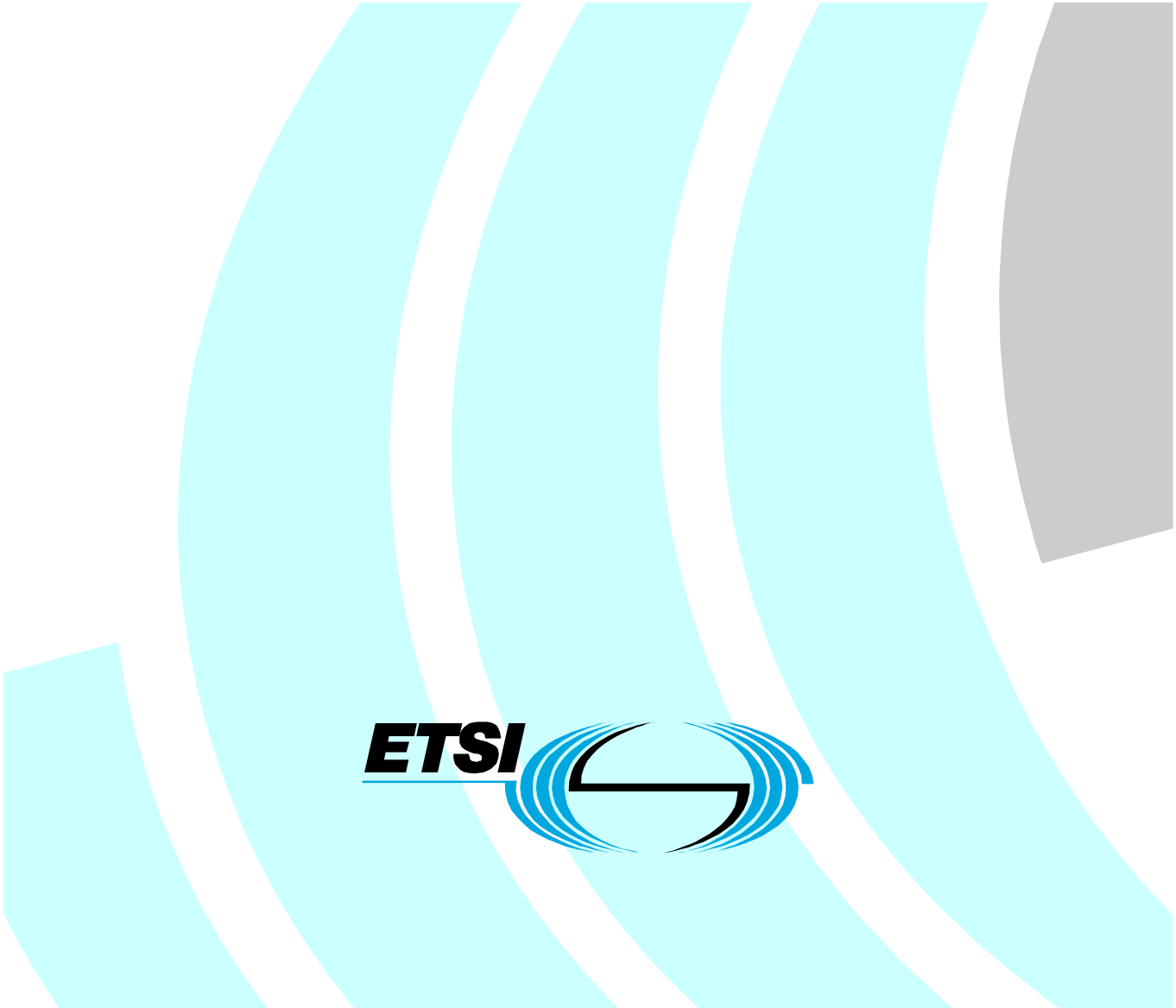
ETSI ES 202 196-8 V1.1.1 (2003-08)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Test Suite Structure and Test Purposes (TSS&TP); Part 8: Data session control SCF



Reference
DES/SPAN-120088-8

Keywords
API, OSA, TSS&TP

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Services and Protocols for Advanced Networks (SPAN).

The present document is part 8 of a multi-part deliverable. Full details of the entire series can be found in part 1 [6].

To evaluate conformance of a particular implementation, it is necessary to have a set of test purposes to evaluate the dynamic behaviour of the Implementation Under Test (IUT). The specification containing those test purposes is called a Test Suite Structure and Test Purposes (TSS&TP) specification.

1 Scope

The present document provides the Test Suite Structure and Test Purposes (TSS&TP) specification for the Data Session Control SCF of the Application Programming Interface (API) for Open Service Access (OSA) defined in ES 201 915-8 [1] in compliance with the relevant requirements, and in accordance with the relevant guidance given in ISO/IEC 9646-2 [4] and ETS 300 406 [5].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI ES 201 915-8: "Open Service Access (OSA); Application Programming Interface (API); Part 8: Data Session Control SCF (Parlay 3)".

[2] ETSI ES 202 170: "Open Service Access (OSA); Application Programming Interface (API); Implementation Conformance Statement (ICS) proforma specification for Framework and SCFs".

[3] ISO/IEC 9646-1: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts".

[4] ISO/IEC 9646-2: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 2: Abstract Test Suite specification".

[5] ETSI ETS 300 406: "Methods for Testing and Specification (MTS); Protocol and profile conformance testing specifications; Standardization methodology".

[6] ETSI ES 202 196-1: "Open Service Access (OSA); Application Programming Interface (API); Test Suite Structure and Test Purposes (TSS&TP); Part 1: Overview".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 201 915-8 [1], ISO/IEC 9646-1 [3] and ISO/IEC 9646-2 [4] and the following apply:

abstract test case: Refer to ISO/IEC 9646-1 [3].

Abstract Test Method (ATM): Refer to ISO/IEC 9646-1 [3].

Abstract Test Suite (ATS): Refer to ISO/IEC 9646-1 [3].

Implementation Under Test (IUT): Refer to ISO/IEC 9646-1 [3].

Lower Tester (LT): Refer to ISO/IEC 9646-1 [3].

Implementation Conformance Statement (ICS): Refer to ISO/IEC 9646-1 [3].

ICS proforma: Refer to ISO/IEC 9646-1 [3].

Implementation eXtra Information for Testing (IXIT): Refer to ISO/IEC 9646-1 [3].

IXIT proforma: Refer to ISO/IEC 9646-1 [3].

Test Purpose (TP): Refer to ISO/IEC 9646-1 [3].

3.2 Abbreviations

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

API	Application Programming Interface
ATM	Abstract Test Method
ATS	Abstract Test Suite
CM	Control Manager
DS	Data Session
DSC	Data Session Control
ICS	Implementation Conformance Statement
IUT	Implementation Under Test
IXIT	Implementation eXtra Information for Testing
LT	Lower Tester
OSA	Open Service Access
TP	Test Purpose
TSS	Test Suite Structure

4 Test Suite Structure (TSS)

Data Session Control SCF

- IpDataSessionControlManager (CM)
- IpDataSession (DS)

5 Test Purposes (TP)

5.1 Introduction

For each test requirement a TP is defined.

5.1.1 TP naming convention

TPs are numbered, starting at 01, within each group. Groups are organized according to the TSS. Additional references are added to identify the actual test suite (see table 1).

Table 1: TP identifier naming convention scheme

Identifier: <suite_id>_<group>_<nnn>		
<suite_id>	= SCF name:	"DSC" for **D**ata **S**ession **C**ontrol SCF
<group>	= group number:	two character field representing the group reference according to TSS
<nn>	= sequential number:	(01-99)

5.1.2 Source of TP definition

The TPs are based on ES 201 915-8 [1].

5.1.3 Test strategy

As the base standard ES 201 915-8 [1] contains no explicit requirements for testing, the TPs were generated as a result of an analysis of the base standard and the ICS specification ES 202 170 [2].

The TPs are only based on conformance requirements related to the externally observable behaviour of the IUT and are limited to conceivable situations to which a real implementation is likely to be faced (see ETS 300 406 [5]).

5.2 TPs for the Data Session Control SCF

All ICS items referred to in this clause are as specified in ES 202 170[2] unless indicated otherwise by another numbered reference.

All parameters specified in method calls are valid unless specified.

The procedures to trigger the SCF to call methods in the application are dependant on the underlying network architecture and are out of the scope of this test specification. Those method calls are preceded by the words "Triggered action".

5.2.1 Data Session Control

5.2.1.1 IpDataSessionControlManager

Test DSC_CM_01

Summary: **IpDataSessionControlManager**, mandatory methods, successful

Reference: ES 201 915-8 [1], clause 8.4

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Check: valid value of TpAssignmentID is returned

2. Triggered Action: cause IUT to call **reportNotification()** method on the tester's (application's) IpAppDataSessionControlManager interface.
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **destroyNotification()**
   Parameters: assignmentID
   Check: no exception is returned

Test DSC_CM_02

Summary: **IpDataSessionControlManager**, all methods, successful

Reference: ES 201 915-8 [1], clause 8.4

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Check: valid value of TpAssignmentID is returned

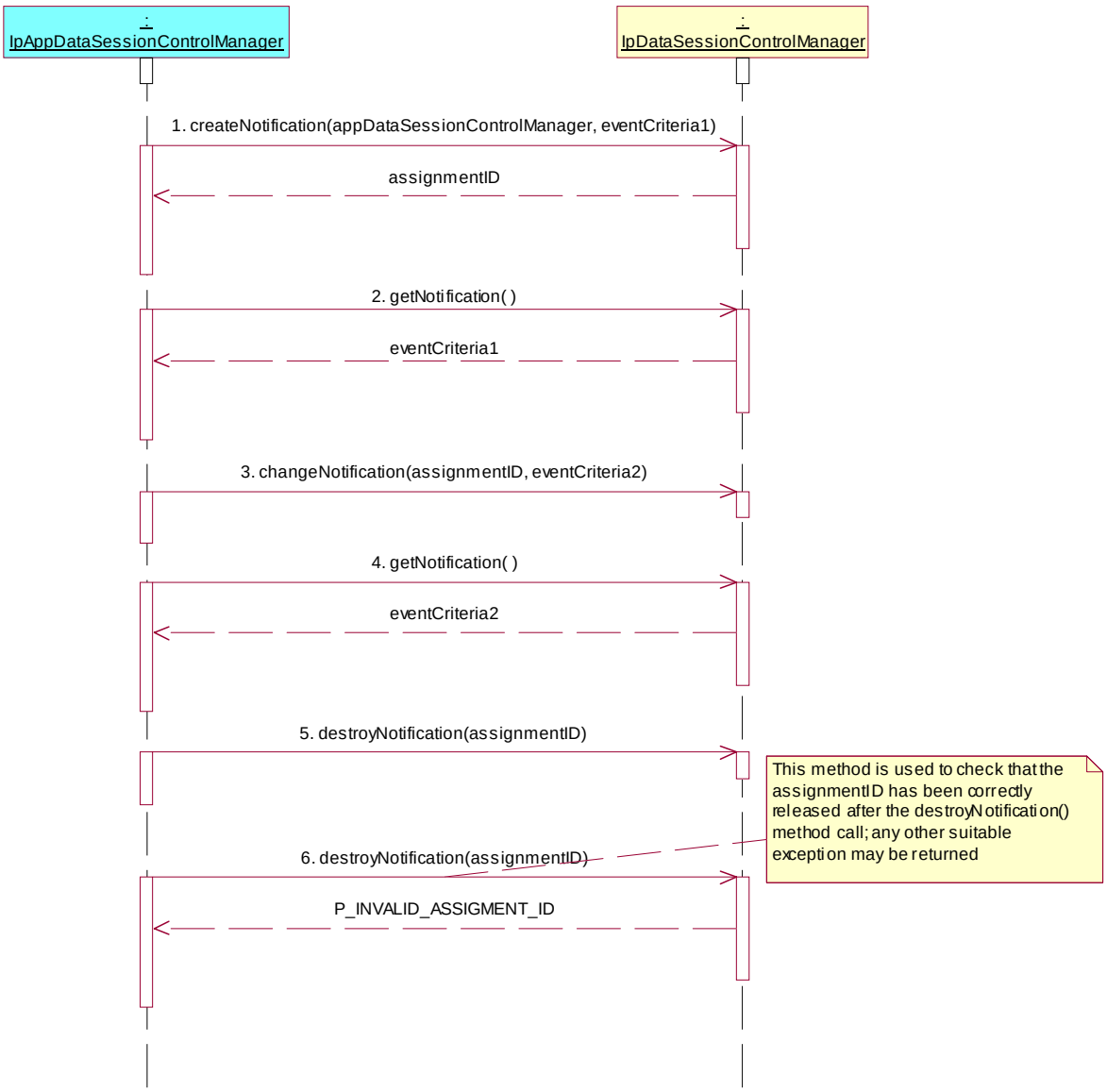
2. Method call **getNotification()**
   Parameters: none
   Check: value of TpDataSessionEventCriteria as specified in 1. is returned

3. Method call **changeNotification()**
   Parameters: assignmentID, eventCriteria (different to 1.)
   Check: no exception is returned

4. Method call **getNotification()**
   Parameters: none
   Check: value of TpDataSessionEventCriteria as specified in 3. is returned

5. Method call **destroyNotification()**
   Parameters: assignmentID
   Check: no exception is returned

6. Method call **destroyNotification()**
   Parameters: assignmentID as received in1. and destroyed in 5.
   Check: P_INVALID_ASSIGMENT_ID or any suitable exception is returned


IpAppDataSessionControlManager
IpDataSessionControlManager
1. createNotification(appDataSessionControlManager, eventCriteria1)
assignmentID
2. getNotification( )
eventCriteria1
3. changeNotification(assignmentID, eventCriteria2)
4. getNotification( )
eventCriteria2
5. destroyNotification(assignmentID)
This method is used to check that the assignmentID has been correctly released after the destroyNotification() method call; any other suitable exception may be returned
6. destroyNotification(assignmentID)
P_INVALID_ASSIGMENT_ID

Test DSC_CM_03

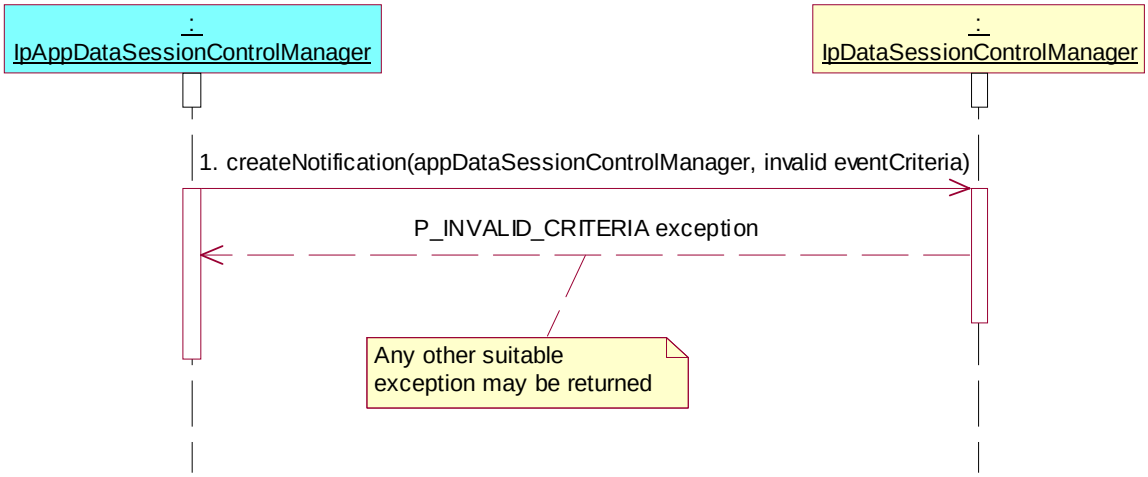
Summary: **IpDataSessionControlManager**, createNotification(), P_INVALID_CRITERIA exception

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), invalid eventCriteria
   Check: P_INVALID_CRITERIA or any suitable exception is returned



Test DSC_CM_04

Summary: **IpDataSessionControlManager**, destroyNotification(), P_INVALID_ASSIGMENT_ID exception

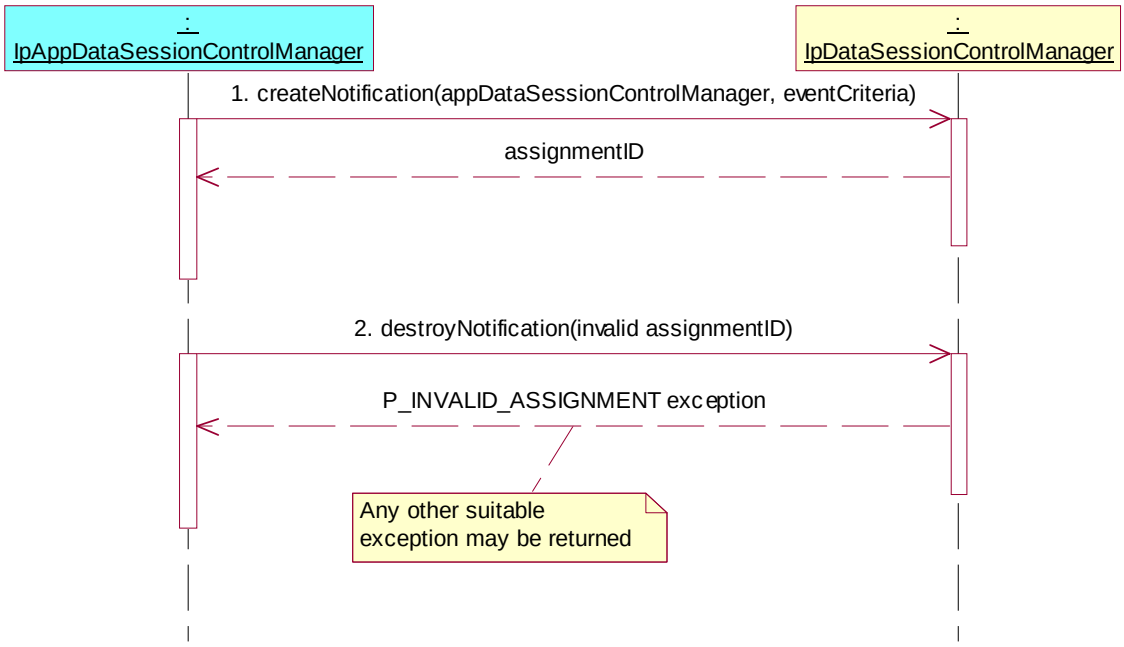
Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Check: valid value of TpAssignmentID is returned

2. Method call **destroyNotification()**
   Parameters: invalid assignmentID
   Check: P_INVALID_ASSIGMENT_ID or any suitable exception is returned



Test DSC_CM_05

Summary: **IpDataSessionControlManager**, changeNotification(), P_INVALID_ASSIGMENT_ID exception

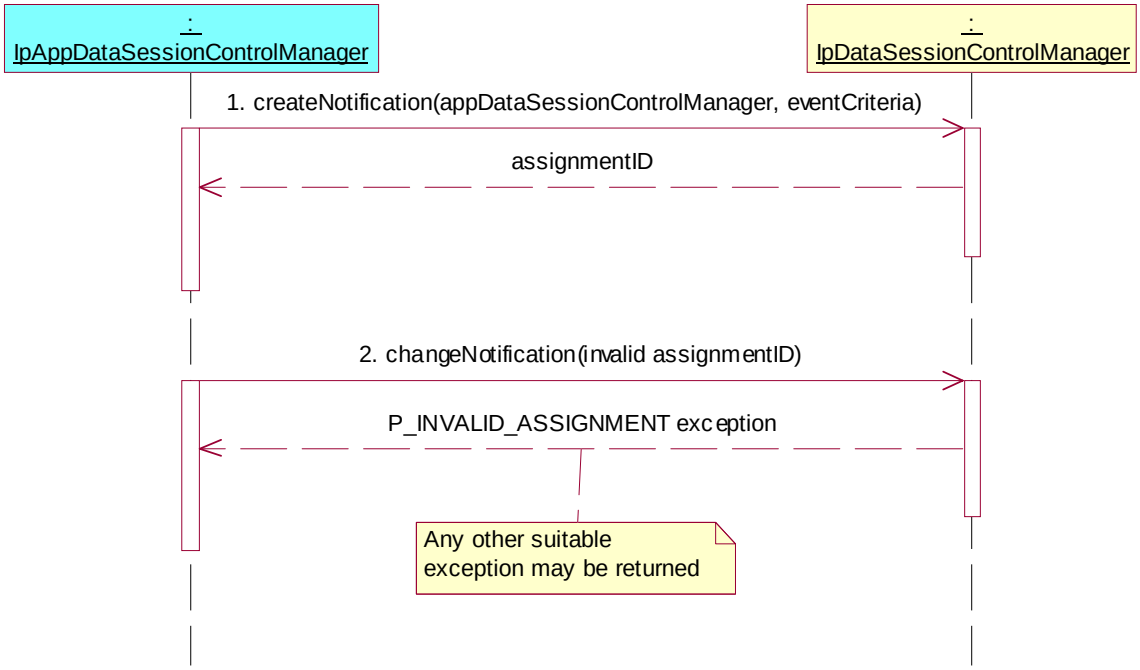
Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Check: valid value of TpAssignmentID is returned

2. Method call **changeNotification()**
   Parameters: invalid assignmentID, eventCriteria
   Check: P_INVALID_ASSIGMENT_ID or any suitable exception is returned



Test DSC_CM_06

Summary: **IpDataSessionControlManager**, changeNotification(), P_INVALID_CRITERIA exception

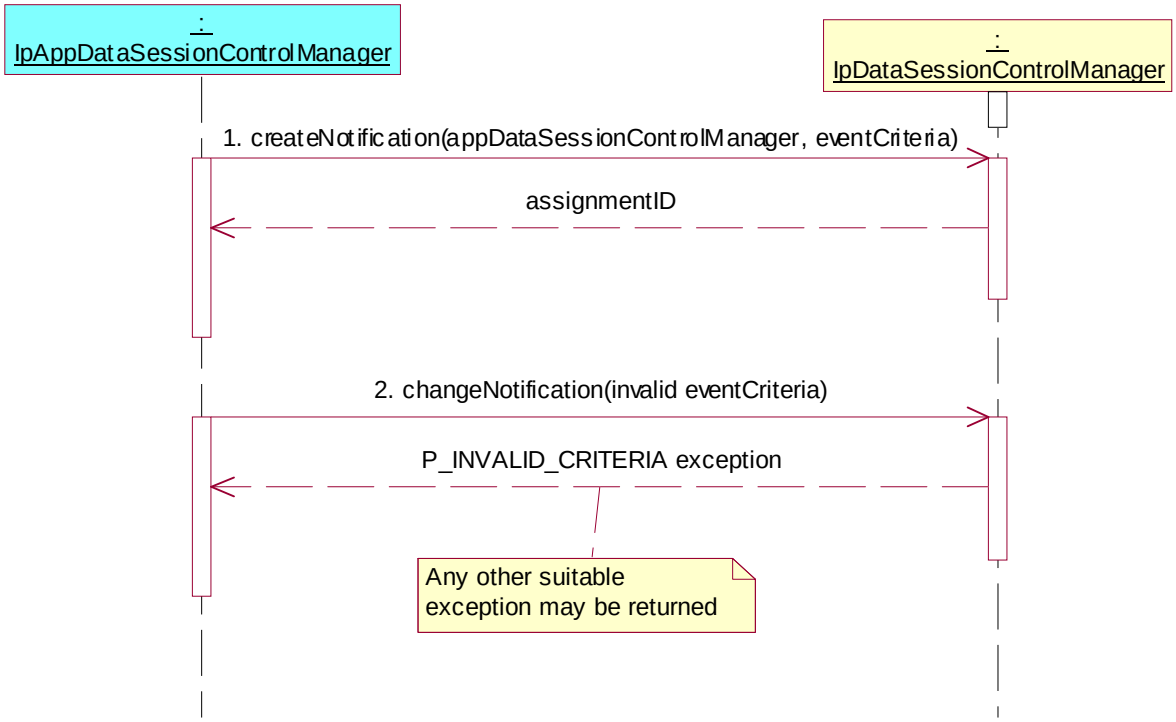
Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Check: valid value of TpAssignmentID is returned

2. Method call **changeNotification()**
   Parameters: assignmentID, invalid eventCriteria
   Check: P_INVALID_CRITERIA or any suitable exception is returned



Test DSC_CM_07

Summary: all methods, successful, two createNotification() method calls

Reference: ES 201 915-8 [1], clause 8.4

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Check: valid value of TpAssignmentID is returned

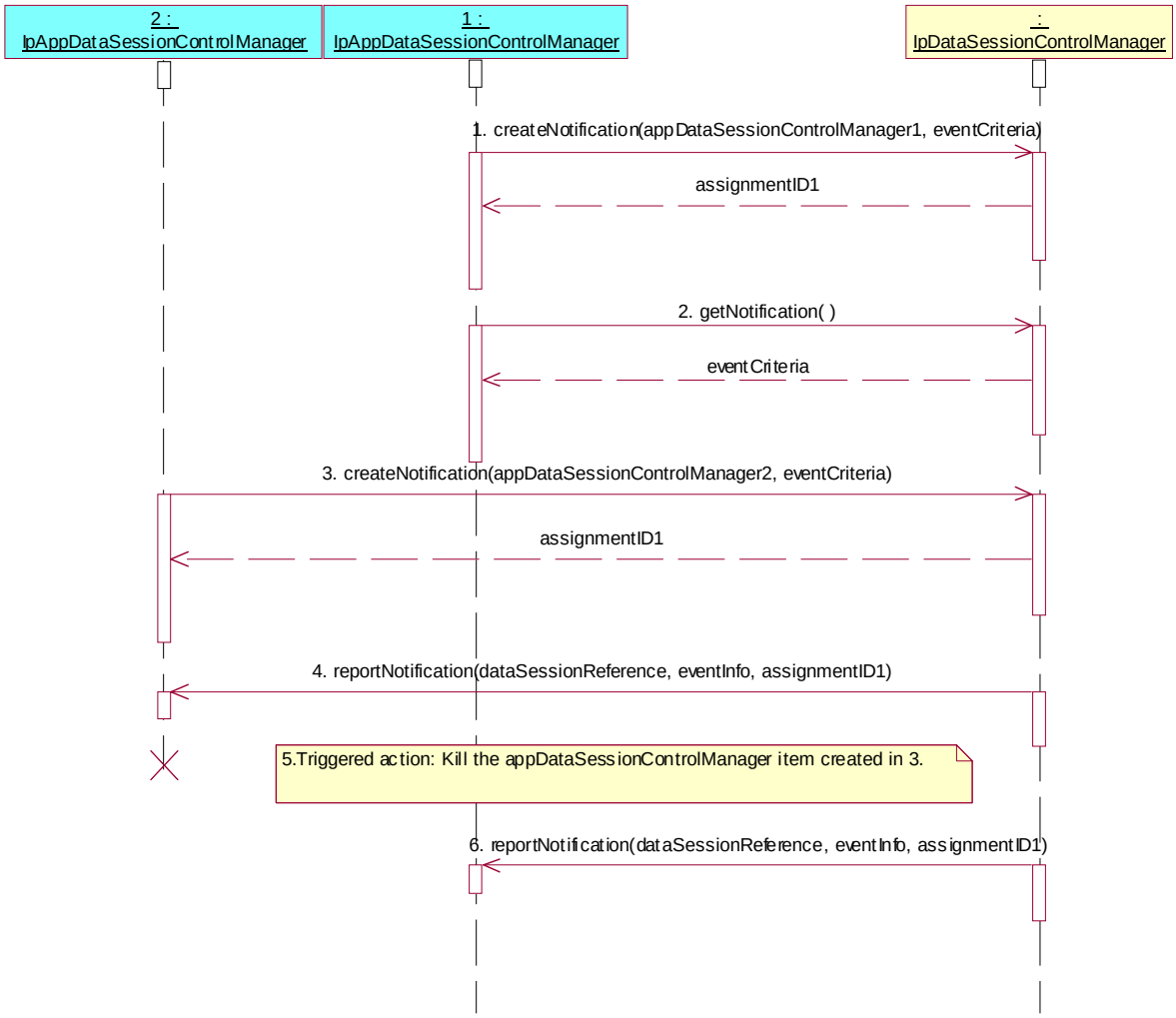
2. Method call **getNotification()**
   Parameters: none
   Check: value of TpDataSessionEventCriteria as specified in 1. is returned

3. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL) different from 1., eventCriteria
   Check: same value of TpAssignmentID as in 1. is returned

4. Triggered Action: cause IUT to call **reportNotification()** method on the tester's (application's) IpAppDataSessionControlManager interface
   Parameters: dataSessionReferencer, eventInfo, assignmentID
   Check: The interface given in 3. receives the event

5. Triggered action: Kill the appDataSessionControlManager item referenced in 3.

6. Triggered Action: cause IUT to call **reportNotification()** method on the tester's (application's) IpAppDataSessionControlManager interface.
   Parameters: dataSessionReference, eventInfo, assignmentID
   Check: The interface given in 1. receives the event.


2 :
IpAppDataSessionControlManager
1 :
IpAppDataSessionControlManager
:
IpDataSessionControlManager
1. createNotification(appDataSessionControlManager1, eventCriteria)
assignmentID1
2. getNotification( )
eventCriteria
3. createNotification(appDataSessionControlManager2, eventCriteria)
assignmentID1
4. reportNotification(dataSessionReference, eventInfo, assignmentID1)
5.Triggered action: Kill the appDataSessionControlManager item created in 3.
6. reportNotification(dataSessionReference, eventInfo, assignmentID1)

5.2.1.2 IpDataSession

Test DSC_DS_01

Summary: **IpDataSession**, release after connectReq(), successful with interrupt after reportNotification()

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Parameter values:
   eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
   eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   Check: valid value of TpAssignmentID is returned

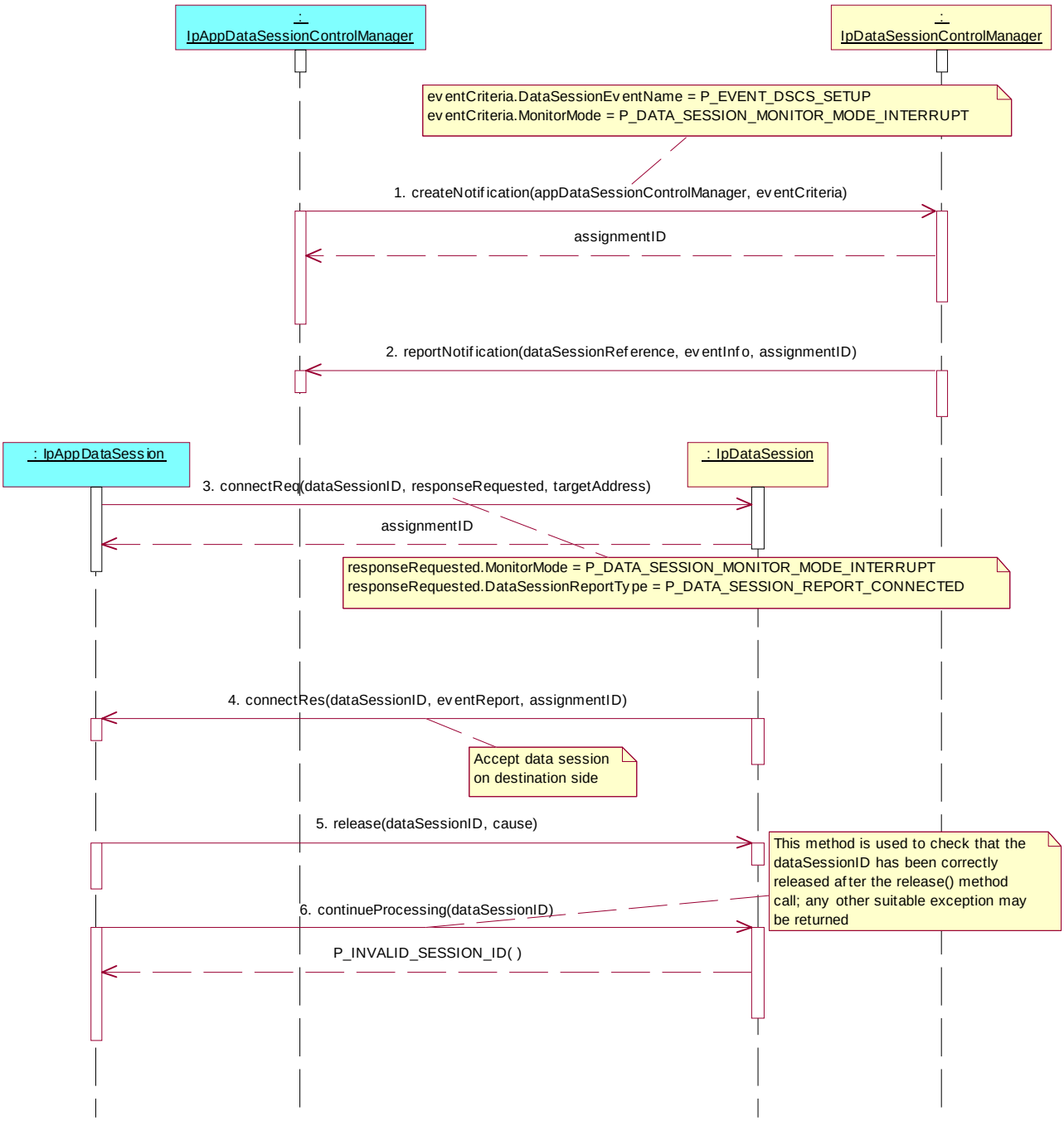
2. Triggered action: cause IUT to call **reportNotification()** method on the tester's (application's) **IpAppDataSessionControlManager** interface, i.e. start to set-up data session
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **connectReq()**
   Parameters: dataSessionID, responseRequested, targetAddress
   Parameter values:
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
   Check: valid value of TpAssignmentID is returned

4. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. accept the data session on the destination side
   Parameters: dataSessionID, eventReport, assignmentID

5. Method call **release()**
   Parameters: dataSessionID, cause
   Check: no exception is returned

6. Method call **continueProcessing()**
   Parameters: dataSessionID
   Check: P_INVALID_SESSION_ID or any suitable exception is returned


: IpAppDataSessionControlManager
: IpDataSessionControlManager
eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
1. createNotification(appDataSessionControlManager, eventCriteria)
assignmentID
2. reportNotification(dataSessionReference, eventInfo, assignmentID)
: IpAppDataSession
: IpDataSession
3. connectReq(dataSessionID, responseRequested, targetAddress)
assignmentID
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
4. connectRes(dataSessionID, eventReport, assignmentID)
Accept data session on destination side
5. release(dataSessionID, cause)
This method is used to check that the dataSessionID has been correctly released after the release() method call; any other suitable exception may be returned
6. continueProcessing(dataSessionID)
P_INVALID_SESSION_ID( )

Test DSC_DS_02

Summary: **IpDataSession**, deassignDataSession() after connectReq(), successful.

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Parameter values:
   eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
   eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   Check: valid value of TpAssignmentID is returned

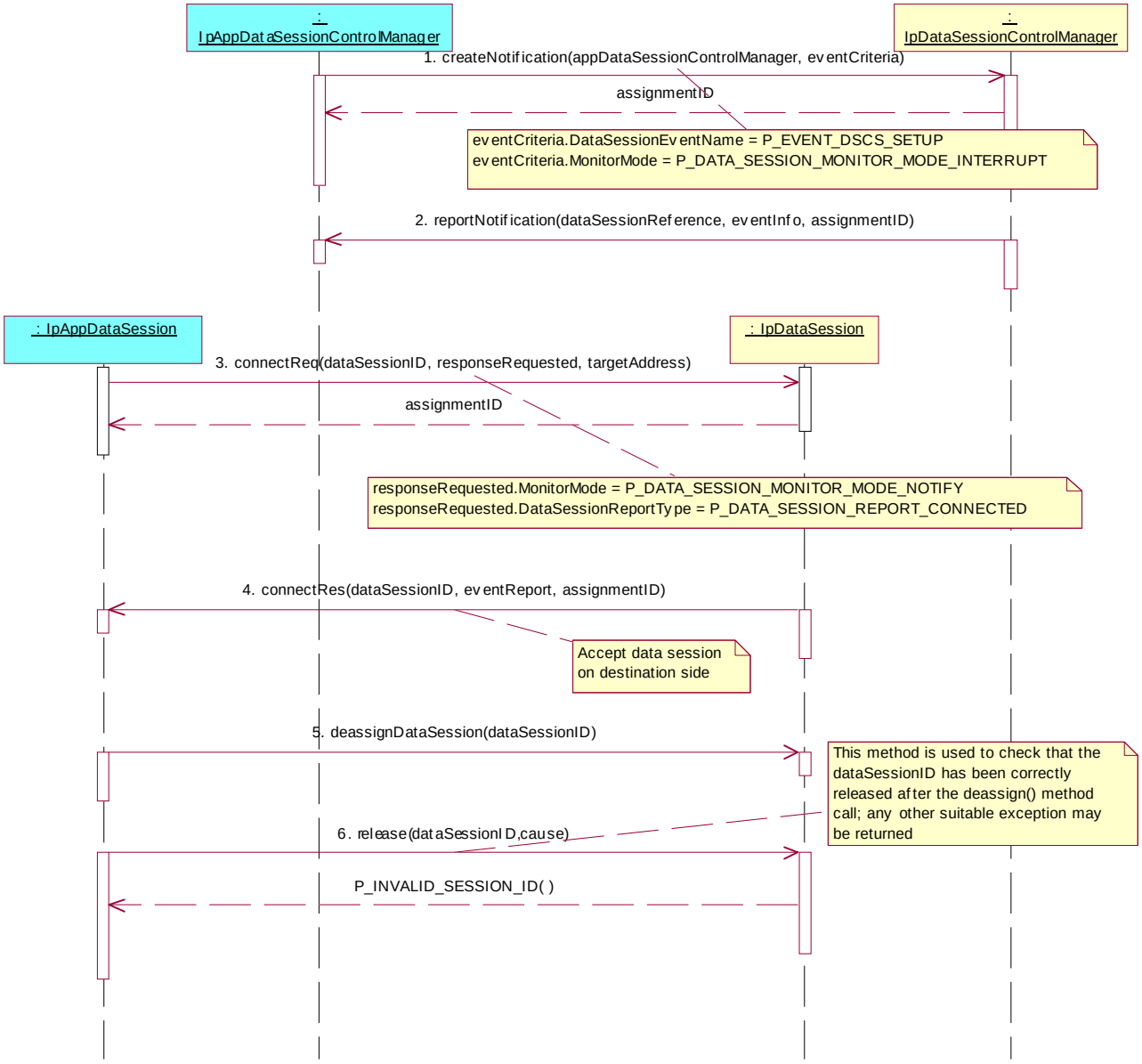
2. Triggered action: cause IUT to call **reportNotification()** method on the tester's (application's) **IpAppDataSessionControlManager** interface, i.e. start to set-up data session
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **connectReq()**
   Parameters: dataSessionID, responseRequested, targetAddress
   Parameter values:
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_NOTIFY
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
   Check: valid value of TpAssignmentID is returned

4. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. accept the data session on the destination side
   Parameters: dataSessionID, eventReport, assignmentID

5. Method call **deassignDataSession()**
   Parameters: dataSessionID
   Check: no exception is returned

6. Method call **release()**
   Parameters: dataSessionID, cause
   Check: P_INVALID_SESSION_ID or any suitable exception is returned


: IpAppDataSessionControlManager
: IpDataSessionControlManager
1. createNotification(appDataSessionControlManager, eventCriteria)
assignmentID
eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
2. reportNotification(dataSessionReference, eventInfo, assignmentID)
: IpAppDataSession
: IpDataSession
3. connectReq(dataSessionID, responseRequested, targetAddress)
assignmentID
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_NOTIFY
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
4. connectRes(dataSessionID, eventReport, assignmentID)
Accept data session on destination side
5. deassignDataSession(dataSessionID)
This method is used to check that the dataSessionID has been correctly released after the deassign() method call; any other suitable exception may be returned
6. release(dataSessionID,cause)
P_INVALID_SESSION_ID( )

Test DSC_DS_03

Summary: **IpDataSession**, connectReq() with two trigger events, successful with interrupt after reportNotification()

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Parameter values:
   eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
   eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   Check: valid value of TpAssignmentID is returned

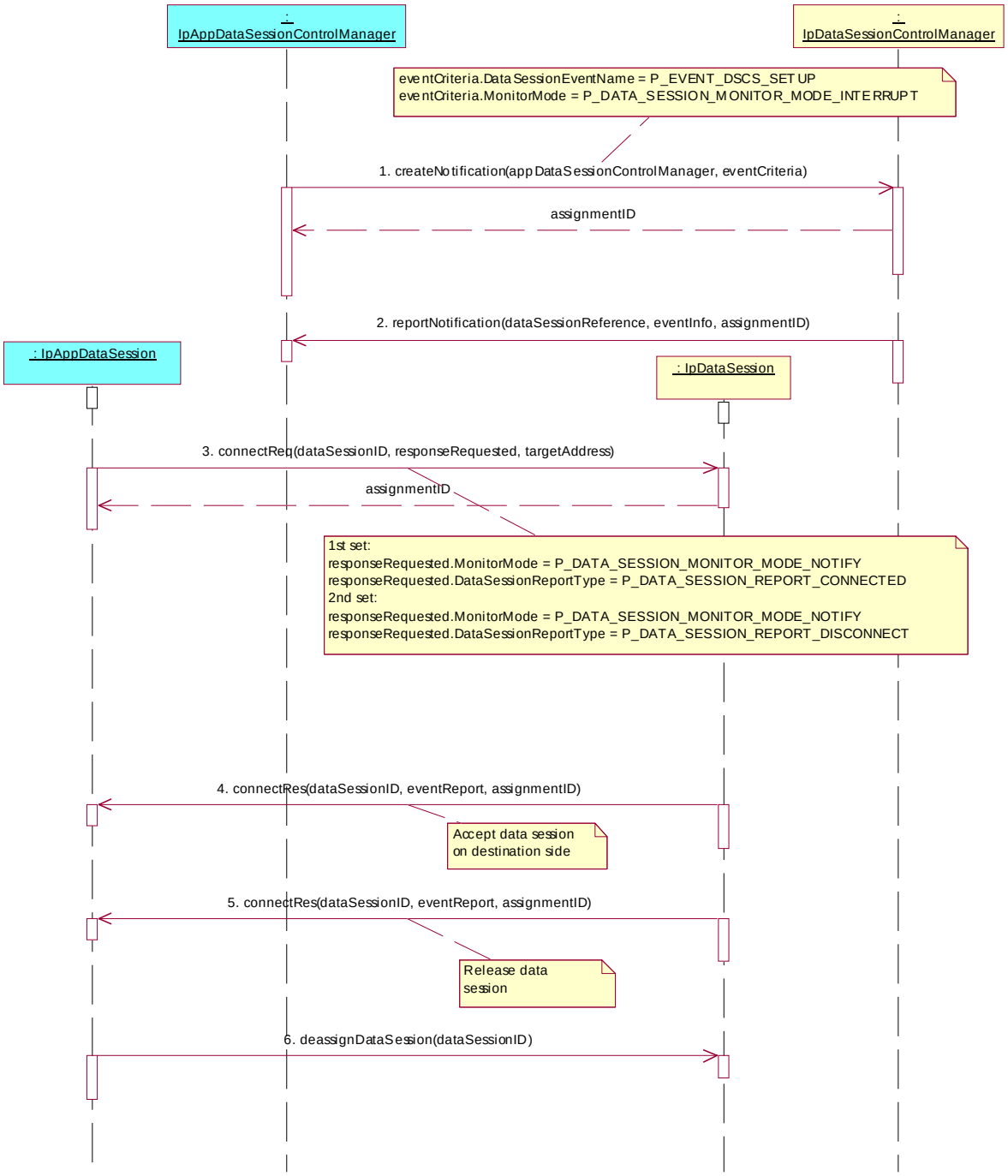
2. Triggered action: cause IUT to call **reportNotification()** method on the tester's (application's) **IpAppDataSessionControlManager** interface, i.e. start to set-up data session
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **connectReq()**
   Parameters: dataSessionID, responseRequested, targetAddress
   Parameter values:
   1st set
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_NOTIFY
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
   2nd set
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_NOTIFY
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_DISCONNECT
   Check: valid value of TpAssignmentID is returned

4. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. accept the data session on the destination side
   Parameters: dataSessionID, eventReport, assignmentID

5. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. release the data session
   Parameters: dataSessionID, eventReport, assignmentID

6. Method call **deassignDataSession()**
   Parameters: dataSessionID
   Check: no exception is returned


: IpAppDataSessionControlManager
: IpDataSessionControlManager
eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
1. createNotification(appDataSessionControlManager, eventCriteria)
assignmentID
2. reportNotification(dataSessionReference, eventInfo, assignmentID)
: IpAppDataSession
: IpDataSession
3. connectReq(dataSessionID, responseRequested, targetAddress)
assignmentID
1st set:
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_NOTIFY
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
2nd set:
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_NOTIFY
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_DISCONNECT
4. connectRes(dataSessionID, eventReport, assignmentID)
Accept data session on destination side
5. connectRes(dataSessionID, eventReport, assignmentID)
Release data session
6. deassignDataSession(dataSessionID)

Test DSC_DS_04

Summary: **IpDataSession**, superviseDataSessionReq()

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Parameter values:
   eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
   eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   Check: valid value of TpAssignmentID is returned

2. Triggered action: cause IUT to call **reportNotification()** method on the tester's (application's) **IpAppDataSessionControlManager** interface, i.e. start to set-up data session
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **superviseDataSessionReq()**
   Parameters: dataSessionID, treatment, bytes
   Check: no exception is returned

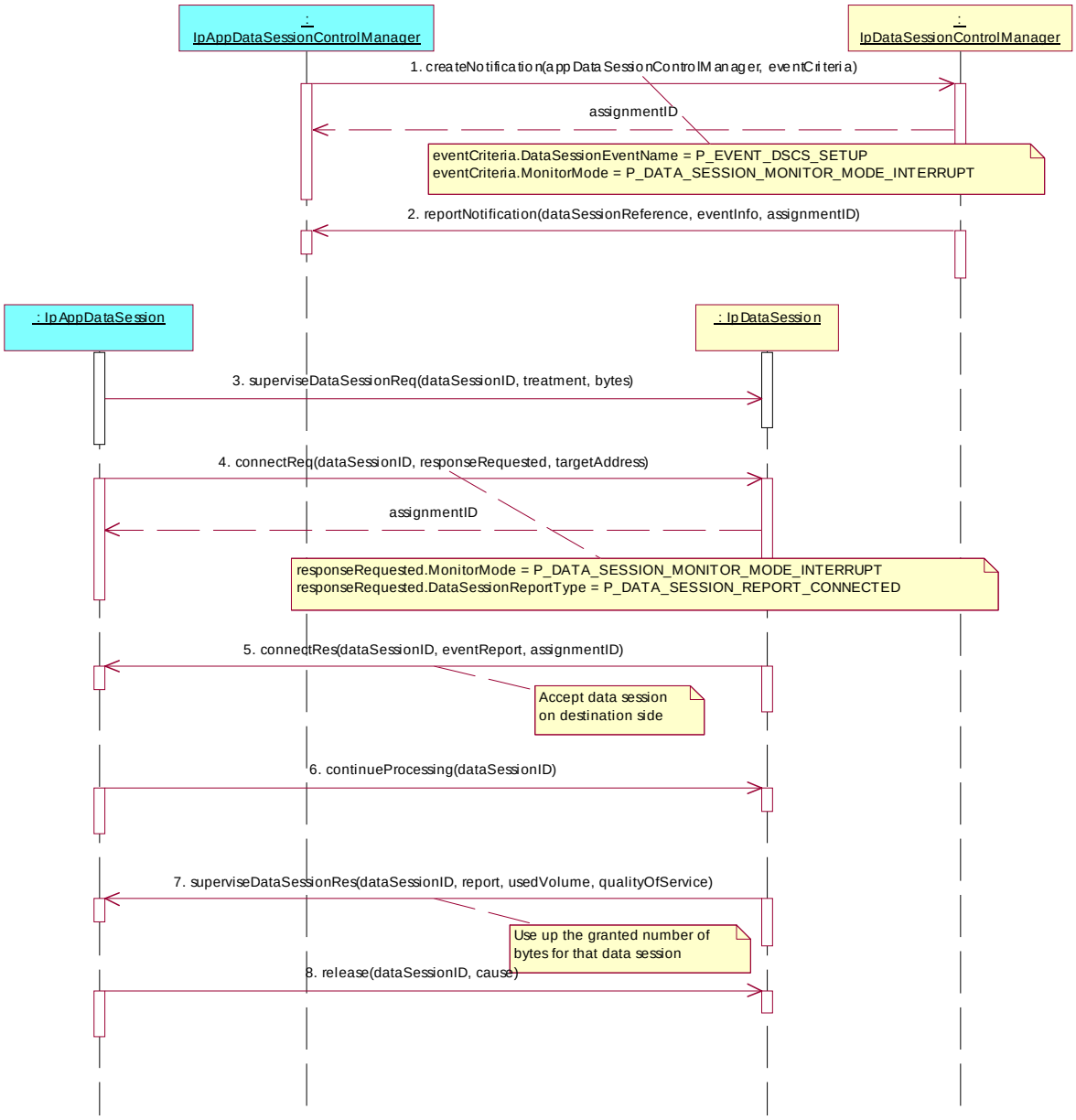
4. Method call **connectReq()**
   Parameters: dataSessionID, responseRequested, targetAddress
   Parameter values:
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
   Check: valid value of TpAssignmentID is returned

5. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. accept the data session on the destination side
   Parameters: dataSessionID, eventReport, assignmentID

6. Method call **continueProcessing()**
   Parameters: dataSessionID
   Check: no exception is returned

7. Triggered action: cause IUT to call **superviseDataSessionRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. use up the granted number of bytes for that data session
   Parameters: dataSessionID, report, usedVolume, qualityOfService

8. Method call **release()**
   Parameters: dataSessionID, cause
   Check: no exception is returned


: IpAppDataSessionControlManager
: IpDataSessionControlManager
1. createNotification(appDataSessionControlManager, eventCriteria)
assignmentID
eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
2. reportNotification(dataSessionReference, eventInfo, assignmentID)
: IpAppDataSession
: IpDataSession
3. superviseDataSessionReq(dataSessionID, treatment, bytes)
4. connectReq(dataSessionID, responseRequested, targetAddress)
assignmentID
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
5. connectRes(dataSessionID, eventReport, assignmentID)
Accept data session on destination side
6. continueProcessing(dataSessionID)
7. superviseDataSessionRes(dataSessionID, report, usedVolume, qualityOfService)
Use up the granted number of bytes for that data session
8. release(dataSessionID, cause)

Test DSC_DS_05

Summary: **IpDataSession**, setDataSessionChargePlan()

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Parameter values:
   eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
   eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   Check: valid value of TpAssignmentID is returned

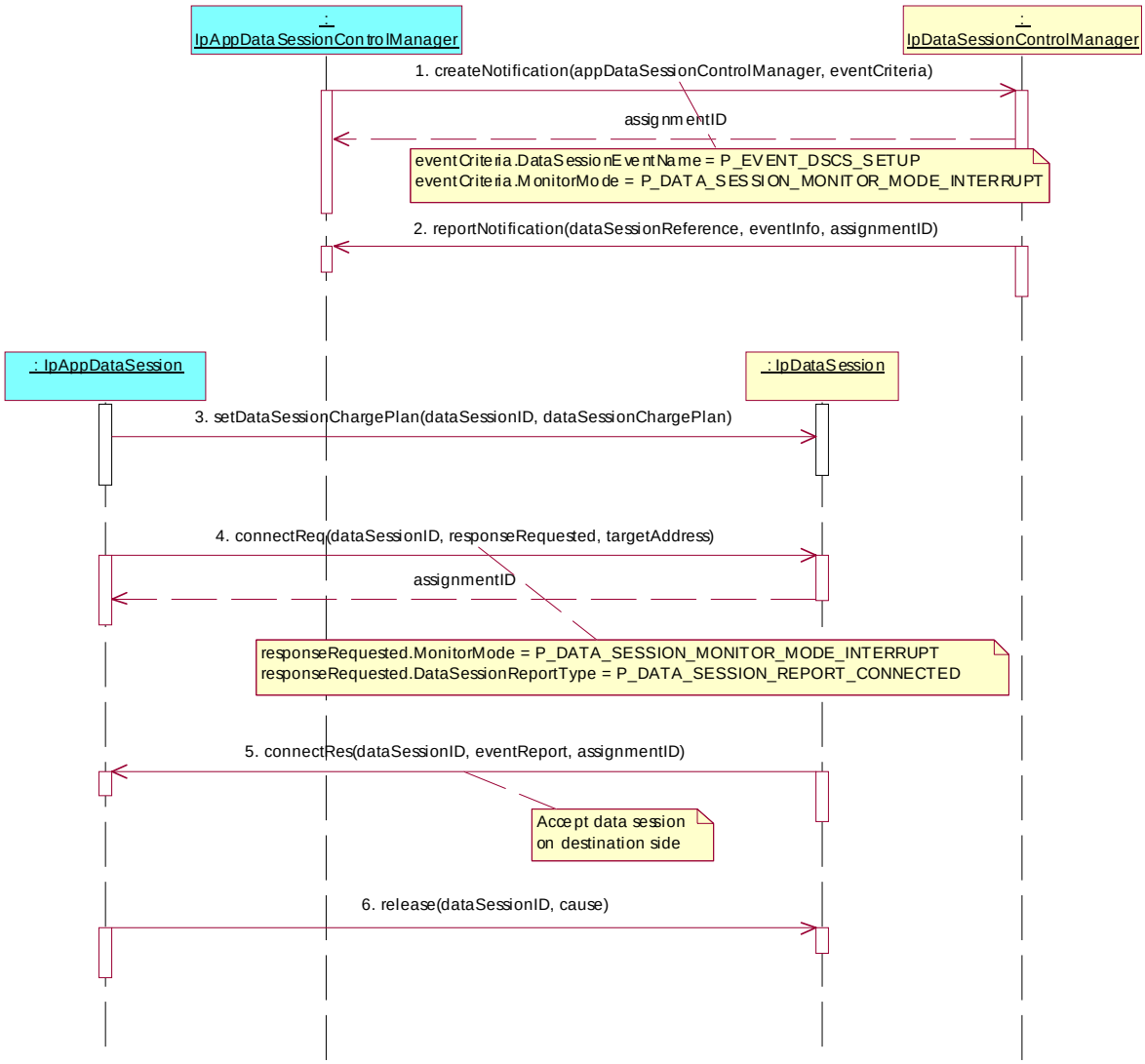
2. Triggered action: cause IUT to call **reportNotification()** method on the tester's (application's) **IpAppDataSessionControlManager** interface, i.e. start to set-up data session
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **setDataSessionChargePlan()**
   Parameters: dataSessionID, dataSessionChargePlan
   Check: no exception is returned

4. Method call **connectReq()**
   Parameters: dataSessionID, responseRequested, targetAddress
   Parameter values:
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
   Check: valid value of TpAssignmentID is returned

5. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. accept the data session on the destination side
   Parameters: dataSessionID, eventReport, assignmentID

6. Method call **release()**
   Parameters: dataSessionID, cause
   Check: no exception is returned


: IpAppDataSessionControlManager
: IpDataSessionControlManager
1. createNotification(appDataSessionControlManager, eventCriteria)
assignmentID
eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
2. reportNotification(dataSessionReference, eventInfo, assignmentID)
: IpAppDataSession
: IpDataSession
3. setDataSessionChargePlan(dataSessionID, dataSessionChargePlan)
4. connectReq(dataSessionID, responseRequested, targetAddress)
assignmentID
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
5. connectRes(dataSessionID, eventReport, assignmentID)
Accept data session
on destination side
6. release(dataSessionID, cause)

Test DSC_DS_06

Summary: **IpDataSession**, setAdviceOfCharge()

Reference: ES 201 915-8 [1], clauses 7.4.1 and 8.3

Preamble: Registration of the IUT (Data Session Control SCF) and the tester (application) to the framework. The tester must have obtained a reference to an instance of the IpDataSessionControlManager interface through selecting that service and signing the required service agreement.

Test Sequence:

1. Method call **createNotification()**
   Parameters: appDataSessionControlManager (non-NULL), eventCriteria
   Parameter values:
   eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
   eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   Check: valid value of TpAssignmentID is returned

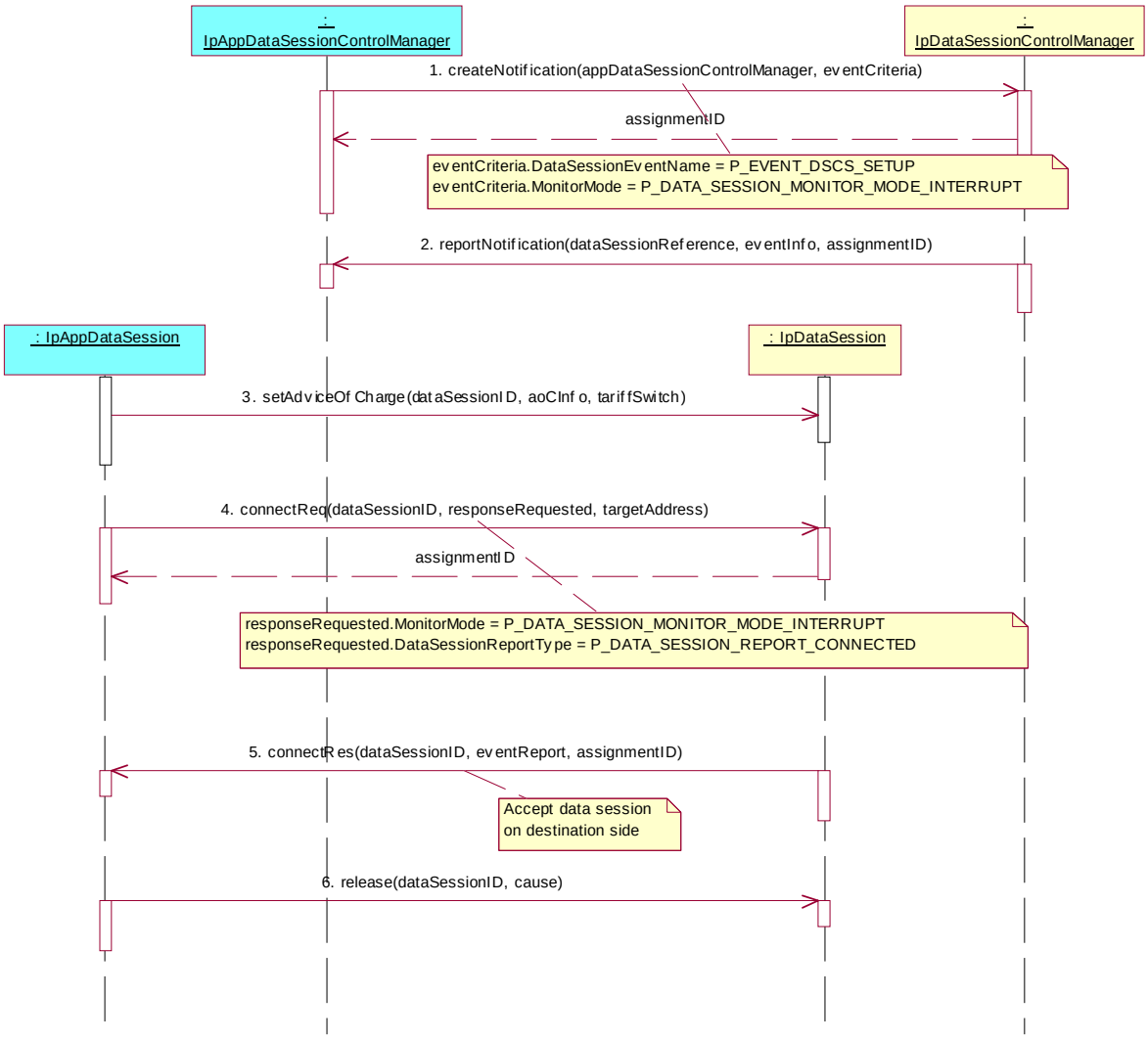
2. Triggered action: cause IUT to call **reportNotification()** method on the tester's (application's) **IpAppDataSessionControlManager** interface, i.e. start to set-up data session
   Parameters: dataSessionReference, eventInfo, assignmentID

3. Method call **setAdviceOfCharge()**
   Parameters: dataSessionID, aoCInfo, tariffswitch
   Check: no exception is returned

4. Method call **connectReq()**
   Parameters: dataSessionID, responseRequested, targetAddress
   Parameter values:
   responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
   responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
   Check: valid value of TpAssignmentID is returned

5. Triggered action: cause IUT to call **connectRes()** method on the tester's (application's) **IpAppDataSession** interface, i.e. accept the data session on the destination side
   Parameters: dataSessionID, eventReport, assignmentID

6. Method call **release()**
   Parameters: dataSessionID, cause
   Check: no exception is returned


: IpAppDataSessionControlManager
: IpDataSessionControlManager
1. createNotification(appDataSessionControlManager, eventCriteria)
assignmentID
eventCriteria.DataSessionEventName = P_EVENT_DSCS_SETUP
eventCriteria.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
2. reportNotification(dataSessionReference, eventInfo, assignmentID)
: IpAppDataSession
: IpDataSession
3. setAdviceOfCharge(dataSessionID, aoCInfo, tariffSwitch)
4. connectReq(dataSessionID, responseRequested, targetAddress)
assignmentID
responseRequested.MonitorMode = P_DATA_SESSION_MONITOR_MODE_INTERRUPT
responseRequested.DataSessionReportType = P_DATA_SESSION_REPORT_CONNECTED
5. connectRes(dataSessionID, eventReport, assignmentID)
Accept data session
on destination side
6. release(dataSessionID, cause)

History

Document history		
V1.1.1	June 2003	Membership Approval Procedure MV 20030801: 2003-06-03 to 2003-08-01
V1.1.1	August 2003	Publication