocation support

In case of deferred IP allocation (UE IP allocation deferred during default bearer establishment) on receiving Gx-CCR-I without framed IP Address, the Gx session & the corresponding S9 session/ subsession (for roaming subscriber) shall be created.

On UE IP allocation at V-PCEF, the V-PCEF notifies the V-PCRF in the next Gx-CCR-U with UE\_IP\_ADDRESS\_ALLOCATE event. The existing Gx session (and S9 session/ subsession) shall be bound to the framed IP Address.

**Note** – In this case on receiving Gx-CCR-U, while updating framed IP Address in Gx session (& S9 session/ subsession), check for Overlapping IP and handle the scenario as explained in Overlapping IP (Missed logout) support.

**Note** – Similar on UE\_IP\_ADDRESS\_RELEASE event notification in Gx-CCR-U from V-PCEF, the framed IP Address is unbound from Gx session (& S9 subsession).

#### S9ClientSubsystem

The S9ClientSubsystem (using the WorkflowEngine) listens/polls on the following interfaces

1. Internal-Triggers from the PCRF (Trigger from Gx Subsystem) :
   1. to create the following workflows
      1. S9ClientSessionEstablishment
      2. S9ClientSessionModification
      3. S9ClientSessionTermination
   2. resume the following workflows (On reciving answer from H-PCRF)
      1. S9ClientReAuthorization

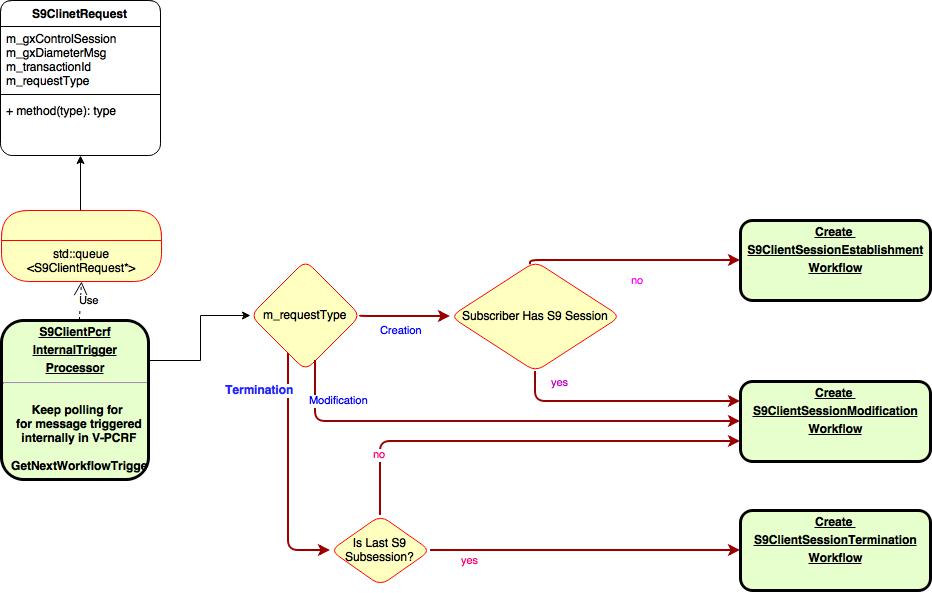


Figure : S9ClientPcrfInternalTriggerProcessor creating S9Client Workflows

1. External Triggers, receive the S9 Diameter messages from remote diameter peer
   1. To create the following workflow
      1. S9ClientReAuthorization
   2. Resume the following workflows
      1. S9CCRProcedure

##### S9Client workflow to handle S9 CCR procedure

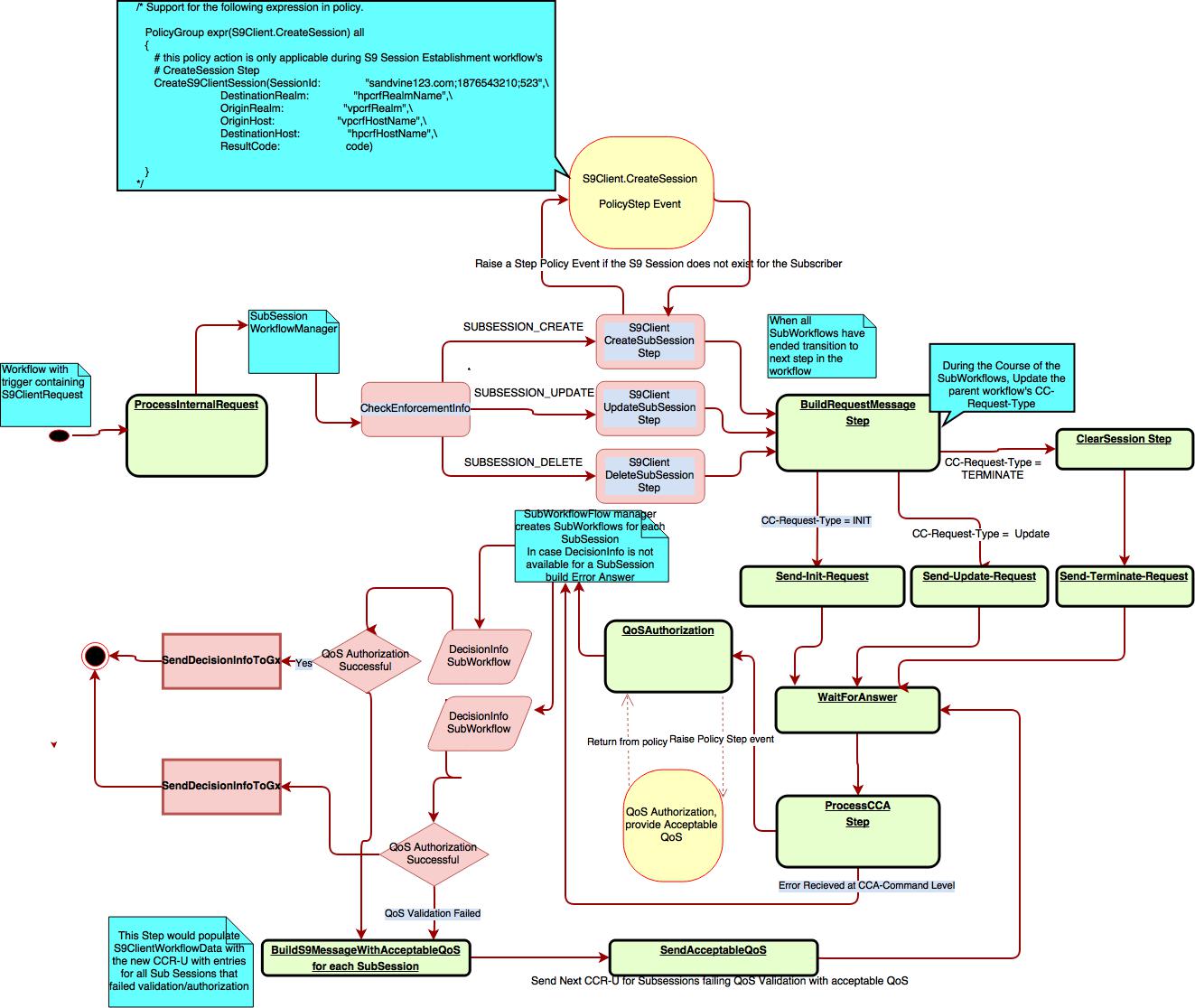


Figure : S9Client Credit Control workflow

##### S9Client workflow to handle RAR procedure

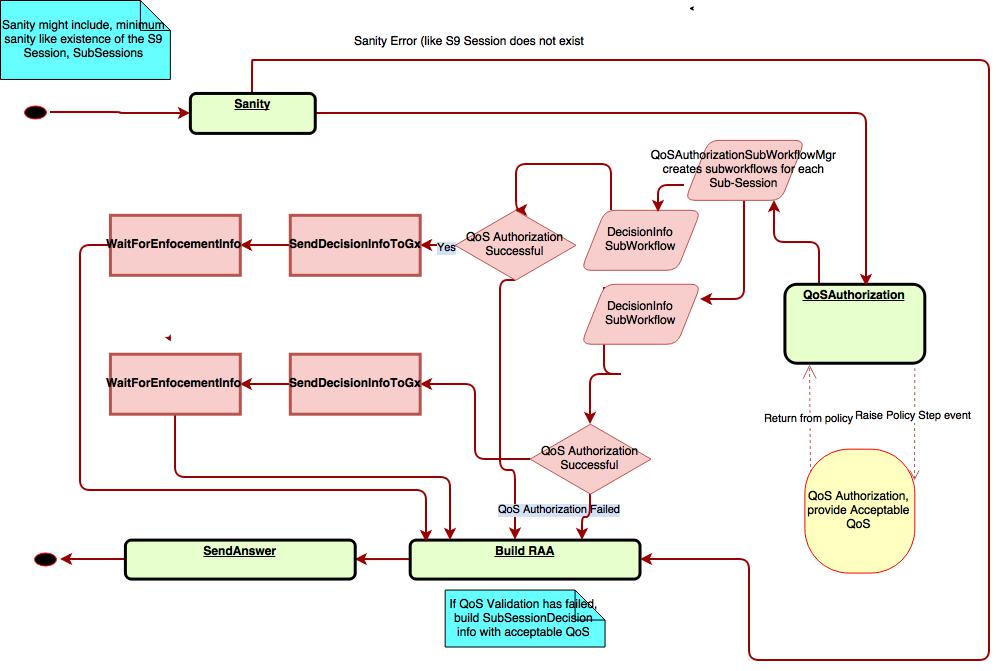


Figure : S9Client Reauthorization workflow

##### QoS Authorization/Validation at S9Client

QoS authorization needs to be done for

1. BEARER\_LEVEL – This is for the QoS-Information avp received at the Sub-Session level
2. APN\_LEVEL – This is for the APN aggregate max bit rate received for an access session (Sub-Session -> GxSession)
3. DEFAULT\_EPS\_BEARER – This is the Default-EPS-Bearer-QoS received over S9 interface
4. PCC\_RULE\_LEVEL – This is for QoS-Information received for a PCC Rule

The design proposes we have a single policy action AcceptableQoS with the definition as follows

AcceptableQoS (‘Sub-Session-Id’ : <integer>,

Level : <enum definition each of the levels>

'Max-Requested-Bandwidth-UL' : < integer>

'Max-Requested-Bandwidth-DL' : <integer>

'Max-Guaranteed-Bandwidth-UL': <integer>

'Max-Guaranteed-Bandwidth-DL' : <integer>

'QCI' : <integer>)

Now some of the args are only applicable to specific LEVELs. But I thought it would be simpler, and less effort to implement these as a single policy action.

Note: We could choose to prefix subsystem name before Action name. (i.e. S9.AcceptableQoS). But we do have other existing policy actions without subsystem name prefix.

Accordingly here are the policy actions to support QoS validation at the above defined levels

|  |  |
| --- | --- |
| LEVEL | API |
| BEARER\_LEVEL | AcceptableQoS(  'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : BEARER\_LEVEL /\* Bearer Id is implicitly available in subsystem\*/ ,\                                  'Max-Requested-Bandwidth-UL' : maxUplinkRatePermitted, \                                  'Max-Requested-Bandwidth-DL' : maxDownlinkRatePermitted) |
| APN\_LEVEL | AcceptableQoS( 'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : APN\_LEVEL, \                                 'APN-Aggregate-Max-Bitrate-UL' : apnMaxUplinkRatePermitted, \                                 'APN-Aggregate-Max-Bitrate-DL' : apnMaxDownlinkRatePermitted) |
| DEFAULT\_EPS\_BEARER | AcceptableQoS( 'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : DEFAULT\_EPS\_BEARER, \                                  'QoS-Class-Identifier' = ALLOWED\_DEFAULT\_EPS\_QCI\_PER\_ROAMING\_AGREEMENT) |
| PCC\_RULE\_LEVEL | AddRuleFailingQoSAuthorization('Sub-Session-Id' : subSession.'Subsession-Id', \                                                      RuleName : pccRuleToInstall.'Charging-Rule-Name')  AcceptableQoS( 'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : PCC\_RULE\_LEVEL, \                                  'Max-Requested-Bandwidth-UL' : pccRuleAvailableQoS.MaxBitRateUplink, \                                  'Max-Requested-Bandwidth-DL' : pccRuleAvailableQoS.MaxBitRateDownlink, \                                  'Guaranteed-Bitrate-UL' : pccRuleAvailableQoS.GuaranteedBitRateUplink ,\                                  'Guaranteed-Bitrate-DL' : pccRuleAvailableQoS.GuaranteedBitRateDownlink) |

Internal Data structure to hold the acceptable QoS provided as a policy action:

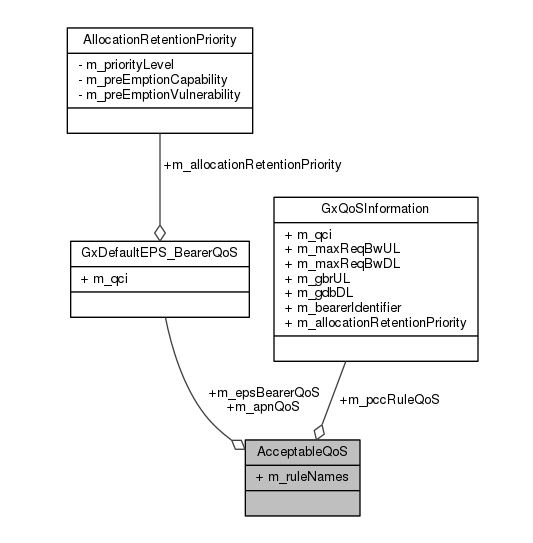


Figure : Acceptable QoS

Here is a sample policy on QoS authorization during PCC rule request procedure (CCR)

|  |
| --- |
| PolicyGroup expr(S9Client.QoSAuthorization.S9.'CC-Answer')  {          integer "maxDownlinkRatePermitted" = "expression deriving allowed rate as per roaming agreement"          integer "maxUplinkRatePermitted" = "expression deriving allowed rate as per roaming agreement"          integer "apnMaxUplinkRatePermitted" = "expression for max rate as per roaming agreement"          integer "apnMaxDownlinkRatePermitted" = "expression for max rate as per roaming agreement"          foreach subSession in S9Client.QoSAuthorization.S9.'CC-Answer'.'Sub-Session-Decision-Info'          {            foreach cmdLevelQoSInfo in subSession.'QoS-Information'            {              PolicyGroup expr (any \                              (cmdLevelQoSInfo.'Max-Requested-Bandwidth-DL' >  maxDownlinkRatePermitted) \                              ,(cmdLevelQoSInfo.'Max-Requested-Bandwidth-UL' >  maxUplinkRatePermitted) \                              ) all              {                  log values ("QoS authorization failed for Session: ", S9Client.QoSAuthorization.S9.'CC-Answer'.'Session-Id')                  log values ("CC Request number: ", S9Client.QoSAuthorization.S9.'CC-Answer'.'CC-Request-Number')                  log values ("Providing acceptable QoS")                  AcceptableQoS(  'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : BEARER\_LEVEL /\* Bearer Id is implicitly available in subsystem\*/ ,\                                  'Max-Requested-Bandwidth-UL' : maxUplinkRatePermitted, \                                  'Max-Requested-Bandwidth-DL' : maxDownlinkRatePermitted)              }              PolicyGroup expr (any \                               (cmdLevelQoSInfo.'APN-Aggregate-Max-Bitrate-UL' > apnMaxUplinkRatePermitted), \                               (cmdLevelQoSInfo.'APN-Aggregate-Max-Bitrate-DL'> apnMaxDownlinkRatePermitted) \                               )all              {                  log values ("QoS authorization failed for max apn rate")                  AcceptableQoS( 'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : APN\_LEVEL, \                                 'APN-Aggregate-Max-Bitrate-UL' : apnMaxUplinkRatePermitted, \                                 'APN-Aggregate-Max-Bitrate-DL' : apnMaxDownlinkRatePermitted)              }            }            # end iterating through QoS-Information at the sub-session level            PolicyGroup expr (subSession.'Default-EPS-Bearer-QoS'.'QoS-Class-Identifier' > \                  ALLOWED\_DEFAULT\_EPS\_QCI\_PER\_ROAMING\_AGREEMENT)            {                  AcceptableQoS( 'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : DEFAULT\_EPS\_BEARER, \                                  'QoS-Class-Identifier' = ALLOWED\_DEFAULT\_EPS\_QCI\_PER\_ROAMING\_AGREEMENT)            }            # we would expect the policy to maintain the QoS Bit Rates already allocated for a Sub-Session/Session/Subscriber            # the policy would maintain the quota of QoS already allocated, and ensure every new rule debits from            # this Quota of QoS allowed.            Table.Operator\_X\_Subscriber\_RoamingQuota:cursor pccRuleAvailableQoS = \              Table.Operator\_X\_Subscriber\_RoamingQuota:cursor[S9Client.QoSAuthorization.Subscriber.Name]            bool "ruleFailedQoSAuthorization" = false            foreach pccRuleToInstall in subSession.'Charging-Rule-Install'            {              PolicyGroup              {                  PolicyGroup(any (pccRuleToInstall.'QoS-Information'.'Guaranteed-Bitrate-UL' > pccRuleAvailableQoS.MaxBitRateUplink), \                          (pccRuleToInstall.'QoS-Information'.'Guaranteed-Bitrate-DL' > pccRuleAvailableQoS.MaxBitRateDownlink), \                          (pccRuleToInstall.'QoS-Information'.'QoS-Class-Identifier' > ALLOWED\_QCI\_PER\_SDF) \                          )                    {                          set ruleFailedQoSAuthorization = true and \                          AddRuleFailingQoSAuthorization('Sub-Session-Id' : subSession.'Subsession-Id', \                                                      RuleName : pccRuleToInstall.'Charging-Rule-Name')                    }                  PolicyGroup all                    {                     #else part debit QoS already allocated                     set pccRuleAvailableQoS.MaxBitRateUplink = pccRuleToInstall.'QoS-Information'.'Max-Requested-Bandwidth-UL'                     set pccRuleAvailableQoS.MaxBitRateDownlink = pccRuleToInstall.'QoS-Information'.'Max-Requested-Bandwidth-DL'                    }              }             }             #end iterating through pcc rules             # provide acceptable QoS from policy to subsystem             if expr (ruleFailedQoSAuthorization = true) then \                  AcceptableQoS( 'Sub-Session-Id' : subSession.'Subsession-Id', \                                  Level : PCC\_RULE\_LEVEL, \                                  'Max-Requested-Bandwidth-UL' : pccRuleAvailableQoS.MaxBitRateUplink, \                                  'Max-Requested-Bandwidth-DL' : pccRuleAvailableQoS.MaxBitRateDownlink, \                                  'Guaranteed-Bitrate-UL' : pccRuleAvailableQoS.GuaranteedBitRateUplink ,\                                  'Guaranteed-Bitrate-DL' : pccRuleAvailableQoS.GuaranteedBitRateDownlink)          }          #end iterating through sub-sessions  } |

##### Gx Diameter Message to S9 Diameter Message

1. A utility class AppMsgCopy is introduced for translating a Diameter message to another Diameter message, where only specific AVPS are copied. The given Diameter message maybe of different applications. While copying from source message the AVPs are NOT decoded and copied in raw format (byte-stream). Thus the translation shall be optimal. Also in the source message specific AVPs can be avoided while coping. The pseudo code for Diameter message class is as follows:

|  |
| --- |
| class AppMsgCopy  {  static const size\_t DIAMETER\_HEADER\_OFFSET = 20;  static bool **CopyDiameterAppSpecificAVP**(TinyDiameter::DiameterWriter &packet,  DiameterMsg\* from,  MAP dicardAVPs)  {  // All processing on raw data stream of received message  OctetString msgData = from.Data();  osiUint32 msgSize = from.Size();  if (DIAMETER\_HEADER\_OFFSET <= msgSize)  {  // No AVPs to extract!  return false;  }  // Skip Diameter header & Goto AVPs  for (size\_t index = DIAMETER\_HEADER\_OFFSET;  index <= msgSize;  index += avpSize)  {  rawAvp = msgData[index];  osiUint32 avpCode = msgData[index + 0] << 24 |  msgData[index + 1] << 16 |  msgData[index + 2] << 8 |  msgData[index + 3];    /\* AVP-Size = AVP Header + AVP Value \*/  osiUint32 avpSize = 0x00FFFFFF &  msgData[index + 4] << 16 |  msgData[index + 5] << 8 |  msgData[index + 6];  if (avpCode NOT in MAP discardAVPs)  {  packet.AppendRawAvp(&rawAvp, avpSize );  }  }  }  } |

Figure : AppMsgCopy class to copy specific AVPs from Diameter Message

In order to facilitate raw AVP writing new method AppendRawAvp is added in existing DiameterHeaderWriter class. This method shall append the specific AVP block at the last in the destination message. The pseudo code for AppendRawAvp method is as follows:

|  |
| --- |
| // To append raw AVP stream at the end of m\_data.  // rawAvp – Raw AVP data stream  // avpSize – rawAvp length    bool DiameterHeaderWriter::AppendRawAvp(char \*rawAvp, avpSize)  {  m\_lastBytes = 0;  if (!m\_nextAvp) return false; // nextAvp NOT valid! where to write?  if (m\_nextAvp + avpSize > m\_end) return false; // no space to write!  memcpy(m\_nextAvp, rawAvp, avpSize);  m\_nextAvp += avpSize; // point to next avp position  m\_lastBytes = avpSize; // update last bytes wrote  m\_len = AvpAlignLen(m\_len) + m\_lastBytes; // update msg length  return m\_nextAvp != 0;  } |

Figure : AppendRawAvp method added in DiameterHeaderWriter class

The following pseudo code is an example, where the utility class AppMsgCopy is used while building S9-CCR-I message from received Gx-CCR-I message during S9 session establishment. PopulateDiscardAVPs() shall be both application & message specific.

|  |
| --- |
| BuildS9CCR(...)  {  ...  // Received Gx Diameter Msg:  DiameterMsg\* gxMsg;  DiameterMsg\* ccr = stack->NewMsg();  TinyDiameter::DiameterHeaderWriter header(ccr->Begin());  // Build CCR Message header  DiameterParser::MakeHeader(header, ...);  // AVPs Writer  TinyDiameter::DiameterWriter packet(&header, ccr->End());  /\* Populate following S9 specific Diameter AVPs and discard them in Gx message  < Session-Id >  { Auth-Application-Id }  { Origin-Host }  { Origin-Realm }  { Destination-Realm }  { CC-Request-Type }  { CC-Request-Number }  [ Destination-Host ]  [ Origin-State-Id ]  \*/  // List of AVPs to be discarded in received Gx Msg  MAP discardAVPs = PopulateDiscardAVPs();  // Copy Gx CCR message content  AppMsgCopy::CopyDiameterAppSpecificAVP(packet, gxMsg, discardAVPs);  ...  } |

Figure : Example - Building S9 CCR-I

1. From workflow point of view:
2. S9SessionEstablishment/Modification/Termination/ReAuthAnswer:

There is no Sanity Step in these workflows.

1. But in S9ClientSessionEstablisment/Modification workflow's ProcessCCA step AVPs will be parsed before QoS Authorization. After QoS Authorization, SubsessionDecisionInfo interface message will be created.
2. Function prototype for parsing AVPs received in CCA-I/U and RAR will as:

|  |
| --- |
| Void S9ClientSanityStep::ParseChargingRuleInstall( TinyDiameter::DiameterInputParser& parser,  const TinyDiameter::AvpKey& chargingRuleInstallKey, uint32\_t& errorType, std::vector< TinyDiameter::AvpStringType> & chargingRuleInstallInstances)  {  static const TinyDiameter::AVPPath chargingRuleInstallPath(chargingRuleInstallKey);    /\*  1. In a loop call GetAvpValue() for multiple instances of ChargingRuleInstall.  2. For each instance push back each TinyDiameter::AvpGroupType into chargingRuleInstallInstances std::vector.  \*/  } |

#### RxServer Subsystem

V-PCRF host will be acting as both RxClient and RxServer.

##### RxAFSessionEstablishment

When V-PCRF receives Rx AAR-I message, RxAFSessionEstablishment workflow is created. If a S9 session exists (i.e. S9 data session) for the associated IP-CAN, RxCreateSessionStep will transition to RxBuildMessageForHPCRF Step. Otherwise RxAFSessionEstablishmentStep will continue with existing workflow path. AAA Response from H-PCRF unblocks RxAFSessionEstablishment workflow and Response will be forwarded to V-AF.

**Step Description -**

RxSanityStep - This step parses and validates incoming AAR-I message. If any AVP is invalid, workflow will transition to RxBuildErrorAnswerStep else RxCreateSessionStep.

|  |
| --- |
| RxSanityStep::Run()  {   1. Parsing and validation of Rx AAR message AVPs. 2. It stores information received in AAR-I message to WorkflowData. 3. If Parsing/Validation of AAR-I message AVP is successful, Step return code is Success/Done and workflow will transition to RxCreateSession Step. 4. Otherwise workflow return code is Error and workflow will be updated with corresponding error (like DIAMETER\_INVALID\_AVP, DIAMETET\_INVALID\_AVP\_LENGTH) and workflow will transition to RxBuildErrorAnswer step.   } |

RxCreateSessionStep - This step creates Rx Session and checks whether request will be forwarded to H-PCRF based on S9 session existence for IP-CAN. If IP has Gx Session and S9 Subsession present(in case of V-PCRF), then forward this request to H-PCRF. Else if only S9 Subsession is present(in case of H-PCRF), transition to RuleDecisionStep.

|  |
| --- |
| RxCreateSessionStep::Run()  {   1. If Rx Session is already exists for received Session-Id in AAR-I message, update workflowData with DUPLICATE\_AF\_SESSION Error Result-Code and workflow will transition to RxBuildErrorAnswer Step. 2. If IP-CAN(Gx) session and S9 Subsession exists for IP received IP in AAR-I message(in case of V-PCRF), Create Rx Control Session, mark it as RoamingSession and workflow will transition to RxBuildMessageForS9 Step. 3. If S9 Session/Subsession exists and IP-CAN(Gx) session doesn’t exist (in case of H-PCRF), Create Rx Control Session. Step return code is Done/Success workflow will transition to RxRuleDecision Step. 4. Rx Control Session will be updated with information stored in workflowData(in RxSanityStep), so that this information will be persisted.   } |

RxBuildMessageForS9Step - This step builds new message from the original message received from V-AF by replacing Origin-Host, Origin-Realm, Destination-Host, Destination-Realm AVPs.

|  |
| --- |
| RxBuildMessageForS9Step::Run()  {   1. Form Diameter Message from received AAR-I message and modify Origin-Host, Origin-Realm AVP to V-PCRF information and Destination-Host and Destination-Realm of H-PCRF(fetched for S9 Session). 2. Original message will be deep copied before changing origin-host/origin-relam and Destination-host/destination-realm. 3. If Build Diameter Message fails, Step return code is Error and workflow will transition to RxBuildErrorAnswer Step. 4. If Diameter Message is built successfully, Step return code is Success/Done and   } |

RxForwardMessageStep - This step forwards message to destined peer.

|  |
| --- |
| RxForwardMessageStep::Run()  {   1. Send Diameter Message to Diameter Stack. 2. If Message is forwarded successfully, Step return code is Success/Done and workflow will transition to RxWaitForS9ResponseStep. 3. If Message sent failed, Step return code is Error and workflow will transition to RxBuildErrorAnswer Step.   } |

RxRuleDecisionStep - This step calls for feature evaluation and waits for Service Designer decision.

|  |
| --- |
| RxRuleDecisionStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxBuildSuccessAnswerStep - This steps builds Success AAA if previous steps completed successfully.

|  |
| --- |
| RxBuildSuccessAnswerStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxBuildErrorAnswerStep - This step builds Error AAA if previous steps completed with some error.

|  |
| --- |
| RxBuildErrorAnswerStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxWaitForS9ResponseStep - This step gets blocked when request is forwarded and gets unblocked when response is received from peer.

|  |
| --- |
| RxWaitForS9ResponseStep::Run()  {   1. Blocks workflow and waits for response(AAA) from H-PCRF. 2. When response(AAA) is received, unblocks workflow and workflow transitions to RxProcessResponseFromS9Step.   } |

RxPorcessResponseFromS9Step - This step modifies received response by replacing Origin-Host, Origin-Realm, Destination-Host, Destination-Realm AVPs.

|  |
| --- |
| RxProcessResponseFromS9Step::Run()  {   1. Form Diameter Answer message from received response(AAA) from H-PCRF and modify Origin-Host, Origin-Realm to V-PCRF information and Destination-Host and Destination-Realm to V-AF information. 2. When new message is formed, workflow will transition to RxSendMessageStep.   } |

RxSendMessageStep - This step sends message to destined peer.

|  |
| --- |
| RxSendMessageStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxClearSessionStep - This step clears Rx Session in case some step errors out and Session cleanup is required.

|  |
| --- |
| RxClearSessionStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

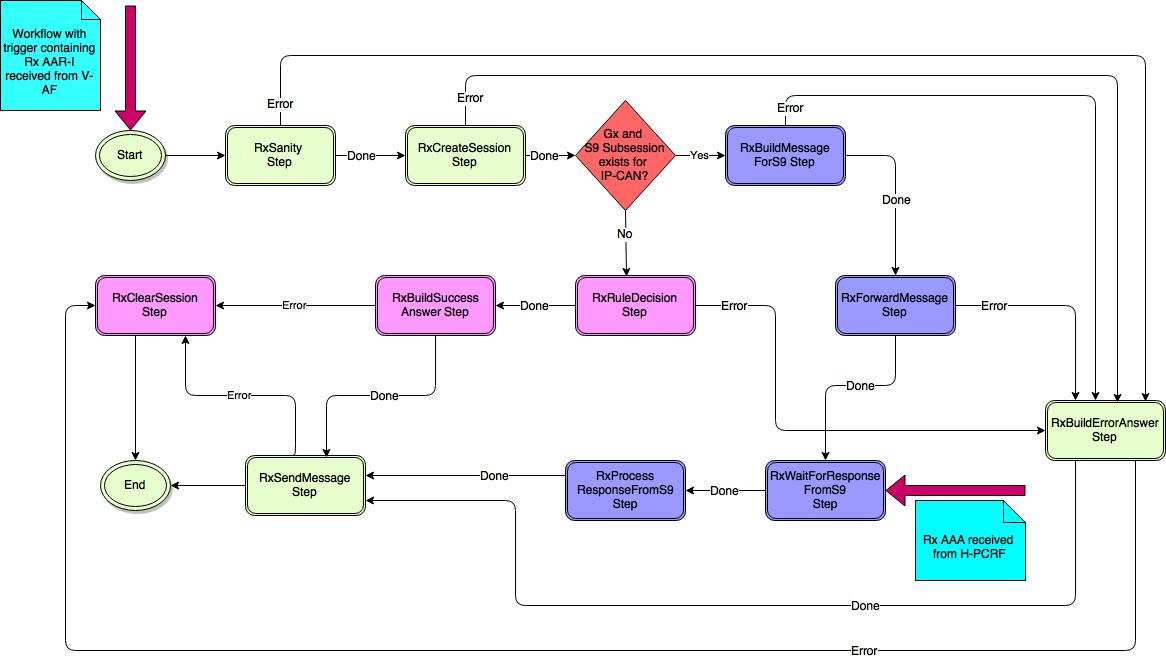


Figure : RxSessionEstablishment Workflow state machine

##### RxAFSessionModification

When V-PCRF receives Rx AAR-U message, RxAFSessionModification workflow is created. If a S9 session exists for the associated IP-CAN, RxUpdateSessionStep will transition to RxBuildMessageForHPCRF Step. Otherwise RxAFSessionModification will continue with existing workflow path. AAA Response from H-PCRF unblocks this workflow and Response will be forwarded to V-AF.

**Step Description -**

RxUpdateSessionStep - This step updates the Rx Session(created with AAR-I) information. This step transitions to RxBuildMessgaeForHPCRFStep if AAR-U message is to be forwarded to H-PCRF based on Rx Session created for visited subscriber. During AAR-I, V-PCRF comes to know that this is roamed in subscriber and rules should be provisioned by H-PCRF.

|  |
| --- |
| RxUpdateSessionStep::Run()  {   1. Rx Control Session will be updated with information stored in workflowData (in RxSanityStep), so that this information will be persisted. 2. If Rx Control Session is marked as RoamingSession (in case of V-PCRF), then workflow will transition to RxBuildMessageForS9 Step and step return code will be Success/Done . 3. Otherwise (In case of H-PCRF), workflow will transition to RxRuleDecisionStep.   } |

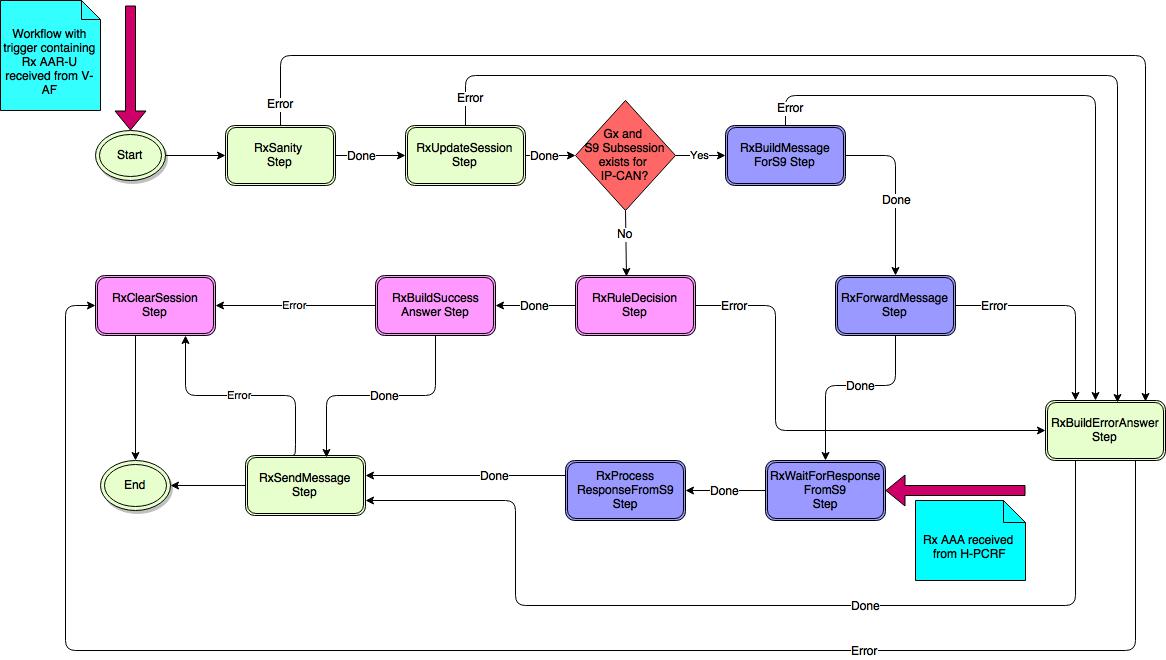


Figure : RxSessionModification Workflow state machine

##### RxAFSessionTermination

When V-PCRF receives Rx STR message, RxAFSessionTermination workflow is created. If a S9 session exists for the associated IP-CAN, RxClearSessionStep will transition to RxBuildMessageForHPCRFStep. Otherwise RxAFSessionTermination will continue with existing workflow path. STA response from H-PCRF unblocks this workflow and response will be forwarded to V-AF after replacing Origin-Host, Origin-Real, Destination-Host, Destination-Realm AVPs.

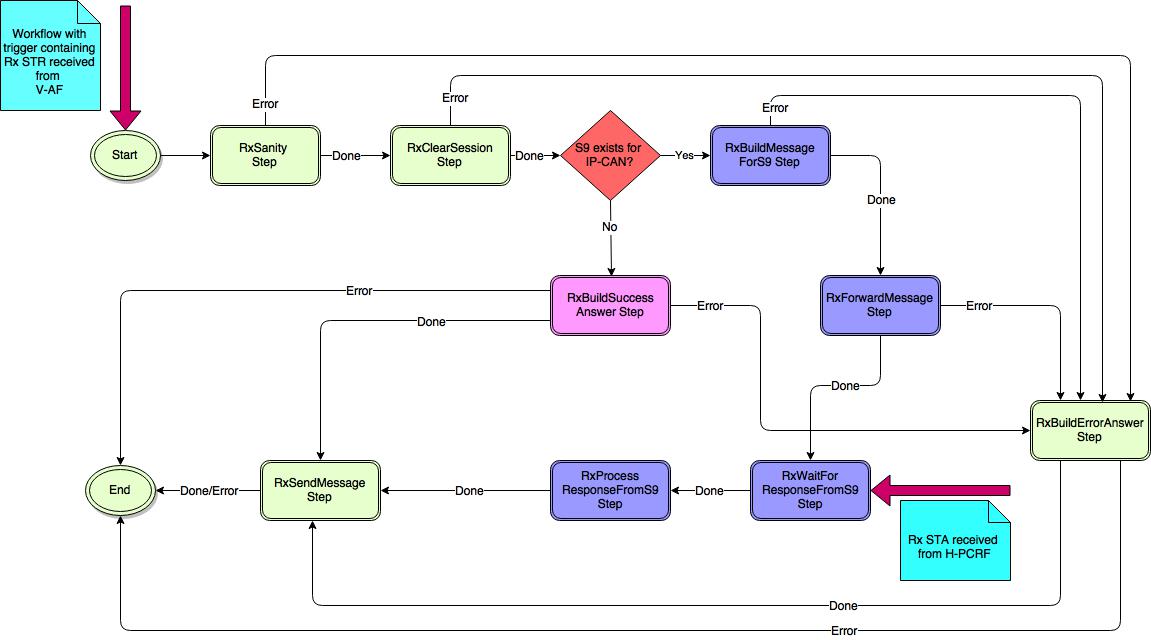


Figure : RxAFSessionTerminationWorkflow state machine

##### RxAFSessionReauthorization

When V-PCRF receives Rx RAR message from H-PCRF, RxAFSessionReauthorization workflow is created. Rx RAR message will be forwarded to V-AF in RxForwardMessage step and workflow will transition to RxWaitForS9ResponseStep. Later RxAFSessionReauthorization workflow will be blocked in RxWaitForS9ResponseStep. When Rx RAA message will be received from V-AF, RxAFSessionReauthorization workflow gets unblocked and process incoming Rx RAA message in RxProcessResponseFromS9Step and forward the same message to H-PCRF in RxSendMessageStep.

When H-PCRF receives Enforcement Request from Service Designer, RxAFSessionReauthorization workflow is created. H-PCRF send Rx RAR message to V-PCRF and gets blocked. Rx RAA message received from V-PCRF unblocks RxAFSessionReauthorization workflow.

RxProcessRAA step validates Rx RAA message and transitions to RxClearSession step if Rx RAA is received with Result-Code DIAMETER\_UNKNOWN\_SESSION\_ID and V-PCRF still has Rx session present.

Step return code “forward” will be used to differentiate between V-PCRF and H-PCRF workflow and workflow will transition to different step with this return code.

**Step Description-**

RxBuildRARRequestMessageStep - This step builds Rx RAR message based on trigger received from Service Designer or Rx RAR message received from H-PCRF(by replacing Origin-Host, Origin-Realm, Destination-Host, Destination-Realm AVPs).

|  |
| --- |
| RxBuildRARequestMessageStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxWaitForRAAStep - This step blocks RxAFSessionReauthorization workflow and gets unblocked when Rx RAA message is received from V-PCRF/V-AF.

|  |
| --- |
| RxWaitForRAAStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxProcessRAAStep - This step validates received Rx RAA message by validating AVPs and decides whether to forward message or not based on type of workflow trigger.

|  |
| --- |
| RxProcessRAAStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxClearSessionStep - This step will do Rx session cleanup as and when required. If Rx RAA is received with DIAMETER\_UNKNOWN\_SESSION\_ID Result-Code and session exists, workflow transitions to RxClearSession step.

|  |
| --- |
| RxClearSessionStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

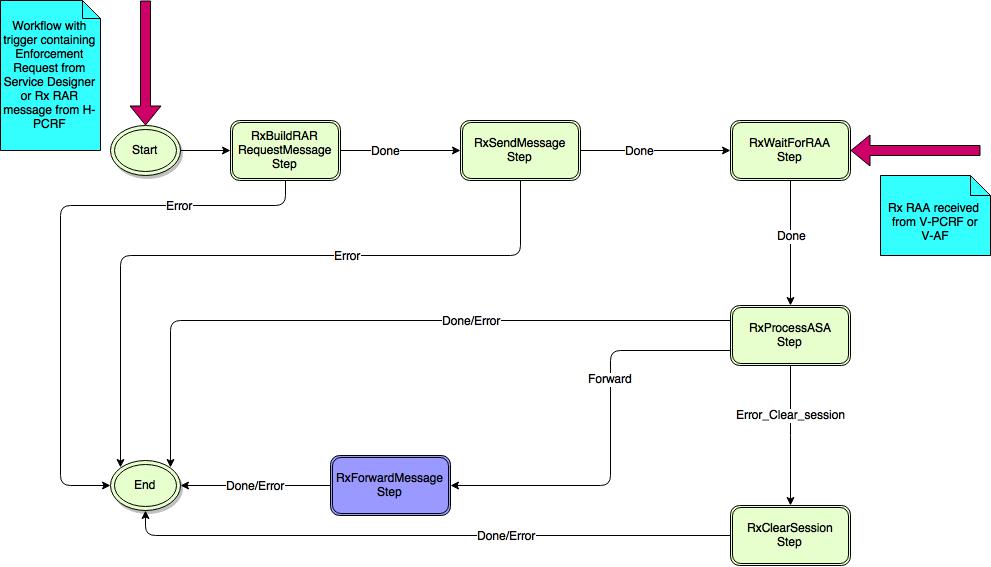


Figure : RxAFSessionReauthorization Workflow state machine at

V-PCRF/H-PCRF

##### RxAFSessionAbort

When V-PCRF receives Rx ASR message from H-PCRF, RxAFSessionAbort workflow is created. Rx ASR message will be forwarded to V-AF and block itself. When Rx ASA message will be received from V-AF, RxAFSessionAbort workflow gets unblocked and forward Rx ASA message to H-PCRF.

When H-PCRF receives Enforcement Request from Service Designer, RxAFSessionAbort workflow is created. H-PCRF send Rx ASR message to V-PCRF and gets blocked. Rx ASA message received from V-PCRF unblocks RxAFSessionAbort workflow.

Step return code “forward” will be used to differentiate between V-PCRF and H-PCRF workflow and workflow will transition to different step with this return code.

**Step Description -**

RxBuildASRequestMessageStep - This step builds Rx ASR message based on trigger received from Service Designer or Rx ASR message received from H-PCRF(by replacing Origin-Host, Origin-Realm, Destination-Host, Destination-Realm AVPs).

|  |
| --- |
| RxBuildASRequestMessageStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxWaitForASAStep - This step blocks RxAFSessionAbort workflow and gets unblocked when Rx ASA message is received from V-PCRF/V-AF.

|  |
| --- |
| RxWaitForASAStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

RxProcessASAStep - This step validates received Rx ASA message by validating AVPs and decides whether to forward message or not based on type of workflow trigger.

|  |
| --- |
| RxProcessASAStep::Run()  {  /\* No change is this step. This step will remain same as existing. \*/  } |

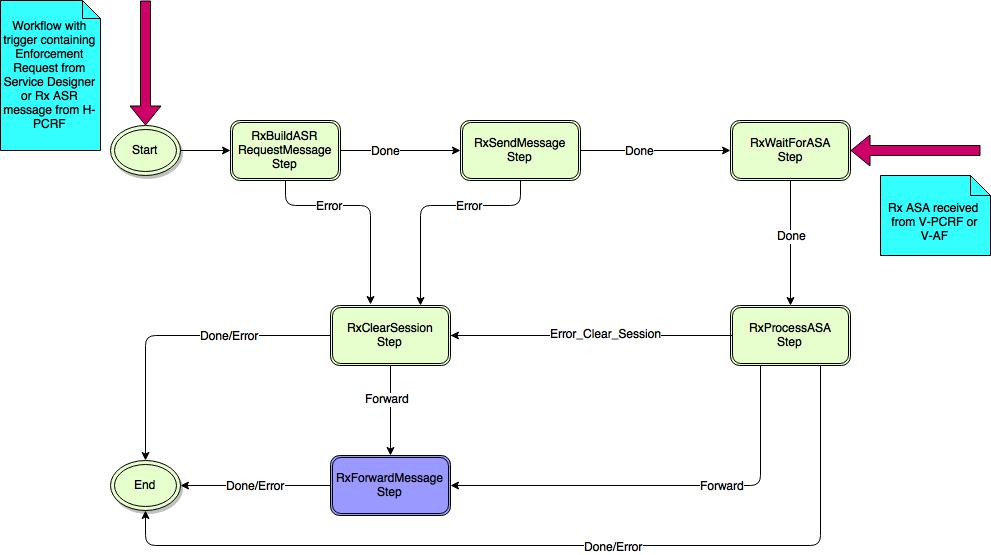


Figure : RxAFSessionAbort Workflow state machine at

V-PCRF/H-PCRF

#### Handling Origin-State-ID change of AF

There are two approaches at V-PCRF to handle change of Origin-State-ID for AF.

A new type of Rx Control Session(created at V-PCRF when AF is in V-PLMN) should have timer(let’s say 2 Hours) and at the expiry of this timer following approaches can be taken.

1. Let’s have a timer in Rx Control Session(2 hours) at V-PCRF and at the expiry of this timer, check whether Origin-State-Id for the AF has been changed. If yes, trigger ASR for this Rx Session and wait for ASA. If ASA is received with DIAMETER\_UNKNOW\_SESSION\_ID(in case session doesn’t exist at AF) and V-PCRF has originated ASR (based on trigger type for workflow. For relay behavior, diameter message received from H-PCRF is the trigger for workflow. Else some internal message is the trigger for workflow), trigger STR towards H-PCRF to cleanup Rx Session at H-PCRF. If V-PCRF is not the originator of this ASR message, forward ASA to H-PCRF.

Trigger type will be different in case of V-PCRF originated a message(workflow) or V-PCRF has relayed/forwarded a message. Like if V-PCRF has forwarded a message received from H-PCRF, diameter msg will be trigger for workflow. In other case some internal trigger will be trigger for workflow. We will be using different trigger type within workflow to identify it easily. A workflow can have different trigger type keyed by trigger name.

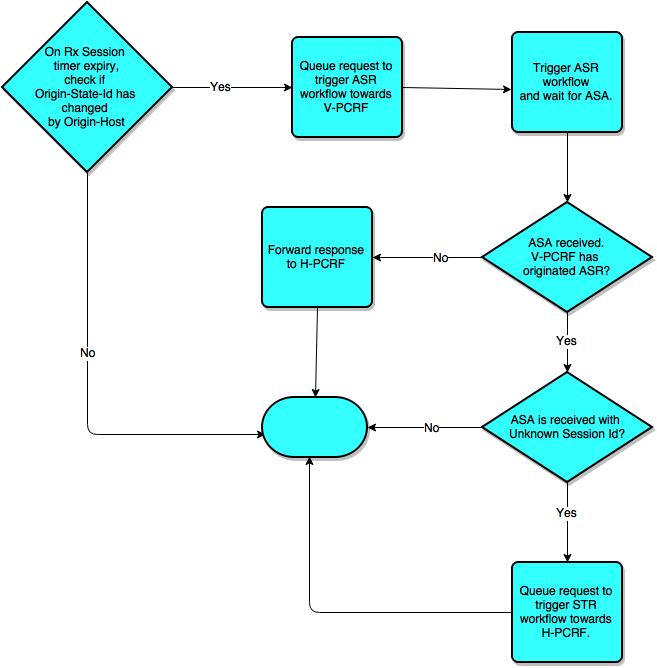


Figure : Handling Origin-State-ID change of AF -1

1. On expiry of Rx Session timer, queue a request to trigger STR workflow towards H-PCRF for Rx Session cleanup (no need of ASR here). As V-PCRF already come to know that Origin-State-Id has changed for AF and Rx Session with Old Origin-State-Id originated by this AF are no more active on AF, so triggering ASR towards AF is extra messaging(as it will be responded by ASA with DIAMETER\_UNKNOWN\_SESSION\_ID). Proceed with cleanup towards H-PCRF by triggering STR.

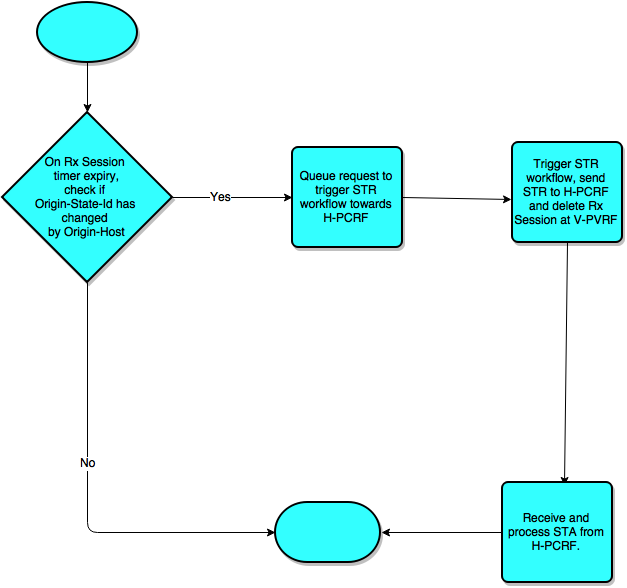


Figure : Handling Origin-State-ID change of AF - 2

### H-PCRF

#### System interaction

#### Service Designer

##### Installation of PCC Rules on S9

The incoming PCC rules on Gx interface for roaming subscribers (home accessed) shall result in installation of the PCC rules on S9 interface.

The PCC rules on S9 interface shall applicable at

* Command level: *\*[ QoS-Information ]* AVP in CCA & RAR message
* Sub-session level: *\*[ QoS-Information ]* AVP in Subsession-Decision-Info AVP in CCA & RAR message

As the QoS rules definition (*QoS-Information* AVP) is same for Gx and S9 interface, the idea is to reuse the existing Gx rule dictionary.

###### Design Option 1 – Mapping of S9Subsession to Gx access type

The idea is to map the newly added S9Subsession to existing Gx access type. Depending on the session type the PCC rules will be installed on either Gx or S9Home.

|  |
| --- |
| **File - ServiceDesignerS9Home.default.xml**  <?xml version="1.0" encoding="utf-8"?>  <Session>  <Name>S9HomeSubsession</Name>  <Attributes>  <Attribute>  <Name>Rules</Name>  <Type>cursor</Type>  <List>true</List>  <PointTo>AppliedRules::Gx</PointTo>  </Attribute>  <Attribute>  <Name>PendingRules</Name>  <Attribute>  <Name>Rules2BeInstall</Name>  <Type>cursor</Type>  <List>true</List>  <PointTo>PendingRules::Gx</PointTo>  </Attribute>  <Attribute>  <Name>Rules2BeUninstall</Name>  <Type>cursor</Type>  <List>true</List>  <PointTo>PendingRules::Gx</PointTo>  </Attribute>  <Attribute>  <Name>Rules2BeModifiedOld</Name>  <Type>cursor</Type>  <List>true</List>  <PointTo>AppliedRules::Gx</PointTo>  </Attribute>    <Attribute>  <Name>Rules2BeModifiedNew</Name>  <Type>cursor</Type>  <List>true</List>  <PointTo>PendingRules::Gx</PointTo>  </Attribute>  </Attribute>    <Attribute>  <Name>DeniedRules</Name>  <MapKey>string</MapKey>  <Type>integer</Type>  </Attribute>    </Attributes>  </Session> |

Figure : ServiceDesignerS9Home.default.xml

Register newly added S9 Access Type in Enforcement workflow processor.

|  |
| --- |
| **File: S9EnforcementWorkflowProcessor.cpp**  S9EnforcementWorkflowProcessor::S9EnforcementWorkflowProcessor() :  WorkflowProcessor(  S9Server::cS9EnforcementWorkflowProcessor /\* "S9EnforcementWorkflowProcessor” \*/),  ...)  {  ...  BusinessRulesCore::Instance().AddEnforcementImplementer(  S9Server::cS9EnforcementWorkflowProcessor);  BusinessRulesCore::Instance().RegisterAccessType(  S9Server::cS9EnforcementWorkflowProcessor, S9Server::cS9HomeSubsession /\* “S9HomeSubsession” \*/);  ...  } |

Figure : New S9-Enforcement-Workflow-Processor

In Service-Designer, map all newly added S9HomeSubsession session type to Gx access type.

|  |
| --- |
| **File: RuleDecision.cpp**  bool RuleDecision::BuildSessionByInterface(const boost::shared\_ptr<DataSession>& dataSession)  {  // Build a set of control session for access types  ...  // Build a set of control session for non-access types  ...    // Build a set of control session for S9Homesubsession type  dataSession->GetControlSessions(  “S9HomeSubsession”, m\_mapOfAccessSessions[“Gx”]);  if(!IsSessionEvaluationAllowed(m\_mapOfAccessSessions[“Gx”]))  {  return false;  }  } |

Figure : Exmaple - S9HomeSubsession session type map to Gx access type

**Status** – ACCEPTED

###### Design Option 2 – Reuse Gx Traffic & Session Rules

A new S9 Access Type shall be added. The newly added S9 Access Type shall reuse the existing Gx Traffic & Session rules schema: GxTrafficRules.xsd & GxSessionRules.xsd and translation & validation classes.

|  |
| --- |
| **File: AccessTypes.h**  //! This class defines the supported access types  namespace AccessTypes  {  enum TypeId  {  eNone,  eGx,  eSPB,  eAGNOSTIC,  ePCMM,  eS9HomeSubsession, // Added for S9  eMaxAccessType  };  static const std::string TypeNames[] = {  "",  "Gx",  "SPB",  "AGNOSTIC",  "PCMM",  "S9HomeSubsession" // Added for S9  };  typedef std::list <std::string> List;  } |

Figure : New S9 Access Type

|  |
| --- |
| **New File - S9AccessType.h**  class S9AccessType  {  public:  //! Returns the definition  static AccessTypeDefinition::Ptr GetDefinition();  };  **New File - S9AccessType.cpp**  AccessTypeDefinition::Ptr S9AccessType::GetDefinition()  {  // Traffic Rules  SchemaFileDefinition::Ptr s9TrafficRules (  new SchemaFileDefinition("GxTrafficRules.xsd", RuleSchema::TrafficRule));  // No translator required for S9 Predefined rules  RuleTypeDefinition::Ptr s9PredefinedRuleTypeDefinition(  new RuleTypeDefinition(GxRulesAttributes::cPCCPreDefined /\* "PCCPreDefined" \*/,  RuleValidator::Ptr(new GxRuleValidator())));  s9TrafficRules->AddRuleDefinition(s9PredefinedRuleTypeDefinition);  // Translator is required for S9 Dynamic rules  RuleTypeDefinition::Ptr s9DynamicRuleTypeDefinition(  new RuleTypeDefinition(GxRulesAttributes::cPCCDynamic /\* "PCCDynamic" \*/,  RuleValidator::Ptr(new GxRuleValidator()),  RuleTranslationFunction::Ptr(new GxDynamicRuleTranslationFunction())));  s9TrafficRules->AddRuleDefinition(s9DynamicRuleTypeDefinition);  AccessTypeDefinition::Ptr s9AccessType (  new AccessTypeDefinition(AccessTypes::eS9HomeSubsession /\* 5, "S9HomeSubsession" \*/,  s9TrafficRules,  PostDecisionValidator::Ptr(new GxPostDecisionValidator()),  PostReloadValidator::Ptr(new GxPostReloadValidator()),  RuleReportHandler::Ptr(new GxRuleReportHandler())));      // Session Rules  SchemaFileDefinition::Ptr s9SessionRules (  new SchemaFileDefinition("GxSessionRules.xsd", RuleSchema::SessionRule));  // Translator is required for S9  RuleTypeDefinition::Ptr s9SessionRuleTypeDefinition(  new RuleTypeDefinition(GxRulesAttributes::cGxSession /\* "GxSessionRule" \*/,  RuleValidator::Ptr(),  RuleTranslationFunction::Ptr(new GxSessionRuleTranslationFunction())));  s9SessionRules->AddRuleDefinition(s9SessionRuleTypeDefinition);  // No translator required for S9 default Session rules  RuleTypeDefinition::Ptr s9DefSessionRuleTypeDefinition(  new RuleTypeDefinition(GxRulesAttributes::cGxDefaultSessionRule /\* "GxDefaultSessionRule" \*/));    s9SessionRules->AddRuleDefinition(s9DefSessionRuleTypeDefinition);  s9AccessType->AddSchemaFile(s9SessionRules);    return s9AccessType;  } |

Figure : S9 Access Type Class definition reusing Gx Rules

Define newly added S9 Access Type in Rule schema manger.

|  |
| --- |
| **File: RuleSchemaManager.h**  void RuleSchemaManager::DefineAccessTypes()  {  DefineAccessType(AgnosticAccessType::GetDefinition());  DefineAccessType(GxAccessType::GetDefinition());  DefineAccessType(S9AccessType::GetDefinition()); // Added for S9  DefineAccessType(SPBAccessType::GetDefinition());  DefineAccessType(PCMMAccessType::GetDefinition());  } |

Figure : S9 Access Type added in Rule-Schema-Manager

**Status** – REJECTED

###### Design Option 3 – New S9 Traffic & Session Rules

A new S9 Access Type shall be added. The newly added S9 Traffic & Session rules schema will be same as existing Gx Traffic & Session rules schema.

The S9 specific rules schema files: S9TrafficRules.xsd & S9SessionRules.xsd shall be added (to start same as GxTrafficRules.xsd & GxSessionRules.xsd respectively).

|  |
| --- |
| **New File - S9AccessType.h**  class S9AccessType  {  public:  //! Returns the definition  static AccessTypeDefinition::Ptr GetDefinition();  };  **New File - S9AccessType.cpp**  AccessTypeDefinition::Ptr S9AccessType::GetDefinition()  {  // Traffic Rules  SchemaFileDefinition::Ptr s9TrafficRules (  new SchemaFileDefinition("S9TrafficRules.xsd", RuleSchema::TrafficRule));  // No translator required for S9 Predefined rules  RuleTypeDefinition::Ptr s9PredefinedRuleTypeDefinition(  new RuleTypeDefinition(S9RulesAttributes::cPCCPreDefined /\* "PCCPreDefined" \*/,  RuleValidator::Ptr(new S9RuleValidator())));  s9TrafficRules->AddRuleDefinition(s9PredefinedRuleTypeDefinition);  // Translator is required for S9 Dynamic rules  RuleTypeDefinition::Ptr s9DynamicRuleTypeDefinition(  new RuleTypeDefinition(S9RulesAttributes::cPCCDynamic /\* "PCCDynamic" \*/,  RuleValidator::Ptr(new S9RuleValidator()),  RuleTranslationFunction::Ptr(new S9DynamicRuleTranslationFunction())));  s9TrafficRules->AddRuleDefinition(s9DynamicRuleTypeDefinition);  AccessTypeDefinition::Ptr s9AccessType (  new AccessTypeDefinition(AccessTypes::eS9 /\* 5, "S9" \*/,  s9TrafficRules,  PostDecisionValidator::Ptr(new S9PostDecisionValidator()),  PostReloadValidator::Ptr(new S9PostReloadValidator()),  RuleReportHandler::Ptr(new S9RuleReportHandler())));      // Session Rules  SchemaFileDefinition::Ptr s9SessionRules (  new SchemaFileDefinition("S9SessionRules.xsd", RuleSchema::SessionRule));  // Translator is required for S9  RuleTypeDefinition::Ptr s9SessionRuleTypeDefinition(  new RuleTypeDefinition(S9RulesAttributes::cS9Session /\* "S9SessionRule" \*/,  RuleValidator::Ptr(),  RuleTranslationFunction::Ptr(new S9SessionRuleTranslationFunction())));  s9SessionRules->AddRuleDefinition(s9SessionRuleTypeDefinition);  // No translator required for S9 default Session rules  RuleTypeDefinition::Ptr s9DefSessionRuleTypeDefinition(  new RuleTypeDefinition(S9RulesAttributes::cS9DefaultSessionRule /\* "S9DefaultSessionRule" \*/));    s9SessionRules->AddRuleDefinition(s9DefSessionRuleTypeDefinition);  s9AccessType->AddSchemaFile(s9SessionRules);    return s9AccessType;  } |

Figure : S9 Access Type Class definition with new S9 Rules

**Note**:

The following C++ classes shall be added for translation & validation:

1. S9RuleValidator
2. S9PostDecisionValidator
3. S9PostReloadValidator
4. S9DynamicRuleTranslationFunction
5. S9SessionRuleTranslationFunction
6. S9RuleReportHandler
7. S9TrafficClassifierCheck

The definition of these classes shall be similar to corresponding Gx classes. Thus they shall be inherited from Gx classes and any S9 specific code shall be added on top of it.

**Status** – REJECTED

In this approach the same Rule-Name can’t be used at both Gx & S9 interface for a subscriber. Thus the policy author shall be forced to use different Rule-Name for Gx & S9.

#### GxServer Subsystem

Gx Sanity Step needs to refactor for parsing AVPs once the common APIs were implemented as mentioned in section “Gx Diameter Message to S9 Diameter Message” (B).

#### S9ServerSubsystem

##### S9ServerSessionEstablishment

When H-PCRF receives S9 CCR-I request, S9ServerSessionEstablishment workflow is created. The state machine for the workflow is as below

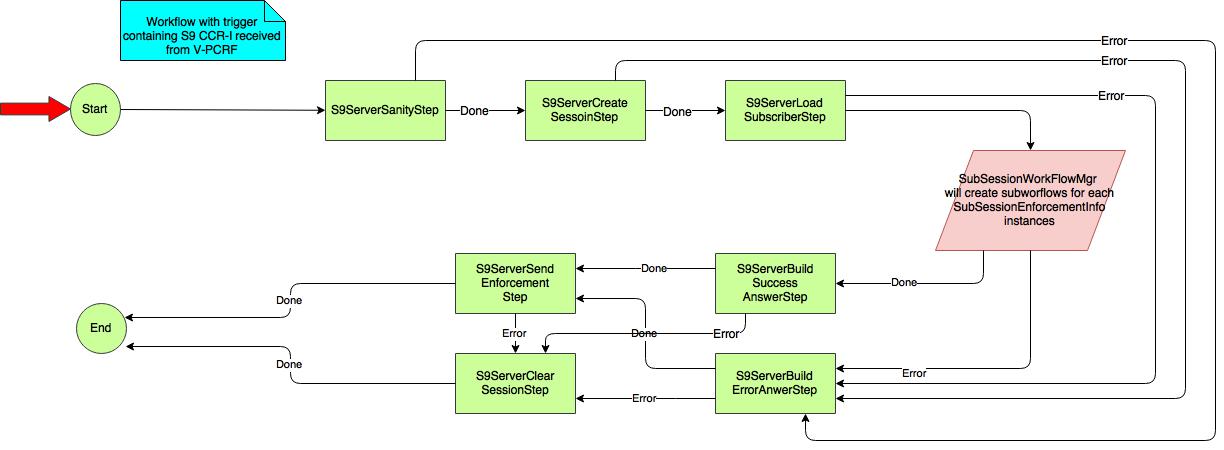


Figure : S9ServerSessionEstablishment workflow state machine

|  |
| --- |
| 1. **S9ServerSanityStep:**   bool S9ServerSanityStep::Run()  {   1. Perform Sanity for the command level AVPs 2. If there are any errors in sanity, transit to S9SeverBuildErrorAnswer Step and reject the S9 CCR-I with appropriate Diameter Error Code INVALID\_AVP/MISSING\_MANDATORY\_AVP 3. Create a diam outgoing message and fill the command level AVPs (**Only in case of Proposal -1** ) 4. Create a vector of size number of SubSessionEnforcmentInfo instances present in the incoming S9 message. This message hold the reference of S9SubSessionDecisionInfo 5. On successful, transit to S9ServerCreateSessionStep.   }     1. **S9CreateSessionStep:**   S9CreateSessionStep will create the parent S9 control session.  bool S9ServerCreateSession::Run()  {   1. If received S9 session ID is already exist, reject the S9 CCR-I with diameter error code SESSION\_ALREADY\_EXISTS   If Maximum number of permission S9 session are reached, reject the S9 CCR-I with diameter error code DIAMETER\_OUT\_OF\_SPACE  If S9 session creation is failed, reject the S9 CCR-I with diameter error code DIAMETER\_UNABLE\_TO\_COMPLY   1. If there is a session already present for the received roaming Subscriber, terminate the old S9 session by initiating S9 RAR-T and create new S9 session only after receiving S9 CCR-T for the old Session. 2. On Successful, transit to S9ServerLoadSubscriber step   }       1. **S9ServerLoadSubscriberStep**   bool S9ServerLoadSubscriberStep::Run()  {   1. If this step fails, transit to S9ServerBuildErrorAnswerStep 2. The S9SubSessionWorkFlowManager, based on the number of SubSessionEnforcement Info instances present in the incoming S9 CCR-I message, it triggers those many S9SubSesiion workflows. On completion of all the sub workflows, S9SubSessionWorkFlowManager will transit to either S9ServerBuildSuccessAnswerStep (If all the subSession operation is resulted in success) or to S9ServerBuildErrorAnswer (If at least one of the SubSession operation is resulted in failure).   }     1. **S9ServerBuildSuccessAnswerStep**   This step will get executed only after the successful completion of all the triggered sub Work Flows  bool S9ServerBuildSuccessAnswerStep::Run()  {   1. Builds the Successful S9 CCA-I/R/T message.   In case of Proposal-1, since the subworkflows would have already encoded SubSessionDecisionInfo, this step updates the command level result code.  In case of Proposal-2, in this step the outgoing diameter message is created and encoded with the SubSessionDecisionInfo provided by Sub Workflows   1. If building of S9 CCA message is failed, the step transits to S9ServerClearSessionStep.   }     1. **S9ServerBuildErrorAnswerStep**   This step will get executed only after the successful completion of all the triggered sub Work Flows  bool S9ServerBuildErrorAnswerStep::Run()  {   1. Builds the Successful S9 CCA-I/R/T message.   In case of Proposal-1, since the subworkflows would have already encoded SubSessionDecisionInfo, this step updates the command level result code.  In case of Proposal-2, in this step the outgoing diameter message is created and encoded with the SubSessionDecisionInfo provided by Sub Workflows   1. If building of S9 CCA message is failed, the step transits to S9ServerClearSessionStep.   }     1. **S9ServerSendEnforcementStep**   bool S9ServerSendEnforcementStep::Run()  {   1. In this setp, the diameter outgoing message will be sent to the peer.   If Peer is not reachable, this step will transit to S9ServerClearSessionStep  }     1. **S9ServerClearSessionStep**   bool S9ServerClearSessionStep::Run()  {   1. Session manager deletes the created S9 Session and its corresponding S9 Sub Sessions are deleted   } |

Figure : LLD for S9ServerSessionEstablishment workflow

###### Overlapping IP (Missed logout) support

On receiving S9-CCR-I with an overlapping IP-Address & Site-Qualifier (If a Gx session OR S9 subsession already exists with same IP-Address & Site-Qualifier for different subscriber), the S9Server shall

* If Gx session exists, initiate termination of existing Gx session by sending Gx-RAR-T to PCEF. On receiving Gx-RAA-T, the Gx session is successfully terminated (followed by exchange of Gx-CCR-T & Gx-CCA-T)
* If S9 subsession exists, initiate termination of existing S9 subsession by sending S9-RAR-T to V-PCRF. Refer Section S9ServerSessionReauthorization- Send S9-RAR-T and wait for S9-RAA-T (successful case). Then wait for S9-CCR-U/T and complete termination with S9-CCA-U/T
* Establish new S9 subsession with the same IP-Address & Site-Qualifier

**Note** – If the Gx session exists for the same subscriber, terminate the existing Gx session by sending Gx-RAR-T and establish new S9 subsession

**Note** – If the S9 subsession exists for the same subscriber, overwrite the existing S9 subsession with new S9 subsession with the latest S9-CCR-I message.

**Note** – If the S9-CCR-I results in creating two subsessions with same IP-Address & Site-Qualifier for the same subscriber, overwrite the first created S9 subsession with second S9 subsession

**Note** – Check for overlapping IP MUST be performed at **CreateSession** step of S9ServerSessionEstablishment workflow.

###### Deferred IP allocation support

In case of deferred IP allocation (UE IP allocation deferred during default bearer establishment) on receiving S9-CCR-I without framed IP Address, the S9 session/ subsession shall be created.

On UE IP allocation at V-PCRF notifies the H-PCRF in the next S9-CCR-U with UE\_IP\_ADDRESS\_ALLOCATE event. The existing S9 session/ subsession shall be bound to the framed IP Address.

**Note** – In this case on receiving S9-CCR-U, while updating framed IP Address in S9 session/ subsession, check for Overlapping IP and handle the scenario as explained in Overlapping IP (Missed logout) support.

**Note** – Similar on UE\_IP\_ADDRESS\_RELEASE event notification in S9-CCR-U from V-PCRF, the framed IP Address is unbound from S9 subsession.

##### S9ServerSessionModification

When H-PCRF receives S9 CCR-U request, S9ServerSessionModification workflow is created. The state machine for the workflow is as below

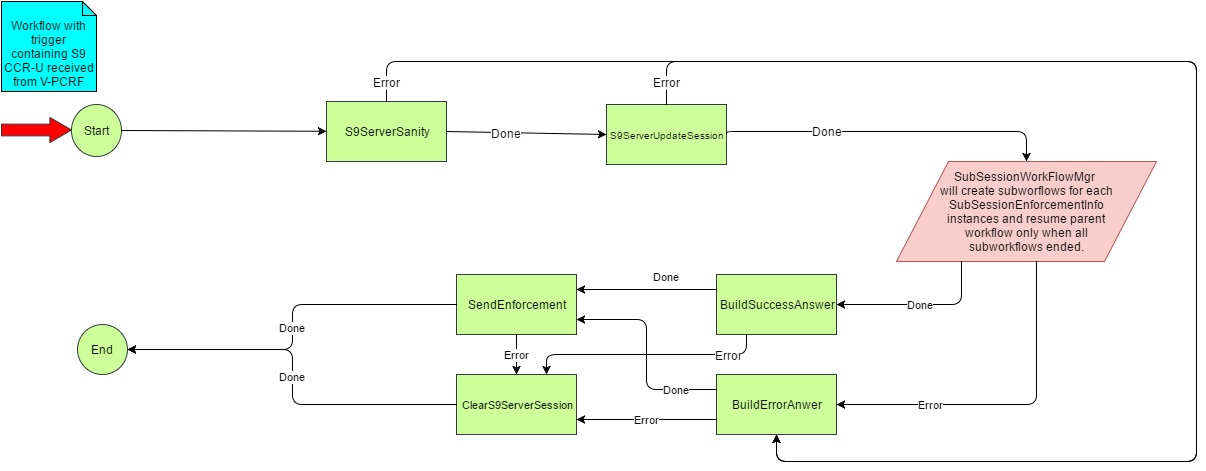


Figure : S9ServerSessionModification workflow state machine

|  |
| --- |
| **1. S9ServerSanityStep:**  bool S9ServerSanityStep::Run()  {   1. Perform Sanity for the command level AVPs 2. If there are any errors in sanity, transit to S9SeverBuildErrorAnswer Step and reject the S9 CCR-U with appropriate Diameter Error Code INVALID\_AVP/MISSING\_MANDATORY\_AVP 3. Create a diam outgoing message and fill the command level AVPs (**Only in case of Proposal -1** ) 4. Create a vector of size number of SubSessionEnforcmentInfo instances present in the incoming S9 message. This message hold the reference of S9SubSessionDecisionInfo 5. On successful, transit to S9ServerUpdateSessionStep.   }     1. **S9UpdateSessionStep:**   S9UpdateSessionStep will update the parent S9 control session.  bool S9ServerUpdateSession::Run()  {   1. If S9 Session doesn’t exist for Session-Id received in CCR-U message, workflowData is updated with DIAMETER\_UNKNOW\_SESSION\_ID and workflow will transition to BuildErrorAnswer and step return code is Error. 2. If S9 session exists, update S9 Session with information received in CCR-U message and stored in workflowData during S9ServerSanity Step. 3. On Successful, transit to S9ServerSubworkflow step.   } |

###### Overlapping IP (Missed logout) support

On receiving S9-CCR-U with an overlapping IP-Address & Site-Qualifier (If a Gx session OR S9 subsession already exists with same IP-Address & Site-Qualifier for different subscriber), the S9Server shall

* If Gx session exists, initiate termination of existing Gx session by sending Gx-RAR-T to PCEF. On receiving Gx-RAA-T, the Gx session is successfully terminated (followed by exchange of Gx-CCR-T & Gx-CCA-T)
* If S9 subsession exists, initiate termination of existing S9 subsession by sending S9-RAR-T to V-PCRF. Refer Section S9ServerSessionReauthorization- Send S9-RAR-T and wait for S9-RAA-T (successful case). Then wait for S9-CCR-U/T and complete termination with S9-CCA-U/T
* Modify S9 session by adding new S9 subsession with the same IP-Address & Site-Qualifier

**Note** – If the Gx session exists for the same subscriber, terminate the existing Gx session by sending Gx-RAR-T and add new S9 subsession in S9 session

**Note** – If the S9 subsession exists for the same subscriber, overwrite the existing S9 subsession with new S9 subsession with the latest S9-CCR-U message.

**Note** – Check for overlapping IP MUST be performed at **UpdateSession** step of S9ServerSessionEstablishment workflow.

###### Deferred IP allocation support

In case of deferred IP allocation (UE IP allocation deferred during default bearer establishment) on receiving S9-CCR-U without framed IP Address, the S9 subsession shall be created.

On UE IP allocation at V-PCRF notifies the H-PCRF in the next S9-CCR-U with UE\_IP\_ADDRESS\_ALLOCATE event. The existing S9 subsession shall be bound to the framed IP Address.

**Note** – In this case on receiving S9-CCR-U, while updating framed IP Address in S9 session/ subsession, check for Overlapping IP and handle the scenario as explained in Overlapping IP (Missed logout) support.

**Note** – Similar on UE\_IP\_ADDRESS\_RELEASE event notification in S9-CCR-U from V-PCRF, the framed IP Address is unbound from S9 subsession.

##### S9ServerSessionTermination

When H-PCRF receives S9 CCR-T request, S9ServerSessionTermination workflow is created. The state machine for the workflow is as follows

Design option 2: This design has 2 proposals

**The Proposal-2 is accepted and shall be commenced for coding.**

**Proposal 1:**

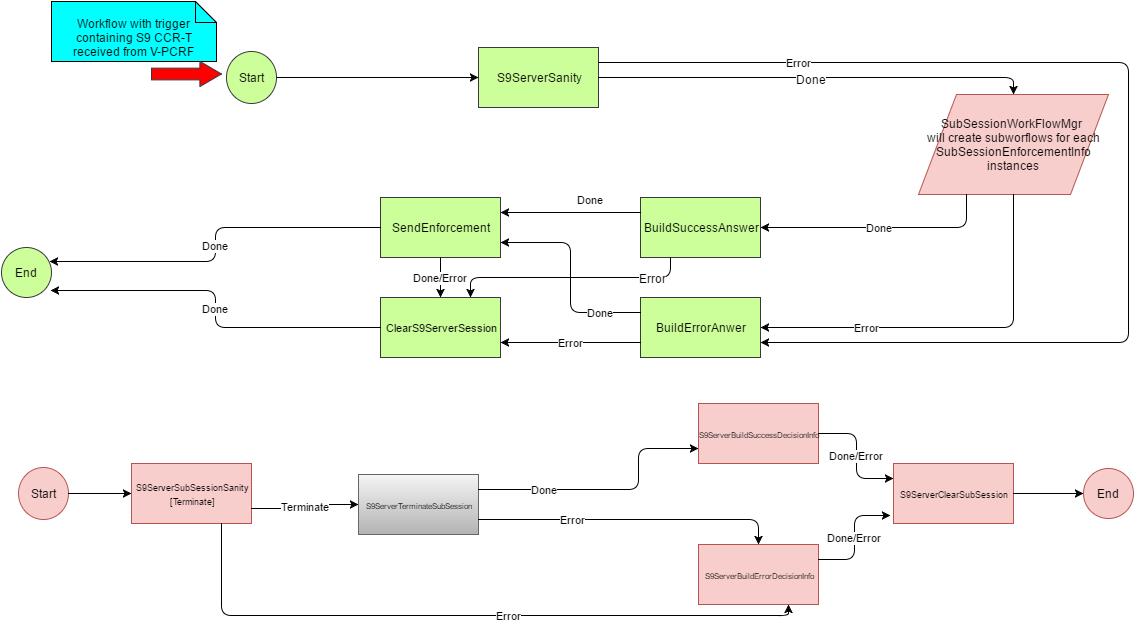


Figure : S9ServerSessionTermination workflow state machine – P1

**Proposal 2:**

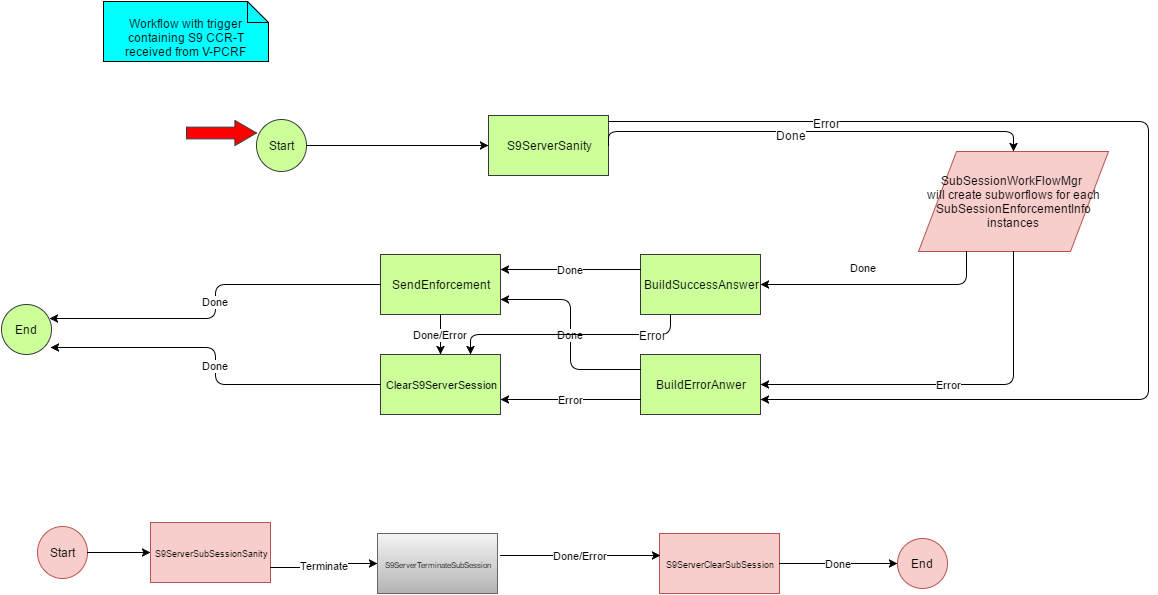
****

Figure : S9ServerSessionTermination workflow state machine – P2

Low Level details for S9ServerSessionTerminate steps:

|  |
| --- |
| Parent Workflow steps: S9ServerSanity, BuildSuccessAnswer, BuildErrorAnswer, ClearS9ServerSession, SendEnforcement.  S9SubSession workflow steps: S9ServerTerminateSubSession, S9ServerBuildSuccessDecisionInfo, S9ServerBuildErrorDecisionInfo, S9ServerClearSubSession.   1. S9ServerSanity::Run()   /\*This is Parent workflow step\*/  {  /\*  -> Command level AVP parsing:  If parsing fails, go to BuildErrorAnswer -> SendEnforcement->ClearSession->End. In this ClearSession scenario should take care of deleting parent S9 Session and SubSessions as well.\*/  If parsing is successful, -> CreateSubWorkflows for SubSessions.  -> Create Command level CCA message and pass this Diameter Msg pointer to SubWorkflows to create and append SubSessionDecisionInfo.  -> Create SubWorkflows for SubSession Sanity.  Note:  WorkflowEngine framework create a step for Parent workflow "SubWorkflowStep" where parent workflow will be blocked and will be waiting for all subWorkflows to report once they are done with their job.  Parent workflow will maintain std::vector<SubSessionIds, SubWorkflows>.  }   1. S9ServerSubSessionSanity::Run()   /\*This is SubSession's SubWorkflow\*/  {  /\*  -> Parse SubSessionEnforcementInfo Group AVP(s). As an input will receive these parameters from SubWorkflowMgr:  (a) start pointer of SubSessionEnforcementInfo.  (b) TinyDiameter::AvpGroupWriter  -> If AVP parsing fails -> Set the status in Parent's WorkflowData and Change state to S9ServerBuildErrorDecisionInfo Step.  -> If AVP parsing is successful -> Set the status in Parent's workflowData as success and change state to S9ServerBuildSuccessDecisionInfo Step.  \*/  }   1. S9ServerSubSessionBuildSuccessDecisionInfoStep::Run()   /\*This is SubSession's SubWorkflow\*/  {  /\*  -> Create SubSessionDecisionInfo grouped AVP with success status of this SubSession and append to the Diameter Msg (having command level AVPs encoded) passed to it.  -> set status of this subSession to parent's workflow data.  \*/  }   1. S9ServerSubSessionBuildErrorDecisionInfoStep::Run()   /\*This is SubSession's SubWorkflow\*/  {  /\*  -> Create SubSessionDecisionInfo grouped AVP with error status of this SubSession and append to the Diameter Msg (having command level AVPs encoded) passed to it.  -> set status of this subSession to parent's workflow data.  \*/  }   1. S9ServerClearSubSession::Run()   /\*This is SubSession's SubWorkflow\*/  {  /\*  Control comes here either subSession message encoding is success/failed to Delete SubSession.  \*/  }   1. S9BuildSuccessAnswer::Run()   /\*This is Parent workflow step\*/  {  /\* Control Comes to this step only when parsing of Command level and SubSession Level both are successful.  For example for S9 CCR-T with 2 subSessions, the CCA-T message will look like as:  CCA-T:  Result-Code: Success. (Command level)  SubSessionDecisionInfo:  Result-Code: Success (SubSession Level)  SubSessionDecisionInfo:  Result-Code: Success (SubSession Level)  SubSessionDecisionInfo:  Result-Code: Success (SubSession Level)  \*/  }   1. S9BuildErrorDecisionAnswer::Run()   /\*This is Parent workflow step\*/  {  /\*Control comes into this step when:  -> Command level AVP parsing failed.  -> Either one of the SubSession level AVP parsing fails.  For example for S9 CCR-T with 2 subSessions, the CCA-T message will look like as:  CCA-T:  Experimental-Result-Code: SubSession Error. (Command level)  SubSessionDecisionInfo:  Result-Code: Success (SubSession Level)  SubSessionDecisionInfo:  Result-Code: Success (SubSession Level)  SubSessionDecisionInfo:  Experiemental-Result-Code: Error (SubSession Level  \*/  }   1. S9SendEnforcement::Run()   /\*This is parent's workflow step\*/  {  /\*  This step will send message to diameter stack.  \*/  }   1. ClearS9ServerSession::Run()   /\*This is parent's workflow step\*/  {  /\*  -> This step will delete parent S9 session.  -> This is possible that control comes here when command level sanity fails. Still need to check if any subSession exists than delete SubSession as well.  \*/  } |

Figure : LLD for S9ServerSessionTermination workflow

##### S9ServerSubSessionWorkFlow

In proposal-1 the S9 diam CCA message is created and passed to each of the subworkflows by filling in Command Level AVPS. Each subworkflow will update the respective SubSessionDecisionInfoAVPs.

After the completion of Subworkflow, main work flow will transit either to BuildSuccessAnswer or BuildErrorAnswer step and in this step, only the command level result code is updated and S9 CCA diam message is sent out.

In Proposal-2, the subworkflow will give the shared pointer of the SubSessionDecisionInfo Object and when the main workflow resumes with either BuildSuccessAnswer or BuildErrorAnswer step and these steps will prepare the outgoing S9 CCA message and send it out.

**The Proposal-2 is accepted and shall be commenced for coding.**

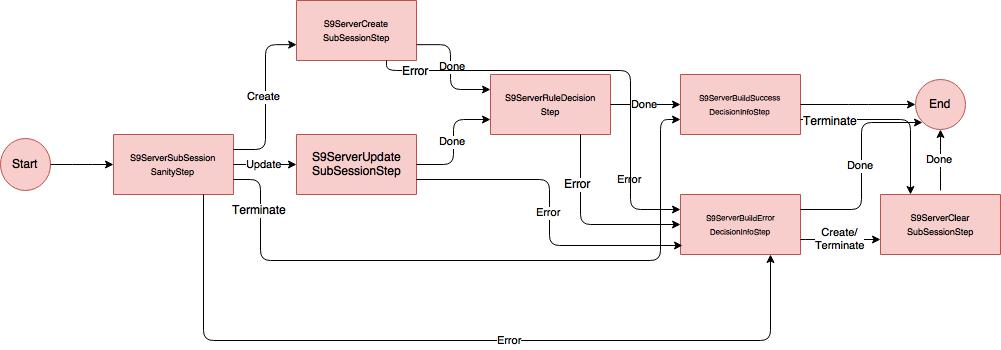
****

Figure : S9ServerSubSession workflow state machine – P1

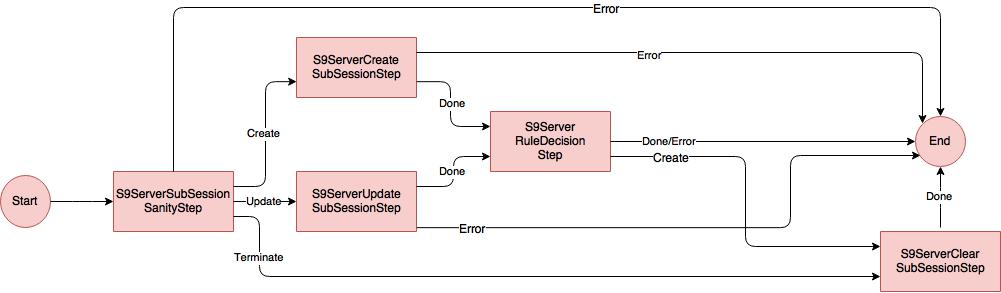


Figure : S9ServerSubSession workflow state machine – P2

|  |
| --- |
| 1. **S9ServerSubsessionSanityStep**   bool S9ServerSubSessionSanityStep::Run()  {   1. Perform Sanity for the SubSessionEnforcemntInfo AVP 2. If sanity resulted in error, encode the SubSessionDecisionInfo AVP and end the subWorkflow (Proposal-1)   If sanity resulted in error, end the subWorkflow (Proposal-2)   1. If SubSessionSanity step resulted in Success, then based on the result code transit to S9ServerCreateSubessionStep/S9ServerUpdateSubessionStep/S9ServerTerminateSubessionStep.   }     1. **S9ServerCreateSubSessionStep**   S9CreateSubSessionStep will create the S9 control session and as well as data session for the received Sub Session.  bool S9ServerSubSessionSanityStep::Run()  {   1. If received S9 sub session ID is already exist, reject the SubSessionEnforcement Info received in S9 CCR-I with SubSessionDecisionInfo result code set to SESSION\_ALREADY\_EXISTS   If Maximum number (total session allowed globally) of permission S9 session are reached, reject the SubSessionEnforcement Info received in S9 CCR-I with SubSessionDecisionInfo result code set to DIAMETER\_OUT\_OF\_SPACE  If S9 sub-session creation is failed, reject the SubSessionEnforcement Info received in S9 CCR-I with SubSessionDecisionInfo result code set to DIAMETER\_UNABLE\_TO\_COMPLY   1. If both framedIPV4 and framedIpV6 are present, S9 session (Data and Control Session) will be created with Ipv4 and IpV6 will be attached to this data session 2. If only either of framed IP Address is present, then the S9 session (Data and Control Session) is created with the available framed IP address 3. If neither of the framed IP address is present, then only the S9 control session is created. 4. On Success, transit to S9ServerRuleDecisionstep 5. On failure, transit to S9ServerBuildErrorDecisionInfoStep (Proposal-1)   On failure, end the subWorkflow (Proposal-2)  }   1. **S9ServerUpdateSubSessionStep**   This step will update the corresponding Sub-session with information received within Subsession-Enforcement-Info AVP for the SubSession identified by Subsession-Id AVP value.  bool S9ServerSubSessionSanityStep::Run()  {   1. If S9 sub-session doesn’t exist, reject the SubSessionEnforcement Info received in S9 CCR-U with SubSessionDecisionInfo result code set to DIAMETER\_UNKONWN\_SESSION\_ID. 2. If S9 Sub-Session exists, update S9 Sub-Session with information received in Subsession-Enforcement-Info AVP stored in subworkflowData. 3. On Success, transit to S9ServerRuleDecisionstep 4. On failure, transit to S9ServerBuildErrorDecisionInfoStep (Proposal-1),   On failure, end the subWorkflow (Proposal-2)  }     1. **S9ServerRuleDecisionStep**   bool S9ServerSubSessionSanityStep::Run()  {   1. Triggers the business rules feature evaluation for the S9 Subsession (Control Session) 2. On failure, transit to S9ServerBuildErrorDecisionInfoStep (Proposal-1)   On failure, end the subWorkflow (Proposal-2)    }    **e.S9ServerClearSubSessionStep**  bool S9ServerClearSubSessionStep::Run()  {   1. Session Manager deletes the sub session information 2. If Sub Session is not present, reply back with the Diam error code   } |

Figure : LLD for S9ServerSubSession workflow state machine

##### S9ServerSessionReauthorization

Due to the internal triggers, Business Rules will trigger an Enforcement-Request for each data-session OR all data-session(s) for a S9 session, which will result in S9 RAR from H\_PCRF to V\_PCRF. This section explains the handling of Enforcement-Request(s).

|  |
| --- |
| File: S9ServerEnforcementWorkflowProcessor.cpp  class S9ServerEnforcementWorkflowProcessor : public WorkflowProcessor  {  typedef struct EnforcementRequestsInfo  {  int numOfEnforcementRequests;  std::vector <boost::shared\_ptr<OutgoingMessage> enforcementRequests;  } EnforcementRequestsInfo;  typedef std::map<std::string, EnforcementRequestsInfo> EnforcementRequestMap;    private:  //! stores all the EnforcementRequest message from the BusinessRuleCore module for a s9session  EnforcementRequestMap m\_pEnforcementRequests;  WorkflowDeclaration::Key\_Type m\_rarWorkflowKey;  public:  //! ctor  S9ServerEnforcementWorkflowProcessor();  //! dtor  virtual ~S9ServerEnforcementWorkflowProcessor();  //! Get Outgoing message from the businessrulecore.  //! Implement the WorkflowProcessor interface.  //! @param apTrigger Trigger of the workflow  //! @return workflow workflow  virtual Workflow::Ptr GetNextWorkflowTrigger(WorkflowTrigger::Ptr& apTrigger);  //! Adds the workflow declarations of this flowProcessor.  virtual void InitWorkflowDeclarations();  private:  //! Processes outgoing message received from business rules and decides  //! the next course of action for this workflow.  //! It's a helper method to GetNextWorkflowTrigger.  //! @param lpMessage Message from business rules.  //! @param apTrigger Trigger of the workflow  //! @return workflow workflow  Workflow::Ptr GetNextWorkflowTriggerHelper(  const boost::shared\_ptr<OutgoingMessage>& lpMessage,  WorkflowTrigger::Ptr& apTrigger);  }; |

Figure : S9ServerEnforcement workflow processor class definition

|  |
| --- |
| **File: S9ServerEnforcementWorkflowProcessor.cpp**  Workflow::Ptr S9ServerEnforcementWorkflowProcessor::GetNextWorkflowTriggerHelper(  const OutgoingMessage::Ptr& lpMessage,  WorkflowTrigger::Ptr& apTrigger)  {  switch (lpMessage->Type())  {  case OutgoingMessage::EnforcementRequest:  {  EnforcementRequestsInfo mapEntry;    // 1. Get the parent S9 session from the data session  std::string s9SessionID = lpMessage->GetSession().GetParentSessionID();    // 2. Check if an entry exists for s9SessionID in m\_pEnforcementRequests  EnforcementRequestMap::iterator it =  m\_pEnforcementRequests.find(s9SessionID);    if (it == m\_pEnforcementRequests.end())  {  // 2.1 No entry in m\_pEnforcementRequests    // a. Store this enforcement request  mapEntry.enforcementRequests.insert(lpMessage);    // b. For all other data-session(s) in S9Session  for( S9Subsession s9subsession in S9Session)  {  // c. If any pending evaluation for data session increment numOfEnforcementRequests  if ( BusinessRulesCore::Instance().IncomingMessageManager::IsDataSessionReadyForEvaluation(s9subsession.GetSessionID()) )  {  mapEntry.numOfEnforcementRequests++;  }  }  if (mapEntry.numOfEnforcementRequests == 0)  {  // d. If no more enforcement request for this s9Session  return S9ServerEnforcementWorkflowProcessor(s9SessionID, mapEntry);  }  else  {  // e. Insert ENTRY - mapEntry in MAP - m\_pEnforcementRequests with KEY - s9SessionID  m\_pEnforcementRequests.insert( std::pair<std::string,EnforcementRequestsInfo>(s9SessionID, mapEntry) );  }  }  else  {  // 2.2 Entry found in m\_pEnforcementRequests    mapEntry = it.second;    // a. Store this enforcement request  mapEntry.enforcementRequests.insert(lpMessage);    // b. Received EnforcementRequest decrement numOfEnforcementRequests  mapEntry.numOfEnforcementRequests--;    // c. Remove old mapEntry from MAP  m\_pEnforcementRequests.erase(it);    if (mapEntry.numOfEnforcementRequests == 0)  {  // c. If no more enforcement request for this s9Session  return S9ServerEnforcementWorkflowProcessor(s9SessionID, mapEntry); }  else  {  // d. Insert new ENTRY - mapEntry in MAP - m\_pEnforcementRequests with KEY - s9SessionID  m\_pEnforcementRequests.insert( std::pair<std::string,EnforcementRequestsInfo>(s9SessionID, mapEntry) );  }  }  return Workflow::Ptr();  }  }  } |

Figure : Processing of outgoing message received from business rules

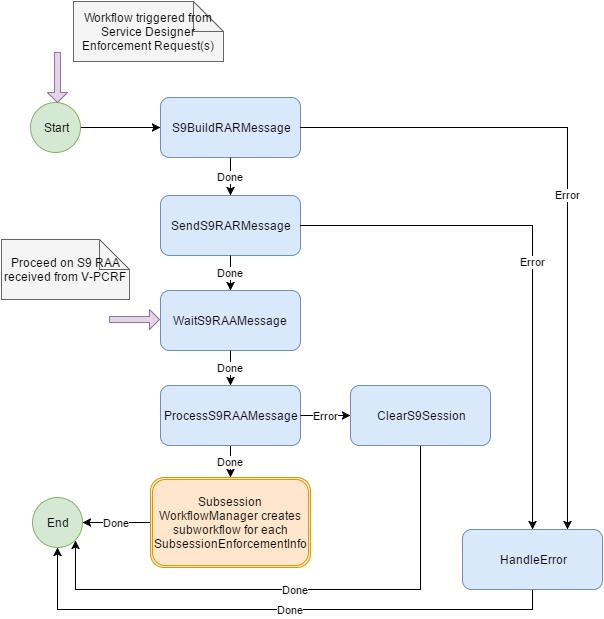


Figure : S9ServerEnforcement workflow state machine

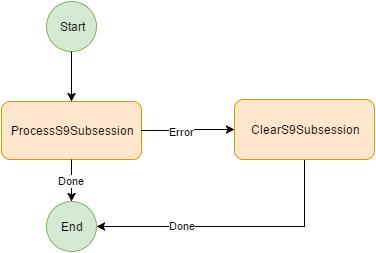


Figure : S9ServerEnforcement sub-workflow state machine

|  |
| --- |
| **1. S9BuildRARMessage Step**  Build S9 RAR Message for each Enforcement-Request from Business-Rule build a S9 Subsession-Decision-Info.  On error, go to HandleError Step  **2. SendS9RARMessage Step**  Send S9 RAR Message to Base-Diameter to send it to V-PCRF  On error, go to HandleError Step  **3. WaitS9RAAMessage Step**  Block on this step & wait for S9 RAA Message from V-PCRF. On failure Base-Diameter will send RAA Message with failure.  **4. ProcessS9RAAMessage Step**  On receiving RAA message check Result-Code at command level.  On failure (SESSION\_UNKNOWN OR PERMANENT\_ERROR) go to ClearS9Session Step.  On success create Sub-workflow for each Subsession-Enforcement-Info and on completion of all sub-workflow, end the workflow  **5. ClearS9Session Step**  Clear S9 session & all its subsession and end the workflow  **6. HandleError Step**  Handle error and send Ack-Rule to Service-Designer and end the workflow |

Figure : Pseudo code for S9ServerEnforcement workflow

|  |
| --- |
| **1. ProcessS9Subsession Step**  Check Result-Code at sub-session level.  On failure (SESSION\_UNKNOWN OR PERMANENT\_ERROR) go to ClearS9Subsession Step.  On failure (QoS Validation Failure):   * Store Acceptable-QoS in subsession workflow data * Send Ack-Rule to Service-Designer * Call Service-Designer Evaluate-Session * End the workflow   On success send Ack-Rule to Service-Designer and end the workflow  **2. ClearS9Subsession Step**  Clear S9 sub-session and end the workflow |

Figure : Pseudo code for S9ServerEnforcement sub-workflow

#### S9 Server Sanity Step

CCR-I Sanity Step:

|  |
| --- |
| 1. S9ServerSanity::Run()   {  /\*S9Server SessionEstablishment\*/  For(Each SubSessionInfo instance)  {  ParseCCRInitialMessage(parser);  }  } |

 If command level sanity fails, will discard the whole S9 CCR-I message and will sent out CCA-I with error code.

If either one of the Subsession Sanity fails, will respond CCA-I for that particular SubSession with error code and other(s) with successful result code.

1. S9 Server CCR-I/U/T and RAA message AVP parsing is the same as Gx Server.
2. According to the current implementation, Gx message parsing is local to Gx server. These AVP parsing should be at common library i.e. diameterAppAvpUtils.
3. Function prototype to parse grouped AVP in diameterAppAvpUtils as:

|  |
| --- |
| Bool DiameterAppParser::ParseChargingRuleReport(  const TinyDiameter::DiameterInputParser& parser,  const TinyDiameter::AvpKey& avpKey,  std::vector< TinyDiameter::AvpStringType>& rawBytes)  {  Uint32\_t chargingRuleReportInstances;  for(;;++chargingRuleReportInstances)  {  /\*  Get each child AVP instance of each chargingRuleReportInstance and push\_back the same to argument rawBytes.  \*/  }  return true;    void S9ServerSanityStep::ParseChargingRuleReport(  TinyDiameter::DiameterInputParser& parser)  {  std::vector< TinyDiameter::AvpStringType> chargingRuleReports;  DiaAppAVPUtils::DiameterAppAVPUtils::ParseChargingRuleReport(  parser,  TinyDiameter:: s\_ChargingRuleReport,  chargingRuleReports);    for(auto& instances: chargingRuleReports)  {  /\*  Handle the errors on AvpStringType validation failures.  Else store the AVP values in S9SubsessionSessionStore.  \*/  }  } |

#### RxServer Subsystem

Rx Server Subsystem workflow will be slight change in its existing workflow’s steps.

In RxCreateSession step of RxSessionEstablishment Workflow, it must check whether Rx AAR-I message IP already has a S9 subsession. If not RxCreateSession step will fail and AAA will be encoded with Result-Code IP-CAN\_SESSION\_NOT\_AVAILABLE.

## System CLI

devices [435]

s9Server [34420]

1 [1]

config [1]

stats [2]

peer [100]

Figure : S9-Server PDB structure overview

devices [435]

s9Client [35420]

1 [1]

config [1]

stats [2]

peer [100]

Figure : S9-Client PDB structure overview

config [1]

stackToAppQueueSize [1]

diameterRegistration [2]

supportedRelease [3]

subscriptionIdType [4]

chargingRuleEncodingMode [5]

Figure : S9-Server config PDB structure

config [1]

stackToAppQueueSize [1]

diameterRegistration [2]

supportedRelease [3]

Figure : S9-Client config PDB structure

|  |
| --- |
| **show service s9-server**  To display the S9 Server KPIs averaged over last minute.  SDE> show service s9-server  MESSAGING  =========  Message                Requests Answers Errors  ---------------------- -------- ------- ------ CreditControlInitial          0       0      0  CreditControlUpdate           0       0      0  CreditControlTerminate        0       0      0  ReAuthorization               0       0      0  \* Requests, Answers and Errors are per second, averaged over the last minute.  **Pdb-Path:**  devices.s9Server.1.stats.messageRateCounters.session [Figure 54]  devices.s9Server.1.stats.errorRateCounters.session [Figure 56] |
| **show service s9-client**  **Pdb-Path:**  devices.s9Client.1.stats.messageRateCounters.session [Figure 54]  devices.s9Server.1.stats.errorRateCounters.session [Figure 56] |

stats [2]

messageRateCounters [10]

creditControlInitial [1]

creditControlUpdate [2]

creditControlTerminate [3]

reAuthorization        [4]

session [10]

Figure : S9-Server & S9-Client overall messageRateCounters PDB structure

|  |
| --- |
| **show service s9-server stats**  MESSAGING  =========  Message                Requests Answers Errors  ---------------------- -------- ------- ------  CreditControlInitial      1,075   1,075      0  CreditControlUpdate          20      20      0  CreditControlTerminate    1,074   1,074      0  ReAuthorization           1,075   1,075      0  SESSION MANAGEMENT  ==================  Operation Enforcements Decisions Errors  --------- ------------ --------- ------  Establish        1,055     1,055      0  Modify              40        20     20  Terminate        1,075     1,075      0  **Pdb-Path:**  devices.s9Server.1.stats.messageCounters.session [Figure 55]  devices.s9Server.1.stats.errorCounters [Figure 57]  devices.s9Server.1.stats.messageCounters.subsession [Figure 55] |
| **show service s9-client stats**  **Pdb-Path:**  devices.s9Client.1.stats.messageCounters.session [Figure 55]  devices.s9Client.1.stats.errorCounters [Figure 57]  devices.s9Client.1.stats.messageCounters.subsession [Figure 55] |

stats [2]

messageCounters [20]

creditControlInitial [1]

creditControlUpdate [2]

creditControlTerminate [3]

reAuthorization        [4]

establishSubsessionEnforcement [1]

modifySubsessionEnforcement [2]

terminateSubsessionEnforcement [3]

establishSubsessionDecision [4]

modifySubsessionDecision [5]

terminateSubsessionDecision [6]

establishSubsessionError [7]

modifySubsessionError [8]

terminateSubsessionError [9]

session [10]

subsession [20]

Figure : S9-Server & S9-Client overall messageCounters PDB structure

stats [2]

errorRateCounters [30]

disconnectedPeer  [1]

invalidAVP [2]

missingAVP [3]

duplicateCreate [4]

invalidSessionID [5]

ipAddressBindingFailed [6]

ipAddressUnbindingFailed [7]

maximumSessionsExceeded [8]

sessionCreateFailed [9]

dataSessionNotFound [10]

discardedAttributeUpdate [11]

discardedRuleUpdate [12]

ruleConfigError    [13]

Figure : S9-Server & S9-Client overall errorRateCounters PDB structure

Note: At present for the feature havn’t come up with the list of debug pdb counters going to support. Going further in the feature will add the required details.

|  |
| --- |
| **show service s9-server errors**  To display the overall S9 Server errors from the start of subsystem.  SDE> show service s9-server errors  MESSAGING  =========  ErrorCode Errors  --------------------- ------  DisconnectedPeer 0  InvalidAVP 0  MissingAVP 0  SESSION AWARENESS  =================  ErrorCode Errors  ----------------------- ------  DuplicateCreate 0  InvalidSessionID 0  IpAddressBindingFailed 0  IpAddressUnbindingFailed 0  MaximumSessionsExceeded 0  SessionCreateFailed 0  DataSessionNotFound 0  ENFORCEMENT  ===========  ErrorCode Errors  ------------------------ ------  DiscardedAttributeUpdate 0  DiscardedRuleUpdate 0  RuleConfigError 0  **Pdb-Path:**  devices.s9Server.1.stats.errorCounters [Figure 57] |
| **show service s9-client errors**  **Pdb-Path:**  devices.s9Client.1.stats.errorCounters [Figure 57] |

|  |
| --- |
| **show service s9-server errors history**  LAST 50 ERRORS  ===============  EventTime ErrorCode Description  ----------------------- ------------------------- -----------------------------------------------  2016-11-02 05:20:29 EDT InvalidSubscriptionID Unsupported value in Subscription-Id-Type (450) AVP  2016-11-02 05:13:49 EDT InvalidSubscriptionID Unsupported value in Subscription-Id-Type (450) AVP  2016-11-02 05:07:09 EDT InvalidSubscriptionID Subscription-Id-Type (450) has invalid length  LAST 10 UNIQUE ERROR CODES  ==========================  EventTime ErrorCode Description  ----------------------- ------------------------- -----------------------------------------------  2016-11-02 05:20:29 EDT InvalidSubscriptionID Unsupported value in Subscription-Id-Type (450) AVP  2016-11-02 05:07:09 EDT InvalidSubscriptionID Subscription-Id-Type (450) has invalid length  **Pdb-Path:**  devices.s9Server.1.stats.errorRateCounters.errorHistoryTable.errorHistoryTableEntry  devices.s9Server.1.stats.errorRateCounters.uniqueErrorHistoryTable.uniqueErrorHistoryTableEntry |
| **show service s9-client errors history**  **Pdb-Path:**  devices.s9Client.1.stats.errorRateCounters.errorHistoryTable.errorHistoryTableEntry [Figure 57]  devices.s9Client.1.stats.errorRateCounters.uniqueErrorHistoryTable.uniqueErrorHistoryTableEntry [Figure 57] |

stats [2]

errorCounters [40]

disconnectedPeer  [1]

invalidAVP [2]

missingAVP [3]

duplicateCreate [4]

invalidSessionID [5]

ipAddressBindingFailed [6]

ipAddressUnbindingFailed [7]

maximumSessionsExceeded [8]

sessionCreateFailed [9]

dataSessionNotFound [10]

discardedAttributeUpdate [11]

discardedRuleUpdate [12]

ruleConfigError    [13]

errorHistoryTable [100]

uniqueErrorHistoryTable [200]

errorHistoryTableEntry [1]

uniqueErrorHistoryTableEntry [1]

index [1]

eventTime [2]

errorCode [3]

description     [4]

index [1]

eventTime [2]

errorCode [3]

description     [4]

Figure : S9-Server & S9-Client overall errorCounters PDB structure

|  |
| --- |
| **show service s9-server peers**  Peer Requests Answers Errors  ------------------ -------- ------- ------  vpcrf2.sandvine.com 244 244 0  vpcrf3.sandvine.com 324 320 4  **Pdb-Path:**  devices.s9Server.1.stats.peer.messageCounters.session [Figure 58]  devices.s9Server.1.stats.peer.errorCounters.session [Figure 59] |
| **show service s9-client peers**  **Pdb-Path:**  devices.s9Client.1.stats.peer.messageCounters.session [Figure 58]  devices.s9Client.1.stats.peer.errorCounters.session [Figure 59] |

|  |
| --- |
| **show service s9-server peer vpcrf.sandvine.com**  MESSAGING  =========  Message Requests Answers Errors LatencyAverage(us) RoundTripTime(ms)  ---------------------- -------- ------- ------ ------------------ -----------------  CreditControlInitial 0 0 0 523 N/A  CreditControlUpdate 0 0 0 1,717 N/A  CreditControlTerminate 0 0 0 325 N/A  ReAuthorization 0 0 0 196 3    SESSION MANAGEMENT  ==================  Operation Enforcements Decisions Errors  ------------- ------------ --------- ------  Establish 1,055 1,055 0  Modify 40 20 20  Terminate 1,075 1,075 0  **Pdb-Path:**  devices.s9Server.1.stats.peer.messageCounters.session [Figure 58]  devices.s9Server.1.stats.peer.errorCounters.session [Figure 59]  devices.s9Server.1.stats.peer.messageCounters.subsession [Figure 58] |
| **show service s9-client peer hpcrf.sandvine.com**  **Pdb-Path:**  devices.s9Client.1.stats.peer.messageCounters.session [Figure 58]  devices.s9Client.1.stats.peer.errorCounters.session [Figure 59]  devices.s9Client.1.stats.peer.messageCounters.subsession [Figure 58] |

peer [100]

messageCounters [10]

creditControlInitial [1]

creditControlUpdate [2]

creditControlTerminate [3]

reAuthorization        [4]

latencyAverage [5]

roundTripTime [6]

session [10]

establishSubsessionEnforcement [1]

modifySubsessionEnforcement [2]

terminateSubsessionEnforcement [3]

establishSubsessionDecision [4]

modifySubsessionDecision [5]

terminateSubsessionDecision [6]

establishSubsessionError [7]

modifySubsessionError [8]

terminateSubsessionError [9]

subsession [20]

Figure : S9-Server & S9-Client peer messageCounters PDB structure

|  |
| --- |
| **show service s9-server peer vpcrf.sandvine.com errors**  MESSAGING  =========  ErrorCode Errors  --------------------- ------  DisconnectedPeer 0  InvalidAVP 0  MissingAVP 0  SESSION AWARENESS  =================  ErrorCode Errors  ----------------------- ------  DuplicateCreate 0  InvalidSessionID 0  IpAddressBindingFailed 0  IpAddressUnbindingFailed 0  MaximumSessionsExceeded 0  SessionCreateFailed 0  DataSessionNotFound 0  ENFORCEMENT  ===========  ErrorCode Errors  ------------------------ ------  DiscardedAttributeUpdate 0  DiscardedRuleUpdate 0  RuleConfigError 0  **Pdb-Path:**  devices.s9Server.1.stats.peer.errorCounters [Figure 59] |
| **show service s9-client peer hpcrf.sandvine.com errors**  **Pdb-Path:**  devices.s9Client.1.stats.peer.errorCounters [Figure 59] |

peer [100]

errorCounters [30]

disconnectedPeer  [1]

invalidAVP [2]

missingAVP [3]

duplicateCreate [4]

invalidSessionID [5]

ipAddressBindingFailed [6]

ipAddressUnbindingFailed [7]

maximumSessionsExceeded [8]

sessionCreateFailed [9]

dataSessionNotFound [10]

discardedAttributeUpdate [11]

discardedRuleUpdate [12]

ruleConfigError    [13]

Figure : S9-Server & S9-Client peer errorCounters PDB structure

## System Alarms

Below steps need to be followed to implement S9 "Unenforced Roaming Rules" alarm

1. Assign an alarm model number.

To do so, in "constants\_dev" stream, add a new alarm model macro in "/vobs/fw-constants/constants/alarmIndex.h" with the next available alarm model index under SDE section.

For example add the below line in the "alarmIndex.h file". The number 194 might change based on the availability on the day of actual implementation.

|  |
| --- |
| #define ALARM\_MODEL\_INDEX\_UNENFORCED\_ROAMING\_RULES 194 |

1. Assign SNMP Notification IDs for the S9 "Unenforced Roaming Rules" alarm

To do so, in constants\_dev stream, add a unique trap ID in "/vobs/fw-constants/constants/notifications.h" with the next available trap Id.

For example add the below line in the "notifications.h" file. The macro values might change based on the availability on the day of actual implementation.

|  |
| --- |
| #define SNMP\_TRAP\_UNENFORCED\_ROAMING\_RULES\_PREFIX 46  #define SNMP\_TRAP\_UNENFORCED\_ROAMING\_RULES\_MAJOR 1  #define SNMP\_TRAP\_UNENFORCED\_ROAMING\_RULES\_CLEAR 2 |

1. Define Alarm Index Configuration

Create 000.alarmIndex.s9\_subsystem.conf.svcpp alarm index configuration file in S9 folder (/vobs/fw/support/S9). This entry will reference the alarm model index that is defined previously in Step1.

|  |
| --- |
| alarmIndex UnenforcedRoamingRulesIndex ALARM\_MODEL\_INDEX\_UNENFORCED\_ROAMING\_RULES |

1. Implement PDB variables defined in section ["System CLI"](#_System_CLI)
2. Create subtraps\_s9.mib.svcpp in S9 folder (/vobs/fw/support/S9) and define SNMP notifications as below.

|  |
| --- |
| #include <constants/notifications.h>  svRoamingPrefix OBJECT IDENTIFIER ::= { svNotifications SNMP\_TRAP\_UNENFORCED\_ROAMING\_RULES\_PREFIX }  svRoamingNotifications OBJECT IDENTIFIER ::= { svRoamingPrefix 1 }  svUnenforcedRoamingRuleNotification NOTIFICATION-TYPE  OBJECTS {  svClusterConfigName,  sysName,  svSeverity,  svRoamingStatsUndefinedSessions,  svRoamingStatsInvalidRequests,  svRoamingStatsUndefinedRules,  svRoamingStatsInvalidAnswers  }  STATUS current  DESCRIPTION  "This notification is sent if total erroneous messages are 5% or more of the total messages received.  This percentage calculation is done every 15 minutes. Following can be the observed errors for an s9 message -  1. svRoamingUndefinedSessions – This error counter is incremented when S9 server cannot find a corresponding session either for CCR-U, CCR-T, CCA-U or CCA-T  2. svRoamingInvalidRequests – This error counter is incremented when S9 server receives an invalid request like the Mandatory AVP is missing or the AVP has invalid field values  3. svRoamingUndefinedRules – This error counter is incremented when S9 server rules are rejected by V-PCEF.  4. svRoamingInvalidAnswers - This error counter is incremented when S9 client receives an invalid answer like the Mandatory AVP is missing or the AVP has invalid field values"  ::= { svRoamingNotifications SNMA\_TRAP\_UNENFORCED\_ROAMING\_RULES\_MAJOR}  svUnenforcedRoamingRuleClearNotification NOTIFICATION-TYPE  OBJECTS {  svClusterConfigName,  sysName,  svSeverity,  svRoamingStatsUndefinedSessions,  svRoamingStatsInvalidRequests,  svRoamingStatsUndefinedRules,  svRoamingStatsInvalidAnswers  }  STATUS current  DESCRIPTION  "This notification is sent as a clear notification if total erroneous messages are less than 5% of total messages received.  This percentage calculation is done every 15 minutes. Following can be the observed errors for an s9 message -  1. svRoamingUndefinedSessions – This error counter is incremented when S9 server cannot find a corresponding session either for CCR-U, CCR-T, CCA-U or CCA-T  2. svRoamingInvalidRequests – This error counter is incremented when S9 server receives an invalid request like the Mandatory AVP is missing or the AVP has invalid field values  3. svRoamingUndefinedRules – This error counter is incremented when S9 server rules are rejected by V-PCEF.  4. svRoamingInvalidAnswers - This error counter is incremented when S9 client receives an invalid answer like the Mandatory AVP is missing or the AVP has invalid field values"  ::= { svRoamingNotifications SNMA\_TRAP\_UNENFORCED\_ROAMING\_RULES\_CLEAR} |

1. Create a BUILD.yaml file in S9 folder (/vobs/fw/support/S9) and add source files as below.

|  |
| --- |
| - pdb2mib\_gen:  name: s9.pdb.my  srcs: [s9.pdb]  - svcpp\_gen:  name: subtraps\_s9.my  srcs: [subtraps\_s9.mib.svcpp] |

1. Define the Alarm Model Configuration file "s9.alarm.conf" in S9 folder (/vobs/fw/support/S9)

|  |
| --- |
| alarm UnenforcedRoamingRulesIndex major "Unenforced Roaming Rules" svUnenforcedRoamingRuleNotification  alarm UnenforcedRoamingRulesIndex clear "Unenforced Roaming Rules Cleared" svUnenforcedRoamingRuleClearNotification |

1. Define the Alarm Trigger and Event Configuration file "events.s9.conf" in S9 folder (/vobs/fw/support/S9)

|  |
| --- |
| NotificationEvent UnenforcedRoamingRuleEvent svUnenforcedRoamingRuleNotification -n -i svSeverity.0 4  NotificationEvent UnenforcedRoamingRuleClearEvent svUnenforcedRoamingRuleClearNotification -n -i svSeverity.0 1  Expression UnenforcedRoamingRuleCountExp -t uint32 -c "Unenforced Roaming Rule count" -r 900 "(100 \* ($1+$2+$3+$4) >= 5\*($5+$6+$7+$8+$9+$10+$11+$12))"  ExpressionObject UnenforcedRoamingRuleCountExp.1 -D svRoamingStatsUndefinedSessions  ExpressionObject UnenforcedRoamingRuleCountExp.2 -D svRoamingStatsInvalidRequests  ExpressionObject UnenforcedRoamingRuleCountExp.3 -D svRoamingStatsUndefinedRules  ExpressionObject UnenforcedRoamingRuleCountExp.4 -D svRoamingStatsInvalidAnswers  ExpressionObject UnenforcedRoamingRuleCountExp.5 -D svRoamingStatsCCRI  ExpressionObject UnenforcedRoamingRuleCountExp.6 -D svRoamingStatsCCRU  ExpressionObject UnenforcedRoamingRuleCountExp.7 -D svRoamingStatsCCRT  ExpressionObject UnenforcedRoamingRuleCountExp.8 -D svRoamingStatsRAA  ExpressionObject UnenforcedRoamingRuleCountExp.9 -D svRoamingStatsCCAI  ExpressionObject UnenforcedRoamingRuleCountExp.10 -D svRoamingStatsCCAU  ExpressionObject UnenforcedRoamingRuleCountExp.11 -D svRoamingStatsCCAT  ExpressionObject UnenforcedRoamingRuleCountExp.12 -D svRoamingStatsRAR  Monitor UnenforcedRoamingRuleTgr "Unenforced roaming rule trigger" -e UnenforcedRoamingRuleEvent UnenforcedRoamingRuleCountExp == 1  Monitor UnenforcedRoamingRuleClearTgr "Unenforced roaming rule clear trigger" -e UnenforcedRoamingRuleClearEvent UnenforcedRoamingRuleCountExp == 0 |

1. To generate the S9 MIB file, create the file “/vobs/fw/mibs/sandvine-s9.yaml”

The sample contents of this file are shown below.

|  |
| --- |
| ---  name: svS9Info  oid: 39000 <To be decided at the time of implementation>  parent: svDevices  description: >  Structure of management information for the Sandvine S9 module.  subsections:  - name: svS9  oid: 1  subsections:  - //support/s9:s9.pdb.my  - //support/s9: subtraps\_s9.my |

## System Configuration and Installation

### Packaging

**SDE Platform Update (svSDEPlugins.xml)**

The SDE platform configuration file **svSDEPlugins.xml** located in /usr/local/sandvine/etc/ MUST be updated with the following new lines:

|  |
| --- |
| <plugin>  <Name>S9</Name>  <Infras>Platform8XX,svsde</Infras>  <Enabled>true</Enabled>  <environment>S9</environment>  </plugin> |

**Note** – S9 subsystem is enabled by default. The policy file MUST have S9 specific custom policy for proper functioning of S9.

**Packaging of S9 specific files**

The following S9 specific files MUST be modified to accommodate S9:

1. svsde.spec
2. scdpd.spec
3. sde\_policy\_common.spec
4. s9.spec (new)

### S9 Policy Packages

S9 Server and Client these are two different subsystems. There will be two different policy packages for S9 client and server

Policy.conf:

Package S9\_client \

S9\_client\_nds\_report "true" \

debug\_level "S9\_CLIENT\_DEBUG\_LEVEL\_INFO"

Package S9\_server \

S9\_server\_nds\_report "true" \

debug\_level "S9\_SERVER\_DEBUG\_LEVEL\_INFO"

Policy packaging for SDE and SPB (NDS Reports).

Note: Make framework for policy stream is still need to be implemented. Once that is done the below infra will change accordingly.

|  |
| --- |
| S9 Client policy packaging for SDE and SPB:  Path: /vobs/fw-policy/products/svs9client   1. The following sub directories and files will be created at this path:   Makefile  Makefile.FREEBSD  Makefile.LINUX  linux  nds  sde   1. On running “make release” command at this path, both linux policy RPM and NDS RPMs will be created.   svS9Client-7.60-9002.el7.x86\_64.rpm  svS9Client\_nds-7.60-9002.el7.x86\_64.sym.rpm   1. NDS packages and help page related html files will be located at:   /vobs/fw-policy/products/svs9client/nds/sandvine/reports/   1. On installing policy package RPM on SDE following files should get installed:   /usr/local/sandvine/policypkg/s9\_client  /usr/local/sandvine/etc/s9\_client/policypkg  Note: For building RPMs can refer svsyclient model, as this was discussed and agreed by SMEs during feature development. |
| S9 Server:  Path: /vobs/fw-policy/products/svs9server   1. The following sub directories and files will be created at this path:   Makefile  Makefile.FREEBSD  Makefile.LINUX  linux  nds  sde   1. On running “make release” command at this path, both linux policy RPM and NDS RPMs will be created.   svS9Server-7.60-9002.el7.x86\_64.rpm  svS9Server\_nds-7.60-9002.el7.x86\_64.sym.rpm   1. NDS packages and help page related html files will be located at:   /vobs/fw-policy/products/svs9server/nds/sandvine/reports/   1. On installing policy package RPM on SDE following files should get installed:   /usr/local/sandvine/policypkg/s9\_server  /usr/local/sandvine/etc/s9\_server/policypkg  Note: For building RPMs can refer svsyclient model, as this was discussed and agreed by SMEs during feature development. |

### Configuration

**Diameter S9 Dictionary**

The S9 dictionary 3gpp\_29\_215.xml.sample with all 3GPP supported S9 messages & AVPs are defined.

The link for the S9 diameter dictionary is as follows http://wtl-wiki-1/ccase/Documents/Engineering/Projects/Logs/sde\_8\_10/Features/DiameterS9/Dictionary/Diameter/3gpp\_29\_215.xml.sample

**Diameter Configuration**

The Diameter dictionary file d**iam\_config.xml** located in /usr/local/sandvine/etc MUST be modified to include S9 dictionary as follows:

|  |
| --- |
| <diameter>  <general>  <product>SV DIAMETER</product>  <vendor-id> 11610</vendor-id>  <dictionaries>  <dictionary name="Gx" file="/usr/local/sandvine/etc/diameter/dictionary/dictionaryGx.xml"/>  <dictionary name="Rx" file="/usr/local/sandvine/etc/diameter/dictionary/dictionaryRx.xml"/>  <dictionary name="S9" file="/usr/local/sandvine/etc/diameter/dictionary/dictionaryS9.xml"/>  </dictionaries>  </general>  </diameter> |

**Diameter S9 Peer**

The **diam\_peer\_config.xml** in /usr/local/sandvine/etc MUST be configured as follows:

|  |
| --- |
| <diameter-peer>  <peers>  <peer>  <identity>remote.sandvine.com</identity>  <hostname>remote.sandvine.com</hostname>  <realm>sandvine.com</realm>  <dictionaries>  <dictionary>Gx</dictionary>  <dictionary>Rx</dictionary>  <dictionary>S9</dictionary>  </dictionaries>  <security>  <ca-file>CApath</ca-file>  <cert-file>CERTpath</cert-file>  <key-file>KEYpath</key-file>  </security>  <reachable-realms>  <realm>sandvine.com</realm>  </reachable-realms>  </peer>  </peers>  <identities>  <local-identity>  <identity>local.sandvine.com</identity>  <display-name>local.sandvine.com</display-name>  <realm>sandvine.com</realm>  <dictionaries>  <dictionary>Gx</dictionary>  <dictionary>Rx</dictionary>  <dictionary>S9</dictionary>  </dictionaries>  <security>  <ca-file>CApath</ca-file>  <cert-file>CERTpath</cert-file>  <key-file>KEYpath</key-file>  </security>  <server-config>  <ip>2.0.0.1</ip>  <port>3868</port>  <watchdog-timeout>10</watchdog-timeout>  <cer-timeout>5</cer-timeout>  <transport-protocol>tcp</transport-protocol>  <accept>  <remote-peer>remote.sandvine.com</remote-peer>  </accept>  </server-config>  <client-config>  <ip>2.0.0.1</ip> -->  <watchdog-timeout>10</watchdog-timeout>  <cea-timeout>5</cea-timeout>  <connection-timeout>1</connection-timeout>  <transport-protocol>tcp</transport-protocol>  <connect>  <remote-peer>remote.sandvine.com</remote-peer>  </connect>  </client-config>  </local-identity>  </identities>  </diameter-peer> |

**Session Dictionary**

The session dictionary S9.xml MUST be located at /usr/local/sandvine/etc/session/dictionary

Note– The basic dictionary structure is captured in initial version of document. As the design progresses for other sections, the respective owner shall update this section with specific attributes.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Session>  <Name>S9</Name>  <Keys>  <Key>  <Name>Id</Name>  <Type>string</Type>  </Key>  </Keys>  <Attributes>  <Attribute>  <Name>Origin\_Host</Name>  <Type>string</Type>  </Attribute>  <Attribute>  <Name>Origin\_Realm</Name>  <Type>string</Type>  </Attribute>  <Attribute>  <Name>Peer\_Handle</Name>  <Type>integer</Type>  </Attribute>  <Attribute>  <Name>Origin\_State\_Id</Name>  <Type>integer</Type>  </Attribute>  <Attribute>  <Name>Release\_Version</Name>  <Type>integer</Type>  </Attribute>  </Attributes>  </Session> |

**Subscriber Dictionary**

The subscriber dictionary subscriber.xml MUST be located at /usr/local/sandvine/etc/subscriber/dictionary

## Session Model

This section deals with the association between subsessions and their parent sessions.

The intention is to introduce subsession and parent session association in a generic manner so that future subsystems can leverage this implementation.

However policy expressions and examples in this section would be S9 specific. How to make it a generic implementation would be discussed in the LLD.

### Design Considerations

1. The parent session should be accessible from within the Subsession context.
2. When a Subsession is created, ended, time-out – relevant policy events must be raised.
3. The Subsession dictionary of H-PCRF would be different from that of V-PCRF.
4. The subsessions must be represented adequately in PCRF CLIs.
   1. At the same time, subsession’s CLI representation must not add unnecessary information to already existing CLIs – point is with the introduction of subsessions, all old CLIs should continue to make sense without duplication of information.

### Design Options.

#### Independent Subsession Model – Created Subsession ID

STATUS: Accepted.

Under this Session Model, a subsession would be just another control Session which is created independently and from a session subsystem perspective has an independent lifecylce. The triggers to create and destroy these subsessions would have to be managed by the application, ie, S9 in this case.

**SDE Created SubSession ID:**

S9 policy package will provide a policy function which will create a deterministic combination out of the S9 session ID, a pre-configured unique S9 subsession string and the subsession Identifier as generated by the VPCRF and then create the subsession with keyed with this uniquely created session ID.

* + 1. This session ID would naturally be a string.

|  |
| --- |
| Implicit Access  Session.AttributeView.*S9Home*.<Attribute-Name>  Session.AttributeView.*SubsessionS9Home*. <Attribute-Name>  Explicit Access  Session[*“session\_id\_s9”*].AttributeView.S9Home. <Attribute-Name>  Session[“session\_id\_s9\_sv\_1”].AttributeView.SubsessionS9Home. <Attribute-Name>  + session\_id\_s9 : is the S9 session id as generated at the VPCRF (and recieved by the HPCRF)  + \_sv\_ : is the unique S9 subsession string as configured at the VPCRF and HPCRF.  + 1 : is the subsession’s identifier as as generated at the VPCRF (and recieved by the HPCRF)  \*Note: Visited session model would be the same too, only the dictionary name would contain ‘visited’ instead of ‘home’ |

**Raising Events**

Since, in this model, a subsession is exactly a control session, all events raised for a Control Session by the session subsystem would be raised for a subsession as well.

The S9 parent session would be treated akin to a Linked Control Session as is the case for Sy. Therefore all events raised for a Linked Control Session would be raised for S9 session as well.

**Features:**

1. Only one S9 session would be created. This S9 session would also be referred to as the S9 parent session in this document.
2. Any number of subsession for the S9 parent sessions can be created.
3. Policy function would be provided to create the session ID.
4. Another policy function would be provided to de-couple the subsession ID string and return the subsession ID integer.

**Pros**

1. Easiest to implement. There’s probably nothing to implement.
2. Least complexity and therefore bug free.
3. Leverages the existing Session Manager infrastructure and adapts it to the intention of creating subsessions for every parent session.

**Cons**

1. Cannot iterate in policy over subsessions of a S9 control session. This could be achieved by storing the subsession ids in the S9 control session.
2. With this approach, CLIs at Sandvine VPCRF and HPCRF would sh doesnt really exist ow a control session (subsession) but this control session in the network.
   * + 1. In any case, these subsession have to displayed in some CLI.
       2. We could might as well leverage the existing infrastructure and display it as a part of that (with some enhancements to make it look even better)

Note: During feature development will add the required attributes under defined xml below.

|  |
| --- |
| **S9Home.default.xml** |
| <Session>  <Name>**S9Home**</Name>  <Keys>  <Key>  <Name>Id</Name>  <Type>string</Type>  </Key>  </Keys>  <Attributes>  <Attribute>  <Name>Origin\_Host</Name>  <Type>string</Type>  </Attribute>  <Attribute>  <Name>Origin\_Realm</Name>  <Type>string</Type>  </Attribute>  . . . . .  </Attributes>  </Session> |

|  |
| --- |
| **SubsessionS9Home.default.xml** |
| <Session>  <Name>SubsessionS9Home</Name>  <Keys>  <Key>  <Name>Id</Name>  **<Type>String</Type>**  </Key>  </Keys>  <Attributes>  <Attribute>  <Name>Origin\_Host</Name>  <Type>string</Type>  </Attribute>  <Attribute>  <Name>Origin\_Realm</Name>  <Type>string</Type>  </Attribute>  . . . . .  <Attributes>  **</Subsession>**  </Session> |

##### New configuration CLIs

|  |
| --- |
| 1. set config service S9Home subsession-unique-identifier 2. set config service S9Visited subsession-unique-identifier  * a minimum of 1 character and a maximum of 6 characters |

##### Session CLI Modifications.

This CLI will display session details for SubSession Framed-IP-Address not for the parent Session.

|  |
| --- |
| SDE> show service session-management ip <SubSession Framed-IP-Address>  SESSION  =======  Subscriber: 12345678900250  SessionID : 18446742587943869687 // <SubSession ID>  IP ADDRESSES  ============  IpAddress Site Assigned  ------------ ---- -----------------------  9.0.0.250/32 0 2017-01-30 14:11:21 IST  CONTROL SESSIONS  ================  SessionID Protocol EnforcementGroup  ------------- -------- ----------------  S9Session;250 S9 N/A  SESSION ATTRIBUTES  ================== |

|  |
| --- |
| 1. SDE> show service session-management   SESSIONS  ========  Protocol        Sessions Logins Logouts Errors  --------        -------- ------ ------- ------  Data                 766     80      70      0  Gx                   636     60      60      0  PCMM                   0     10       0      0  Rx                     0     10       0      0  SPB                  763     71      71      0  Sy                   632     60      60      0  S9Home               30     40      10      0  S9Visited            10      20      10      0  SubsessionS9Home     60     80      20      0  SubsessionS9Visited  20      40      20      0  \* Logins, Logouts and Errors are per second, averaged over the last minute |

|  |
| --- |
| 1. SDE> show service session-management stats   SESSIONS  ========  Protocol        Sessions Logins Logouts Errors  --------        -------- ------ ------- ------  Data                 766     80      70      0  Gx                   636     60      60      0  PCMM                   0     10       0      0  Rx                     0     10       0      0  SPB                  763     71      71      0  Sy                   632     60      60      0  S9Home               300     400      100      0  S9Visited            100      200      100      0  SubsessionS9Home     600     800      200      0  SubsessionS9Visited  200      400      200      0 |

|  |
| --- |
| 1. SDE> show service session-management session-id s9session:891\_Bell   SESSION  =======  Subscriber: Bell:SipForking:Sub  Protocol : S9  IP ADDRESSES  ============  List out all Ips assiciated with this S9 session (Phase 2)  SESSION ATTRIBUTES  ==================  Name Value Type  ------------------- ------------------- -----------  CC\_Request\_Number 1 [integer]  EnforcementGroup Sandvine [string]  Framed\_IPv4\_Address 8.0.0.1 [ipaddress]  IP\_CAN\_Type 5 [integer]  Negotiated\_Features 0 [integer]  Origin\_Host 2.0.0.2 [string]  Origin\_Realm sandvine.com [string]  Origin\_State\_Id 0 [integer]  Peer\_Handle 3 [integer]  RAT\_Type 1 [integer]  Stale false [bool]  Subscriber\_Name[1] Bell:SipForking:Sub [string] |

|  |
| --- |
| 1. SDE> show service session-management session-id s9session:891\_Bell **subsession** 1   SESSION  =======  Subscriber: Bell:SipForking:Sub  Protocol : S9  IP ADDRESSES  ============  IpAddress Site Assigned  ---------- ---- -----------------------  8.0.0.1/32 0 2017-01-06 16:15:46 IST  SESSION ATTRIBUTES  ==================  Name Value Type  ------------------- ------------------- -----------  CC\_Request\_Number 1 [integer]  EnforcementGroup Sandvine [string]  Framed\_IPv4\_Address 8.0.0.1 [ipaddress]  IP\_CAN\_Type 5 [integer]  Negotiated\_Features 0 [integer]  Origin\_Host 2.0.0.2 [string]  Origin\_Realm sandvine.com [string]  Origin\_State\_Id 0 [integer]  Peer\_Handle 3 [integer]  RAT\_Type 1 [integer]  Stale false [bool]  Subscriber\_Name[1] Bell:SipForking:Sub [string]  \*Note: The attributes printed above would be the attributes that are part of the subsession dictionary. This would not display any attribute that is not a part of the subsession dictionary. So, S9 parent attributes or those of any other subsession will NOT be printed here. |

1. “show service service-designer session control id <s9-parent-id> subsession <subsession\_id>”

The above CLI would display the rules installed on the subsession.

1. “show service service-designer session control id <s9-parent-id>” (Phase 2)

This CLI will provide a consolidated list of all rules installed on every subsession of this S9 session.

##### Policy Package.

The S9 policy package should provide functions to create a unique S9 subsession ID. It must also provide a function to return the subsession id (integer) when provided with the subsession’s control id (string).

#### Independent Subsession Model – Compound Key.

STATUS: Rejected.

The subsession would be keyed by a compound key –

* + 1. this key would be the S9 Session ID (string) as well as the subsession ID (integer).
    2. These two keys would need to provided in a comma separated manner within policy accessors.
    3. Both the keys are provided by the VPCRF to the HPCRF.

|  |
| --- |
| Implicit Access  Session.AttributeView.S9Home.’Origin-Host’  Session[*“session\_id\_s9”*]. AttributeView.SubsessionS9Home. ’Origin-Host’  Explicit Access  Session[*“session\_id\_s9”*].AttributeView.S9Home.’Origin-Host’  Session[“session\_id\_s9”, 1].AttributeView.SubsessionS9Home.’Origin-Host’  \*Note: Visited session model would be the same too, only the dictionary name would contain ‘visited’ instead of ‘home’ |

**Raising Events**

The session subsystem’s events mechanism would need to be updated. Since the subsession too would be a control session but, this control session would be keyed by two keys.

This is a more convoluted implementation. Session Manager stores control sessions in a hash map which is keyed by a string identifier. Now this hash\_map would need to be enhanced to store a key pair. Since hash maps dont inherently dont support multiple key types, we’d need to create a new type of data structure.

**Pros:**

1. I dont really see any benefit in this approach.
2. It is somewhat representative of the subsession model but it’s certainly not the best representation that we could design.

**Cons:**

1. The subsessions would either be stored in a separate hash\_map or the existing control session hash\_map would need to be enhanced into a key-pair data structure. This is going to need a lot more design and implementation effort.
2. Internally, SDE has to store a key combination that represents the S9 session ID and the subsession id. We could might as well go with option 1.
3. Persisting any modification to one subsession could potentially lead to persisting all subsessions. This would have an adverse impact on performance.
4. How does a policy author know which subsession to access?
5. CLIs, events, performance will all need tweaking in order to support this model.

#### True Subsession Model

STATUS: Rejected.

This model would introduce a new tag “Subsession” within the attribute dictionary.

This tag would mean that all attributes defined within it belong to a Subsession and this Subsession belongs to the root node below which it would have been defined.

The Subsession in this scheme would raise events independently, just like a control session does. However, subsessions would be created by explicit events in the occurring after (or along with) the creation of the parent session.

Figuratively, this is how the S9 dictionary can be represented:



Below is what you could expect part of the S9 Session dictionary to look like:

|  |
| --- |
| **S9Home.default.xml** |
| <Session>  <Name>S9Home</Name>  <Keys>  <Key>  <Name>Id</Name>  <Type>string</Type>  </Key>  </Keys>  <Attributes>  <Attribute>  <Name>Origin\_Host</Name>  <Type>string</Type>  </Attribute>  . . . . .  </Attributes>  </Session> |

|  |
| --- |
| **S9Home.Subsession.xml** |
| <Session>  <Name>S9Home</Name>  <Keys>  <Key>  <Name>Id</Name>  <Type>string</Type>  </Key>  </Keys>  **<Subsession>**  **<Name>Subsession</Name>**  **<Keys>**  **<Key>**  **<Name>Id</Name>**  **<Type>integer</Type>**  **</Key>**  **</Keys>**  <Attributes>  <Attribute>  <Name>Origin\_Host</Name>  <Type>string</Type>  </Attribute>  . . . . .  <Attributes>  **</Subsession>**  </Session> |

S9Visited dictionary and the dictionary for its Subsession would be similar to what has been represented in the tables above.

|  |
| --- |
| Implicit Access  S9Home  Session.AttributeView.S9Home.Subsession.’Origin-Host’  Session.AttributeView.S9Home.Subsession.IsTimeout  Session.AttributeView..S9Home.Subsession.ID <- - Integer  Etc.  Session. AttributeView.S9Home.ID < - - String  Session. AttributeView.S9Home. ’Origin-Host’  Session. AttributeView.S9Home. IsTimeout  Etc.  S9Visited  Session. AttributeView.S9Visited.Subsession. ’Origin-Host’  Session. AttributeView..Subsession.IsTimeout  Session. AttributeView. S9Visited.Subsession.ID  Etc.  Session. AttributeView.S9Visited. ’Origin-Host’  Session. AttributeView. S9Visited. IsTimeout  Session. AttributeView. S9Visited. ID  Etc.  Please note, even though the keyword *Subsession* is the same in both *S9Home* and *S9Visited* dictionaries, the dictionaries pointed by both these Subsessions would be different. This keyword is same only because that’s the name given in the dictionary, the name of the subsession can be changed and is not fixed.  That is, you may change *Subsession* to any name one might consider. |

|  |
| --- |
| Explicit Access  S9Home  Session[*“session\_id\_s9”*].AttributeView.S9Home.Subsession[1]. ’Origin-Host’  Session[*“session\_id\_s9”*].AttributeView.S9Home.Subsession[1].IsTimeout  Session[*“session\_id\_s9”*].AttributeView..S9Home.Subsession[1].ID <- - Integer  Etc.  Session[*“session\_id\_s9”*]. AttributeView.S9Home.ID < - - String  Session[*“session\_id\_s9”*]. AttributeView.S9Home. ’Origin-Host’  Session[*“session\_id\_s9”*]. AttributeView.S9Home. IsTimeout  Etc.  S9Visited  Session[*“session\_id\_s9”*]. AttributeView.S9Visited.Subsession[1]. ’Origin-Host’  Session[*“session\_id\_s9”*]. AttributeView..Subsession[1].IsTimeout  Session[*“session\_id\_s9”*]. AttributeView. S9Visited.Subsession[1].ID  Etc.  Session[*“session\_id\_s9”*]. AttributeView.S9Visited. ’Origin-Host’  Session[*“session\_id\_s9”*]. AttributeView. S9Visited. IsTimeout  Session[*“session\_id\_s9”*]. AttributeView. S9Visited. ID  Etc. |

This model has various benefits and is probably the most intuitive amongst the design options we have.

We could create a new tag called “Subsession” in the session dictionary indicating that this node onwards consider the attributes as though part of a subsession.

This approach will require the most amount of changes and testing.

**Raising Events:**

The events raising infrastructure would need to undergo modifications.

**Pros:**

1. This truly adds support for a subsession because it’s flexible and policy access for subsession is through the S9 session.

**Cons:**

1. This would bring in huge modifications to the Session Manager. Not only for support of this new type of control session but also its persistency etc.
2. Persisting any modification to one subsession could potentially lead to persisting all subsessions. This would have an adverse impact on performance.

#### Other Session Models

Below are the design options:

1. Make subsession a structure attribute within the S9 dictionary.

Mimic .IsNew / .IsTimeout etc to be raised for these subsessions.

**STATUS:** Rejected

This is rejected because making subsession a structure attribute is quite un-intuitive.

1. Make S9 session dictionary point to the subsession dictionary.

**STATUS:** Rejected.

This approach too is pretty non-intuitive.

## Event Triggers

**Some Background**

Over the Gx interface:

1. How does the PCRF subscribe for event triggers?
   * There is a global level CLI using which every subscriber would then subscribe to those Event triggers.
   * There are some event triggers that are reported by the PCEF which the PCRF need not subscribe to.
   * Supported features over the Rx interface translate to per Gx session Event Trigger subscription requests.
   * Because this value is at a global level, in case of VPCRF/HPCRF, this could lead to unnecessary traffic over S9.

### Use Cases

1. At the HPCRF,
   1. Ability for each Subsession to subscribe to particular event triggers.
   2. Ability for each Subsession to process event triggers.
   3. Ability for the parent S9 session to subscribe to event triggers. (Phase - 3)
   4. Ability for the parent S9 session to process event triggers. This won’t be supported because event triggers don’t get reported at the command level
   5. Ability to combine event trigger requests into the command level if each Subsession needs to subscriber to it. (Phase - 3)

1. At the VPCRF,
   1. Forward event trigger subscription request received for S9 Subsession over the corresponding Gx session.
   2. Forward event trigger responses of each Gx session over the corresponding S9 Subsession.
   3. Forward event trigger subscription request received for S9 parent session over all the related Gx sessions.
   4. Ability to combine event trigger requests into the command level if each Gx session needs to subscriber to it. (Phase - 3)
      1. OPTIMIZATION for event triggers like RAT type that could potentially apply for the S9 session at command level
   5. Ability for the VPCRF to locally subscribe to additional event triggers. (Phase - 2)

1. Allow Gxserver (legacy) to also subscribe to event triggers from Policy.

### Design Considerations

1. Policy author should be able to control the event subscription lifecycle on S9.
2. The means to subscribe event triggers on H-PCRF - S9 session and S9Subsession must be similar and intuitive.
3. List out independent Event triggers and allow only these to be provisioned from Policy.

### Implementation

**+ Policy\_Subscribed\_Event\_Trigger < - - List - - >**

The above attribute will be added to the S9 and S9Subsession dictionary.

All policy requested event triggers will be added to this list. Removal of any event trigger would also be done only by policy. The subsystem would not clear this list (even if the request failed).

Will be persisted.

**+ Subscribed\_Event\_Trigger < - - List - - >**

This list will contain only the event triggers that have been successfully installed on the PCEF/V-PCRF.

If a RAR fails, then this list must not be updated.

In case of RAR workflow, Service Designer would need to provide a callback mechanism for S9 so that S9 can indicate whether evaluation should continue and a message should be sent from Service Designer to S9 because a new event trigger has been provisioned. This callback must be called only after feature evaluation has been done for the subsession.

Such a callback is not needed to be called during CCR workflows of S9.

Looking at the above two lists, the S9 subsystem will need to figure out whether the event triggers list has been modified and therefore encode all the event triggers in the relevant session/sub-session.

Add the policy attribute to Gx server as well.

From an implementation perspective, a common Event Trigger class will need to be made in DiameterAVPUtils and this class would provide the Event Trigger encode, decode, find a change functionalities to all 3 interfaces - GxServer, S9 parent, S9 subsession.

## Logging and Debugging

Logging:

Create a new header file "S9SvTrace.h" in S9 subsystem with below contents.

|  |
| --- |
| //! @file S9SvTrace.h  //!  //! Description: This file contains the declaration of different trace levels supported in S9 subsystem.  //!  //! It also contains the declarion of trace channel defined for S9 subsystem  #include <svtrace/SvTrace.h>  //! Debug trace levels of S9 subsystem  enum S9\_TRACE\_LEVEL  {  S9\_DEBUG\_LEVEL\_DISABLED=0,  S9\_DEBUG\_LEVEL\_ERROR,  S9\_DEBUG\_LEVEL\_WARNING,  S9\_DEBUG\_LEVEL\_INFO  };  //! Define trace channel support\_s9  SV\_TRACE\_CHANNEL(support\_s9);  //! Set the trace level with specified value  //! @param level is the value to set the trace level  void S9TraceInit(unsigned int level);  //! Function to check if trace is enabled  //! @param S9\_TRACE\_LEVEL is the level given to check if that level is enabled  //! @return true if the given level is enabled. If no param is given as input, it checks with the default value of 1.  bool TraceEnabled(S9\_TRACE\_LEVEL 1); |

Create a new file S9SvTrace.cpp with below contents.

|  |
| --- |
| //! @file S9SvTrace.cpp  //!  //! Description: This file contains definition of trace channel supported in S9 subsystem.  //!  //! It also contains the definition of functions supporting tracing.  #include <svtrace/SvTrace.h>  #include "S9SvTrace.h"  //!To difine the trace channel "support\_s9" add below line  SV\_TRACE\_DEFINE\_CHANNEL(support\_s9);  //! Set the trace level with specified level  void S9TraceInit(unsigned int level)  {  support\_s9->m\_traceLevel = level;  SV\_TRACE\_LEVEL(subsystem\_s9,S9\_DEBUG\_LEVEL\_INFO,"Start tracing with the level %d", level);  }  //! Function to check if trace is enabled , where level is optional parameter  bool TraceEnabled(S9\_TRACE\_LEVEL level)  {  return (support\_s9->m\_traceLevel >= level);  } |

When s9 session is created/updated/terminated , its status(successful/unsuccessful) must be logged to /var/log/svsde\_trace.log

Example: SV\_TRACE\_LEVEL(subsystem\_s9,S9\_DEBUG\_LEVEL\_INFO,"S9 session successfully created for subscriber s1");

Auditing:

The Auditing subsystem design is given for CCR-I and the same should be replicated for other message types like CCR-U, CCR-T, RAA etc.

In the file "/vobs/fw/include/AuditManager.h" add enum RecordType S9\_CCR\_I\_MSG at the end to tell what event we are auditing.

|  |
| --- |
| class AuditManager  {  public:  enum RecordType{  /\* different audit report types are already defined here  SUBSCRIBER\_LOGIN = 1,  SUBSCRIBER\_LOGOUT,  SUBSCRIBER\_LOADED,  SUBSCRIBER\_UPDATE\_FROM\_SPR,  \*/  S9\_CCR\_I\_MSG  }  } |

Assign level for each record type in their corresponding audit level.

|  |
| --- |
| static AuditLevel getTypeLevel(RecordType t)  {  switch(t)  {  /\*different audit report types are already defined here  case SUBSCRIBER\_LOADED:  case SUBSCRIBER\_UPDATE\_FROM\_SPR:  case SESSION\_UPDATE:  case SESSION\_CREATE:  case SESSION\_DELETE:  case GX\_CCR\_I\_MSG:  case GX\_CCA\_I\_MSG:  \*/  case S9\_CCR\_I\_MSG:  return FULL\_AUDIT;  /\* if some more reports types are there , define them here  according to their different audit levels\*/  }  } |

We need to map the RecordType to its corresponding message

in "/vobs/fw/support/subsystems/audit/src/AuditManager.cpp"

|  |
| --- |
| void AuditManager::createMap()  {  m\_typesTable[S9\_CCR\_I\_MSG] = "S9\_CCR\_I\_MSG";  /\* some more messages if defined will go here\*/  } |

Introduce a new class in "/vobs/fw/support/common/S9ServerAudit.h" in S9 Sub system to enable the audit functionality

|  |
| --- |
| //!@ file S9ServerAudit.h  //! Description: This file enables auditing for S9 subsystem  //! Contains declaration of functions needed for auditing    class S9ServerAudit  {  public:  //! Ctor for S9ServerAudit  S9ServerAudit(const S9WorkflowData& S9WorkflowDataPtr);    //! Dtor S9ServerAudit  virtual ~S9ServerAudit(){};    //! It is used for formatting the audit string  static const std::string cSeparator;    //! This reference is used for accessing the work flow data  S9WorkflowData& S9WorkflowDataPtr;    //! This reference is used for calling APIs of AuditManager  AuditManager& m\_audit;    //! The log level of report type  AuditManager::LogLevel m\_auditLogLevel;    //! Map to store report type as key and its log level as value  static const std::map<AuditManager::RecordType reportType, AuditManager::LogLevel AuditLevel> m\_reportAuditLevel;    //! This function is called from different classes to peform audit on different report types.  void PerformAudit(ReportType reportType);    //! This is called when Audit needs to be written for CCRI message.  WriteDefaultAuditForCCRI();  /\* Some more functions like WriteDefaultAuditForCCRI()  are needed for different messages.\*/  } |

This class is initialized in the "class S9WorkflowData".

Add a data member m\_audit as below.

|  |
| --- |
| class S9WorkflowData  {  S9ServerAudit m\_audit;  } |

The m\_audit is initialized in constructor of S9WorkflowData as below.

|  |
| --- |
| S9WorkflowData::S9WorkflowData():m\_audit(\*this) |

Add a function to return reference of m\_audit

|  |
| --- |
| //! Function to return reference of m\_audit  //!@return reference of m\_audit  S9ServerAudit& GetAudit()  {  return m\_audit;  } |

In "/vobs/fw/support/common/S9ServerAudit.cpp" define S9ServerAudit

|  |
| --- |
| //!@file S9ServerAudit.cpp  //! Description: This file contains the definition of functions needed for auditing  //! Map to store report type as key and its log level as value  //! initialize the map with values  const std::map<AuditManager::RecordType reportType, AuditManager::LogLevel AuditLevel> S9ServerAudit::m\_reportAuditLevel =  {  {AuditManager::RecordType::S9\_CCR\_I\_MSG,AuditManager::DEFAULT\_LOG}  /\* some more record types are added with their corresponding log level\*/  };  //! Function called from different classes to peform audit on different report types.  void S9ServerAudit::PerformAudit(ReportType reportType)  {  /\* get the corresponding log level from map\*/  map::iterator i= m\_reportAuditLevel.find(reportType);  if(i!= m\_reportAuditLevel.end())  {  switch(i->second)  {  case DEFAULT\_LOG:  switch(reportType)  {  case S9\_CCR\_I\_MSG:  WriteDefaultAuditForCCRI();  break;  /\* Other cases for required message types will go here\*/  default:  assert(0);  }  } /\* Other cases for required log level will go here\*/    }  }  //!Writed Audit for CCRI message.  void S9ServerAudit::WriteDefaultAuditForCCRI()  {  std::string auditSring+ ="CCR-I Received: Request-type:" +cSeparator+"S9WorkflowDataPtr.GetPeerHost()";    m\_audit.write(S9WorkflowDataPtr.GetSubscriberName(),  toString(S9WorkflowDataPtr.GetDataSessionId()),  S9WorkflowDataPtr.GetSessionId(),  AuditManager::ReportType::S9\_CCR\_I\_MSG  auditString);  } |

A typical invocation of PerformAudit function looks like below.

|  |
| --- |
| Example: S9WorkflowDataPtr->GetAudit().PerformAudit(S9\_CCR\_I\_MSG); |

## Reports

**Note:**

1. Workflow expression and session attribute expressions mentioned below are subject to change as under discussion.
2. If DRA begin used between V-PCRF and H-PCRF per Peer reports doesn’t make sense, hence operator-name should be used.

Supporting operator-name need a discussion: whether could be a policy package configuration?

|  |
| --- |
| **SRS: S9-SRS-RPT-0010 and S9-SRS-RPT-0020**  Title - S9 Server Request Message by Message Type & Peer  Title - S9 Server Answer Message by Message Type & Peer  **File Path:** /vobs/fw-policy/products/s9\_server/sde/sandvine/etc/s9\_server/policypkg/policyContent/  **File Name:** **S9Server\_measurement.policy**  **File Contents:**  #! If {[S9ServerPolicyConfig NdsEnabled]} {  Classifier “S9ServerMessageType” values "Credit-Control-Initial" "Credit-Control-Update" "Credit-Control-Terminate" "Re-Authorization"  Classifier “S9ServerPeer” type string  Measurement “S9ServerRequestMessages” over publish\_interval \  Unique by (Classifier.S9ServerMessageType, Classifier.S9ServerPeer)  Publish “S9ServerRequestMessages” Measurement.S9ServerRequestMessages  **File Name:** **s9\_server\_nds\_reports.policy**  **File Contents:**  PolicyGroup expr(Workflow. S9ServerSessionEstablishment.Measure) all  {  If true then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Credit-Control-Initial’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionEstablishment.Measure.S9.’CC-Request’.’Origin-Host’ and \  Increment Measurement.S9ServerRequestMessages  }  PolicyGroup expr(Workflow. S9ServerSessionModification.Measure) all  {  If true then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Credit-Control-Update’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Origin-Host’ and \  Increment Measurement.S9ServerRequestMessages  }  PolicyGroup expr(Workflow. S9ServerSessionTermination.Measure) all  {  If true then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Credit-Control-Terminate’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionTermination.Measure.S9.’CC-Request’.’Origin-Host’ and \  Increment Measurement.S9ServerRequestMessages  }  PolicyGroup expr(Workflow. S9ServerSessionReAuthorization.Measure) all  {  If true then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Re-Authorization’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionReAuthorization.Measure.S9.’Re-Auth-Answer’.’Origin-Host’ and \  Increment Measurement.S9ServerRequestMessages  }  #! } |

|  |
| --- |
| **SRS: S9-SRS-RPT-0030**  Title - S9 Server Errors by Message Type, Error Type & Peer  **File Path:** /vobs/fw-policy/products/s9\_server/sde/sandvine/etc/s9\_server/policypkg/policyContent/  **File Name:** **S9Server\_measurement.policy**  **File Contents:**  Classifier “S9ServerMessageType” values "Credit-Control-Initial" "Credit-Control-Update" "Credit-Control-Terminate" "Re-Authorization"  Classifier “S9ServerErrorType” values string  Classifier “S9ServerPeer” values string  Measurement “S9ServerErrorMessages” \  over publish\_interval \  unique by(Classifier.S9ServerMessageType, Classifier.S9ServerErrorType, Classifier.S9ServerPeer)  publish “S9ServerErrorMessages” Measurement.S9ServerErrorMessages  **File Name:** **s9\_server\_nds\_reports.policy**  **File Contents:**  PolicyGroup expr(Workflow. S9ServerSessionEstablishment.Measure) all  {  If expr(Workflow.S9ServerSessionEstablishment.Measure.ErrorCode is not null) then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Credit-Control-Initial’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionEstablishment.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ServerErrorType = Workflow.S9ServerSessionEstablishment.Measure.ErrorCode and \  Increment Measurement.S9ServerErrorMessages  }  PolicyGroup expr(Workflow. S9ServerSessionModification.Measure) all  {  If expr(Workflow.S9ServerSessionModification.Measure.ErrorCode is not null) then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Credit-Control-Update’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ServerErrorType = Workflow.S9ServerSessionMeasurement.Measure.ErrorCode and \  Increment Measurement.S9ServerErrorMessages  }  PolicyGroup expr(Workflow. S9ServerSessionTermination.Measure) all  {  If expr(Workflow.S9ServerSessionTermination.Measure.ErrorCode is not null) then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Credit-Control-Terminate’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionTermination.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ServerErrorType = Workflow.S9ServerSessionTermination.Measure.ErrorCode and \  Increment Measurement.S9ServerErrorMessages  }  PolicyGroup expr(Workflow. S9ServerSessionReAuthorization.Measure) all  {  If expr(Workflow.S9ServerSessionReAuthorization.Measure.ErrorCode is not null) then \  Set Classifier.S9ServerMessageType = Classifier.S9ServerMessageType.’ Re-Authorization’ and \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionReAuthorization.Measure.S9.’Re-Auth-Answer’.’Origin-Host’ and \  Set Classifier.S9ServerErrorType = Workflow.S9ServerSessionReAuthorization.Measure.ErrorCode and \  Increment Measurement.S9ServerErrorMessages  }  !#} |

|  |
| --- |
| **SRS: S9-SRS-RPT-0040 and S9-SRS-RPT-0050**  Title - S9 Client Request Message by Message Type & Peer  Title - S9 Client Answer Message by Message Type & Peer  **File Path:** /vobs/fw-policy/products/s9\_client/sde/sandvine/etc/s9\_client/policypkg/policyContent/  **File Name:** **S9Client\_measurement.policy**  **File Contents:**  #! If {[S9ClientPolicyConfig NdsEnabled]} {  Classifier “S9ClientMessageType” values "Credit-Control-Initial" "Credit-Control-Update" "Credit-Control-Terminate" "Re-Authorization"  Classifier “S9ClientPeer” type string  Measurement “S9ClientRequestMessages” over publish\_interval \  Unique by (Classifier.S9ClientMessageType, Classifier.S9ClientPeer)  Publish “S9ClientRequestMessages” Measurement.S9ClientRequestMessages  **File Name:** **s9\_client\_nds\_reports.policy**  **File Contents:**  PolicyGroup expr(Workflow. S9ClientSessionEstablishment.Measure) all  {  If true then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Credit-Control-Initial’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionEstablishment.Measure.S9.’CC-Request’.’Destination-Host’ and \  Increment Measurement.S9ClientRequestMessages  }  PolicyGroup expr(Workflow. S9ClientSessionModification.Measure) all  {  If true then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Credit-Control-Update’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionModification.Measure.S9.’CC-Request’.’ Destination-Host’ and \  Increment Measurement.S9ClientRequestMessages  }  PolicyGroup expr(Workflow. S9ClientSessionTermination.Measure) all  {  If true then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Credit-Control-Terminate’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionTermination.Measure.S9.’CC-Request’.’ Destination-Host’ and \  Increment Measurement.S9ClientRequestMessages  }  PolicyGroup expr(Workflow. S9ClientSessionReAuthorization.Measure) all  {  If true then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Re-Authorization’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionReAuthorization.Measure.S9.’Re-Auth-Answer’.’ Destination-Host’ and \  Increment Measurement.S9ClientRequestMessages  }  #! } |

|  |
| --- |
| SRS: SRS-RPT-0070  Title - S9 Client Errors by Message Type, Error Type & Peer  **File Path:** /vobs/fw-policy/products/s9\_client/sde/sandvine/etc/s9\_client/policypkg/policyContent/  **File Name:** **S9Client\_measurement.policy**  **File Contents:**  Classifier “S9ClientMessageType” values "Credit-Control-Initial" "Credit-Control-Update" "Credit-Control-Terminate" "Re-Authorization"  Classifier “S9ClientErrorType” values string  Classifier “S9ClientPeer” values string  Measurement “S9ClientErrorMessages” \  over publish\_interval \  unique by(Classifier.S9ClientMessageType, Classifier.S9ServerErrorType, Classifier.S9ClientPeer)  publish “S9ClientErrorMessages” Measurement.S9ClientErrorMessages  **File Name:** **s9\_client\_nds\_reports.policy**  **File Contents:**  PolicyGroup expr(Workflow. S9ClientSessionEstablishment.Measure) all  {  If expr(Workflow.S9ClientSessionEstablishment.Measure.ErrorCode is not null) then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Credit-Control-Initial’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionEstablishment.Measure.S9.’CC-Request’.’Destination-Host’ and \  Set Classifier.S9ClientErrorType = Workflow.S9ClientSessionEstablishment.Measure.ErrorCode and \  Increment Measurement.S9ClientErrorMessages  }  PolicyGroup expr(Workflow. S9ClientSessionModification.Measure) all  {  If expr(Workflow.S9ClientSessionModification.Measure.ErrorCode is not null) then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Credit-Control-Update’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionModification.Measure.S9.’CC-Request’.’Destination-Host’ and \  Set Classifier.S9ClientErrorType = Workflow.S9ClientSessionMeasurement.Measure.ErrorCode and \  Increment Measurement.S9ClientErrorMessages  }  PolicyGroup expr(Workflow. S9ClientSessionTermination.Measure) all  {  If expr(Workflow.S9ClientSessionTermination.Measure.ErrorCode is not null) then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Credit-Control-Terminate’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionTermination.Measure.S9.’CC-Request’.’Destination-Host’ and \  Set Classifier.S9ClientErrorType = Workflow.S9ClientSessionTermination.Measure.ErrorCode and \  Increment Measurement.S9ClientErrorMessages  }  PolicyGroup expr(Workflow. S9ClientSessionReAuthorization.Measure) all  {  If expr(Workflow.S9ClientSessionReAuthorization.Measure.ErrorCode is not null) then \  Set Classifier.S9ClientMessageType = Classifier.S9ClientMessageType.’ Re-Authorization’ and \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionReAuthorization.Measure.S9.’Re-Auth-Answer’.’Destination-Host’ and \  Set Classifier.S9ClientErrorType = Workflow.S9ClientSessionReAuthorization.Measure.ErrorCode and \  Increment Measurement.S9ClientErrorMessages  }  #!} |

|  |
| --- |
| **SRS: S9-SRS-RPT-0070**  Rx-Server: Sessions by Enforcement per peer, unique by Enforcement Type  **File Path:**  /view/rsaxena\_PCRF\_760\_rxmedia/vobs/fw-policy/products/rx\_server/sde/sandvine/etc/rx\_server/policypkg/policyContent  **File Name:** rxserver\_measurements.policy  **File Contents:**  classifier "RxSessionEnforcementType" values "Gx" "PCMM" “S9Server” “S9Client” |

|  |
| --- |
| **SRS: S9-SRS-RPT-0080**  Session-Management: Session Length by Session Protocol  **File Path:**  /view/rsaxena\_PCRF\_760\_rxmedia/vobs/fw-policy/products/gx\_server/linux/sandvine/etc/gx\_server/policypkg/policyContent  **File Name:** session\_measurements.policy  **File Contents:**  classifier "SessionMgmtProtocol" values "Gx" "Data" "SPB" “S9Server” “S9Client” |

|  |
| --- |
| **SRS: S9-SRS-RPT-0090 and S9-SRS-RPT-0100**  Service-Designer:  Rules unique by Rule Name and Roaming Type (Local & Roaming)  Rules unique by Rule Action and Roaming Type (Local & Roaming)  File Path:  /view/rsaxena\_PCRF\_760\_rxmedia/vobs/fw-policy/products/gx\_server/linux/sandvine/etc/gx\_server/policypkg/policyContent  File Name: service\_designer\_measurement.policy  File Content:  Classifier “ServiceDesignerRoamingType” values "Roaming" "Local"  measurement "ServiceDesignerRules" over publish\_interval \  unique by(Classifier.ServiceDesignerRuleAction, Classifier.ServiceDesignerRuleName, Classifier.ServiceDesignerRoamingType)  publish "ServiceDesignerRules" Measurement.ServiceDesignerRules |

|  |
| --- |
| SRS: S9-SRS-RPT-0101  GxProcessingLatency by MessageType and Roaming Type(Local & Roaming) per peer  File Path:  /view/rsaxena\_PCRF\_760\_rxmedia/vobs/fw-policy/products/gx\_server/linux/sandvine/etc/gx\_server/policypkg/policyContent  **File Name:** gx\_measurements.policy  **File Content:**  Classifier “GxServerRoamingType” values "Roaming" "Local"  Classifier “GxSeverPeer” type string  measurement "GxServerLatency" \  histogram(Histogram.GxServerLatencyHistogram) \  over publish\_interval \  unique by (Classifier.GxServerMessageType, GxServerRoamingType, GxServerPeer)  publish "GxServerLatency" Measurement.GxServerLatency  **File Name:** gx\_nds\_reports.policy  **File Content:**  PolicyGroup expr(Workflow.GxIPCANSessionEstablishment.Measure) all  {  // Set values to GxServerRoamingType and GxServerPeer Classifiers  set classifier.GxServerPeer = Session.AttributeView.Gx.Origin\_Host  set Classifier.GxServerRoamingType = ((Session.AttributeView.Gx.Roaming)? \  Classifier.GxServerRoamingType.Roaming: \  Classifier.GxServerRoamingType.Local) and \  } |

|  |
| --- |
| SRS: S9-SRS-RPT-0102  GxMessages report by MessageType and Roaming Type (Local & Roaming)  **File Path:**  /view/rsaxena\_PCRF\_760\_rxmedia/vobs/fw-policy/products/gx\_server/linux/sandvine/etc/gx\_server/policypkg/policyContent  **File Name:** gx\_measurements.policy  **File Content:**  Classifier “GxServerRoamingType” “Roaming” “Local”  measurement "GxServerRequestMessages" over publish\_interval \  unique by (Classifier.GxServerMessageType, Classifier.GxServerRoamingType)  publish "GxServerRequestMessages" Measurement.GxServerRequestMessages |

|  |
| --- |
| SRS: S9-SRS-RPT-0103  Title - S9 Server EventTriggersReportedPerPeer  **File Path:** /vobs/fw-policy/products/s9\_server/sde/sandvine/etc/s9\_server/policypkg/policyContent/  **File Name:** S9Server\_measurement.policy  **File Contents:**  Classifier “S9ServerEventTriggers” values “IP\_CAN\_CHANGE” “QOS\_CHANGE” “LOSS\_OF\_BEARER” “RAT\_CHANGE” “SGSN\_CHANGE” “USER\_LOCATION\_CHANGE” “UE\_IP\_ADDRESS\_ALLOCATE” “UE\_IP\_ADDRESS\_RELEASE“ “RESOURCE\_MODIFICATION\_REQUEST” “UE\_LOCAL\_IP\_ADDRESS\_CHANGE “  Classifier “S9ServerPeer” type string  Measurement “S9ServerEventTriggersReportedPerPeer” over publich\_interval \  Unique by (Classifier.S9ServerPeer)  Publish “S9ServerEventTriggersReportedPerPeer” Measurement. S9ServerEventTriggersReportedPerPeer  File Name: **s9\_server\_nds\_reports.policy**  File Content:  PolicyGroup expr(Workflow.S9ServerSessionModification.Measure) all  {  If true then \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ServerEventTriggers = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Event-Trigger’  Increment Measurement. S9ServerEventTriggersReportedPerPeer  }  PolicyGroup expr(Workflow.S9ServerSessionTermination.Measure) all  {  If true then \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionTermination.Measure.S9.’CC-Request’.’Event-Trigger’ and \  Set Classifier.S9ServerEventTriggers = Classifier.S9ServerSessionTermination.Measure.S9.’CC-Request’.’Event-Trigger’ and \  Increment Measurement. S9ServerEventTriggersReportedPerPeer  } |

|  |
| --- |
| SRS: S9-SRS-RPT-0104  Title - S9 Client EventTriggersReportedPerPeer  **File Path:** /vobs/fw-policy/products/s9\_client/sde/sandvine/etc/s9\_client/policypkg/policyContent/  **File Name:** S9Client\_measurement.policy  **File Contents:**  Classifier “S9ClientEventTriggers” values “IP\_CAN\_CHANGE” “QOS\_CHANGE” “LOSS\_OF\_BEARER” “RAT\_CHANGE” “SGSN\_CHANGE” “USER\_LOCATION\_CHANGE” “UE\_IP\_ADDRESS\_ALLOCATE” “UE\_IP\_ADDRESS\_RELEASE“ “RESOURCE\_MODIFICATION\_REQUEST” “UE\_LOCAL\_IP\_ADDRESS\_CHANGE “  Classifier “S9ClientPeer” type string  Measurement “S9ClientEventTriggersReportedPerPeer” over publich\_interval \  Unique by (Classifier.S9ClientPeer)  Publish “S9ClientEventTriggersReportedPerPeer” Measurement. S9ClientEventTriggersReportedPerPeer  File Name: **s9\_client\_nds\_reports.policy**  File Content:  PolicyGroup expr(Workflow.S9ClientSessionModification.Measure) all  {  If true then \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionModification.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ClientEventTriggers = Classifier.S9ClientSessionModification.Measure.S9.’CC-Request’.’Event-Trigger’ and \  Increment Measurement. S9ClientEventTriggersReportedPerPeer  }  PolicyGroup expr(Workflow.S9ClientSessionTermination.Measure) all  {  If true then \  Set Classifier.S9ClientPeer = Classifier.S9ClientSessionTermination.Measure.S9.’CC-Request’.’Event-Trigger’ and \  Set Classifier.S9ClientEventTriggers = Classifier.S9ClientSessionTermination.Measure.S9.’CC-Request’.’Event-Trigger’ and \  Increment Measurement. S9ClientEventTriggersReportedPerPeer  } |

|  |
| --- |
| SRS: S9-SRS-RPT-0106  Title - S9 Server UsageReported over period of time per Peer  **File Path:** /vobs/fw-policy/products/s9\_server/sde/sandvine/etc/s9\_server/policypkg/policyContent/  **File Name:** S9server\_measurement.policy  **File Contents:**  Classifier “S9ServerUsageReported” type integer  Classifier “S9ServerPeer” type string  Measurement “S9ServerUsageReportedOverPeriodOfTime” over publish\_interval \  Unique by (Classifier.S9ServerPeer)  Publish “S9ServerUsageReportedOverPeriodOfTime” Measurement. S9ServerUsageReportedOverPeriodOfTime  PolicyGroup expr(Workflow.S9ServerSessionModification.Measure) all  {  If true then \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ServerUsageReported = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Usage-Monitoring-Information’.’Used-Service-Unit’.’CC-Total-Octets’ and \  Increment Measurement. S9ServerUsageReportedOverPeriodOfTime  }  Note: Expression to access cc-total-octets might change as expressions are under discussion. |

|  |
| --- |
| SRS: S9-SRS-RPT-0107  Title - S9 Server GrantsAllocated over period of time per Peer  **File Path:** /vobs/fw-policy/products/s9\_server/sde/sandvine/etc/s9\_server/policypkg/policyContent/  **File Name:** S9server\_measurement.policy  **File Contents:**  Classifier “S9ServerGrantsAllocated” type integer  Classifier “S9ServerPeer” type string  Measurement “S9ServerGrantsAllocatedOverPeriodOfTime” over publish\_interval \  Unique by (Classifier.S9ServerPeer)  Publish “S9ServerGrantsAllocatedOverPeriodOfTime” Measurement. S9ServerGrantsAllocatedOverPeriodOfTime  PolicyGroup expr(Workflow.S9ServerSessionModification.Measure) all  {  If true then \  Set Classifier.S9ServerPeer = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Origin-Host’ and \  Set Classifier.S9ServerUsageReported = Classifier.S9ServerSessionModification.Measure.S9.’CC-Request’.’Usage-Monitoring-Information’.’Granted-Service-Unit’ and \  Increment Measurement. S9ServerGrantsAllocatedOverPeriodOfTime  }  Note: Expression to access granted-service-unit might change as expressions are under discussion. |