



SLOVENSKI STANDARD
SIST ES 202 915-13 V1.1.1:2005
01-januar-2005

**Odpri dostop do storitve (OSA) – Vmesnik za aplikacijsko programiranje (API) –
13. del: Upravljanje politike SCF**

Open Service Access (OSA); Application Programming Interface (API); Part 13: Policy management SCF

iTeh STANDARD PREVIEW
(standards.iteh.ai)

Ta slovenski standard je istoveten z: **ES 202 915-13 Version 1.1.1**

SIST ES 202 915-13 V1.1.1:2005
<https://standards.iteh.ai/catalog/standards/sist/ead798bb-8a0e-4d0d-90b5-9ce22c8d28e8/sist-es-202-915-13-v1-1-1-2005>

ICS:

33.040.01	Telekomunikacijski sistemi na splošno	Telecommunication systems in general
-----------	--	---

SIST ES 202 915-13 V1.1.1:2005 **en**

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[SIST ES 202 915-13 V1.1.1:2005](https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-9ce22c8d28e8/sist-es-202-915-13-v1-1-1-2005)

<https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-9ce22c8d28e8/sist-es-202-915-13-v1-1-1-2005>

ETSI ES 202 915-13 V1.1.1 (2003-01)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Part 13: Policy management SCF



iTeh STANDARD PREVIEW (standards.iteh.ai)

[SIST ES 202 915-13 V1.1.1:2005](https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-9ce22c8d28e8/sist-es-202-915-13-v1-1-1-2005)

<https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-9ce22c8d28e8/sist-es-202-915-13-v1-1-1-2005>



Reference

DES/SPAN-120091-13

Keywords

API, IDL, OSA, UML**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

iTeh STANDARD PREVIEW
(standards.iteh.ai)

SIST ES 202 915-13 V1.1.1:2005<https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-9ce22c8d28c4/etsi-es-202-915-13-v1-1-1-2005>
Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, send your comment to:

editor@etsi.org

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2003.

© The Parlay Group 2003.

All rights reserved.

DECT™, **PLUGTESTS™** and **UMTS™** are Trade Marks of ETSI registered for the benefit of its Members.
TIPHON™ and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

Contents

Intellectual Property Rights	7
Foreword.....	7
1 Scope	8
2 References	8
3 Definitions and abbreviations.....	8
3.1 Definitions	8
3.2 Abbreviations	8
4 Policy Management SCF.....	9
5 Sequence Diagrams	9
5.1 Use of Policy Repository.....	9
5.2 Introduce condition and action into rule.....	11
5.3 Create and receive an event.....	13
5.4 Create and modify domain	15
5.5 ASP offering services to prepaid subscribers	17
6 Class Diagrams.....	20
7 The Service Interface Specifications	22
7.1 Interface Specification Format	22
7.1.1 Interface Class	22
7.1.2 Method descriptions.....	22
7.1.3 Parameter descriptions.....	22
7.1.4 State Model.....	22
7.2 Base Interface.....	22
7.2.1 Interface Class IpInterface	22
7.3 Service Interfaces	23
7.3.1 Overview	23
7.4 Generic Service Interface	23
7.4.1 Interface Class IpService	23
7.4.1.1 Method setCallback().....	23
7.4.1.2 Method setCallbackWithSessionID().....	23
8 Policy Management Interface Classes.....	24
8.1 Interface Class IpPolicyManager.....	24
8.1.1 Method createDomain()	25
8.1.2 Method getDomain().....	26
8.1.3 Method removeDomain()	26
8.1.4 Method getDomainCount().....	26
8.1.5 Method getDomainIterator().....	27
8.1.6 Method findMatchingDomains()	27
8.1.7 Method createRepository()	27
8.1.8 Method getRepository()	28
8.1.9 Method removeRepository().....	28
8.1.10 Method getRepositoryCount()	28
8.1.11 Method getRepositoryIterator()	29
8.1.12 Method startTransaction().....	29
8.1.13 Method commitTransaction()	30
8.1.14 Method abortTransaction()	30
8.2 Interface Class IpPolicy.....	30
8.2.1 Attributes	31
8.2.2 Method getAttribute().....	32
8.2.3 Method setAttribute()	32
8.2.4 Method getAttributes()	32
8.2.5 Method setAttributes().....	33

8.3	Interface Class IpPolicyDomain	33
8.3.1	Attributes	35
8.3.2	Method getParentDomain()	36
8.3.3	Method createDomain()	36
8.3.4	Method getDomain()	36
8.3.5	Method removeDomain()	37
8.3.6	Method getDomainCount()	37
8.3.7	Method getDomainIterator()	37
8.3.8	Method createGroup()	38
8.3.9	Method getGroup()	38
8.3.10	Method removeGroup()	39
8.3.11	Method getGroupCount()	39
8.3.12	Method getGroupIterator()	39
8.3.13	Method createRule()	40
8.3.14	Method getRule()	40
8.3.15	Method removeRule()	40
8.3.16	Method getRuleCount()	41
8.3.17	Method getRuleIterator()	41
8.3.18	Method createEventDefinition()	41
8.3.19	Method getEventDefinition()	42
8.3.20	Method removeEventDefinition()	42
8.3.21	Method getEventDefinitionCount()	43
8.3.22	Method getEventDefinitionIterator()	43
8.3.23	Method generateEvent()	43
8.3.24	Method createNotification()	44
8.3.25	Method destroyNotification()	44
8.3.26	Method createVariableSet()	45
8.3.27	Method getVariableSet()	45
8.3.28	Method removeVariableSet()	45
8.3.29	Method getVariableSetCount()	46
8.3.30	Method getVariableSetIterator()	46
8.3.31	Method setVariable()	46
8.3.32	Method getVariable()	47
8.4	Interface Class IpPolicyGroup	47
8.4.1	Attributes	48
8.4.2	Method getParentDomain()	49
8.4.3	Method getParentGroup()	49
8.4.4	Method createGroup()	50
8.4.5	Method getGroup()	50
8.4.6	Method removeGroup()	50
8.4.7	Method getGroupCount()	51
8.4.8	Method getGroupIterator()	51
8.4.9	Method createRule()	51
8.4.10	Method getRule()	52
8.4.11	Method removeRule()	52
8.4.12	Method getRuleCount()	52
8.4.13	Method getRuleIterator()	53
8.5	Interface Class IpPolicyRepository	53
8.5.1	Attributes	54
8.5.2	Method getParentRepository()	55
8.5.3	Method createRepository()	55
8.5.4	Method getRepository()	55
8.5.5	Method removeRepository()	56
8.5.6	Method getRepositoryCount()	56
8.5.7	Method getRepositoryIterator()	56
8.5.8	Method createCondition()	57
8.5.9	Method getCondition()	57
8.5.10	Method removeCondition()	58
8.5.11	Method getConditionCount()	58
8.5.12	Method getConditionIterator()	58
8.5.13	Method createAction()	59
8.5.14	Method getAction()	59

8.5.15	Method removeAction()	60
8.5.16	Method getActionCount()	60
8.5.17	Method getActionIterator()	60
8.6	Interface Class IpPolicyRule	61
8.6.1	Attributes	63
8.6.2	Method getParentGroup()	64
8.6.3	Method getParentDomain()	65
8.6.4	Method createCondition()	65
8.6.5	Method getCondition()	66
8.6.6	Method removeCondition()	66
8.6.7	Method getConditionCount()	66
8.6.8	Method getConditionIterator()	67
8.6.9	Method createAction()	67
8.6.10	Method getAction()	68
8.6.11	Method removeAction()	68
8.6.12	Method getActionCount()	68
8.6.13	Method getActionIterator()	69
8.6.14	Method setValidityPeriodConditionByName()	69
8.6.15	Method setValidityPeriodCondition()	69
8.6.16	Method getValidityPeriodCondition()	70
8.6.17	Method unsetValidityPeriodCondition()	70
8.6.18	Method setConditionList()	70
8.6.19	Method getConditionList()	71
8.6.20	Method setActionList()	71
8.6.21	Method getActionList()	71
8.7	Interface Class IpPolicyCondition	72
8.7.1	Attributes	73
8.7.2	Method getParentRepository()	74
8.7.3	Method getParentRule()	74
8.8	Interface Class IpPolicyTimePeriodCondition	74
8.8.1	Attributes	75
8.9	Interface Class IpPolicyAction	78
8.9.1	Attributes	79
8.9.2	Method getParentRepository()	79
8.9.3	Method getParentRule()	80
8.10	Interface Class IpPolicyEventDefinition	80
8.10.1	Attributes	80
8.10.2	Method setRequiredAttributes()	81
8.10.3	Method setOptionalAttributes()	81
8.10.4	Method getRequiredAttributes()	82
8.10.5	Method getOptionalAttributes()	82
8.10.6	Method getParentDomain()	82
8.11	Interface Class IpPolicyEventCondition	83
8.11.1	Attributes	83
8.12	Interface Class IpPolicyExpressionCondition	84
8.12.1	Attributes	84
8.13	Interface Class IpPolicyEventAction	85
8.13.1	Attributes	85
8.14	Interface Class IpPolicyExpressionAction	86
8.14.1	Attributes	87
8.15	Interface Class IpPolicyIterator	88
8.15.1	Attributes	88
8.15.2	Method getList()	89
8.16	Interface Class IpAppPolicyDomain	89
8.16.1	Method reportNotification()	89
9	State Transition Diagrams	90
10	Data Definitions	90
10.1	Policy Management Data Definitions	90
10.1.1	TpPolicyConditionListType	90
10.1.2	TpPolicyConditionListElement	90

10.1.3	TpPolicyConditionList.....	90
10.1.4	TpPolicyConditionType.....	90
10.1.5	TpPolicyActionListElement	91
10.1.6	TpPolicyActionList.....	91
10.1.7	TpPolicyActionType.....	91
10.1.8	TpPolicyEvent	91
10.1.9	TpPolicyKeyword.....	91
10.1.10	TpPolicyKeywordSet.....	92
10.1.11	IpPolicyDomain	92
10.1.12	IpPolicyDomainRef	92
10.1.13	IpPolicyRepository	93
10.1.14	IpPolicyRepositoryRef.....	93
10.1.15	IpPolicyGroup.....	93
10.1.16	IpPolicyGroupRef.....	93
10.1.17	IpPolicyRule	93
10.1.18	IpPolicyRuleRef	93
10.1.19	IpPolicyEventDefinition	93
10.1.20	IpPolicyEventDefinitionRef	93
10.1.21	IpAppPolicyDomain	93
10.1.22	IpAppPolicyDomainRef	93
10.1.23	IpPolicyCondition.....	93
10.1.24	IpPolicyConditionRef.....	93
10.1.25	IpPolicyTimePeriodCondition	93
10.1.26	IpPolicyTimePeriodConditionRef	94
11	Policy Management Exception Classes.....	94
Annex A (normative):	OMG IDL Description of Policy Management SCF	95
Annex B (informative):	Contents of 3GPP OSA R5 Policy Management	96
Annex C (informative):	Record of changes.....	97
C.1	Interfaces	97
C.1.1	New	97
C.1.2	Deprecated.....	97
C.1.3	Removed.....	97
C.2	Methods	97
C.2.1	New	97
C.2.2	Deprecated.....	97
C.2.3	Modified.....	98
C.2.4	Removed.....	98
C.3	Data Definitions	98
C.3.1	New	98
C.3.2	Modified.....	98
C.3.3	Removed.....	98
C.4	Service Properties.....	98
C.4.1	New	98
C.4.2	Deprecated.....	99
C.4.3	Modified.....	99
C.4.4	Removed.....	99
C.5	Exceptions	99
C.5.1	New	99
C.5.2	Modified.....	99
C.5.3	Removed.....	99
C.6	Others	99
History	100

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>).

All published ETSI deliverables shall include information which directs the reader to the above source of information.

Foreword

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

The present document is part 13 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 202 915) is structured in the following parts:

- Part 1: "Overview";
- Part 2: "Common Data Definitions";
- Part 3: "Framework";
- Part 4: "Call Control";
- Part 5: "User Interaction SCF";
- Part 6: "Mobility SCF";
- Part 7: "Terminal Capabilities SCF";
- Part 8: "Data Session Control SCF";
- Part 9: "Generic Messaging SCF";
- Part 10: "Connectivity Manager SCF";
- Part 11: "Account Management SCF";
- Part 12: "Charging SCF";
- Part 13: "Policy Management SCF";**
- Part 14: "Presence and Availability Management SCF".

The present document has been defined jointly between ETSI, The Parlay Group (<http://www.parlay.org>) and the 3GPP, in co-operation with a number of JAIN™ Community (<http://www.java.sun.com/products/jain>) member companies.

The present document forms part of the Parlay 4.0 set of specifications.

The present document is equivalent to 3GPP TS 29.198-13 V5.1.0 (Release 5).

1 Scope

The present document is part 13 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs.

The present document specifies the Policy Management Service Capability Feature (SCF) aspects of the interface. All aspects of the Policy Management SCF are defined here, these being:

- Sequence Diagrams
- Class Diagrams
- Interface specification plus detailed method descriptions
- State Transition diagrams
- Data Definitions
- IDL Description of the interfaces

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

iTeh STANDARD PREVIEW

2 References (standards.iteh.ai)

The references listed in clause 2 of ES 202 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

<https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-ec33c2670961/es-202-915-13-v1-1-1-2005>

ETSI ES 202 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview".

ETSI ES 202 915-2: "Open Service Access (OSA); Application Programming Interface (API); Part 2: Common Data Definitions".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 202 915-1 apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 202 915-1 apply.

4 Policy Management SCF

It is expected that more and more OSA services will use policies to express operational criteria. It is also expected that network providers will host policy-enabled services that have been written by 3rd party application service providers. In order to manage policy information and control access to it a policy management service is needed. Consistent with this, a policy management service interface manager, IpPolicyManager, has been defined. All policy management interfaces are accessible from IpPolicyManager.

A number of APIs have been defined to obtain services from a policy management service. These include APIs to create, update or view policy information. Additionally APIs have been defined to facilitate interactions between clients (e.g. a 3rd party application) and any policy enabled service. These include APIs to view policy events, to subscribe to policy events and for the generation of events by clients. All APIs conform to an underlying policy information model.

Clients that perform administrative tasks, e.g. create, update or delete policy information must obtain access to IpPolicyManager using the family of obtainInterface() methods supported by the IpAccess interface. Administrative tasks may be performed through methods supported by IpPolicyManager.

Clients that need to interact with a specific policy enabled service (for non-administrative tasks) can obtain access to that service's interface directly via the selectService() method supported by the IpAccess interface. It should be noted that specific policy enabled services may support additional interfaces and methods that are not defined below. Examples of policy enabled services include: A load balancing service that uses policies to manage application loads on the network, a charging service that determines charging criteria based on policies, a call management service that uses policies to direct end-user calls to appropriate call agents, etc.

The order is as follows:

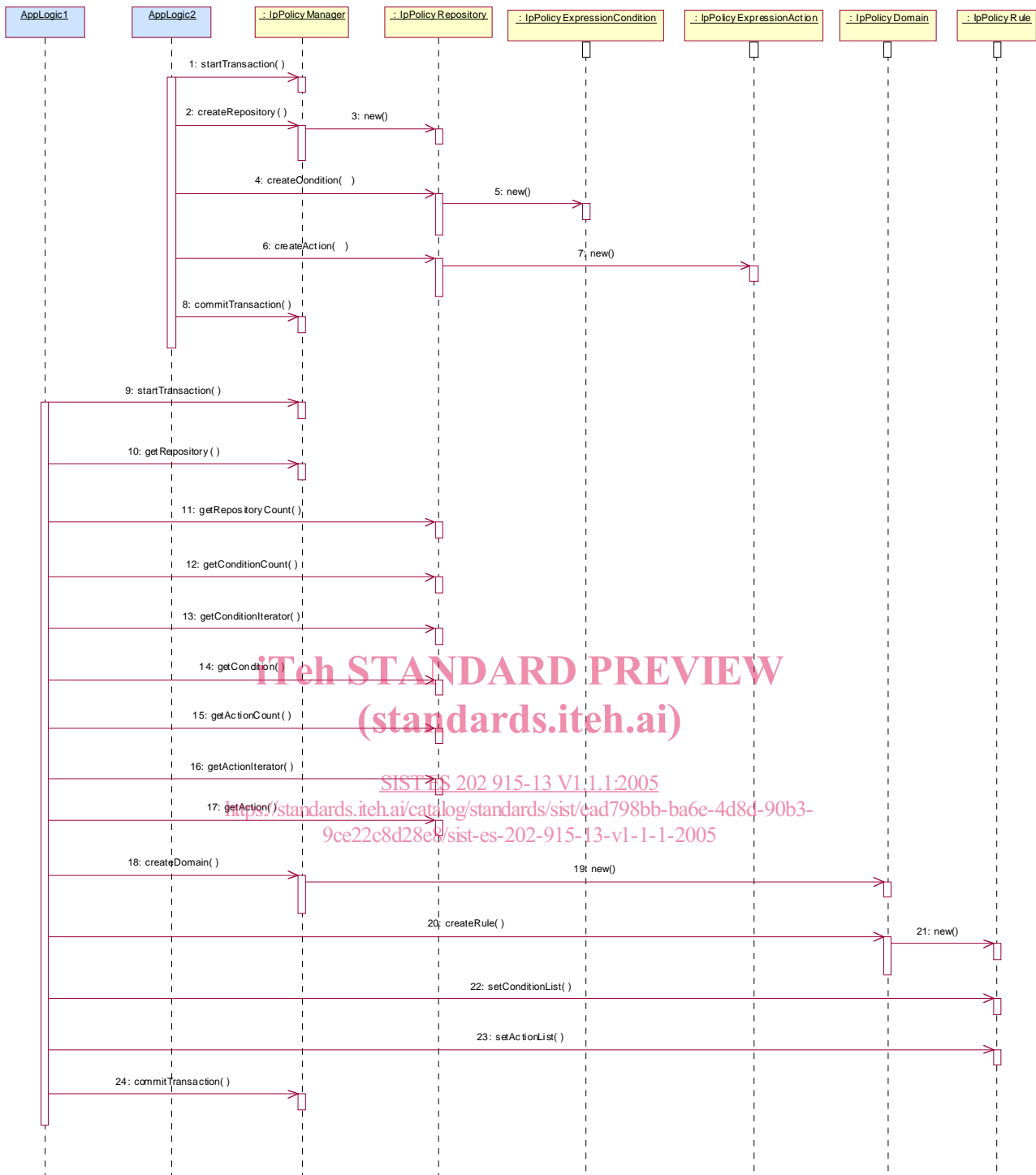
- The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.
- The Class relationships clause show how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.
- The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data Definitions clause shows a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part ES 202 915-2.

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method. Where a method is not supported by an implementation of a Service interface, the exception P_METHOD_NOT_SUPPORTED shall be returned to any call of that method.

5 Sequence Diagrams

5.1 Use of Policy Repository

The example shown here shows the use of a Policy Repository. The repository is meant to hold unattached conditions and actions. The Network Operator can populate the repository with the conditions and actions that it can support. These may indeed be based on 'off-line' negotiations with the application developer. The application developer uses the conditions and actions in the Policy Repository to create rules for his own application. In the example application logic represented by AppLogic1 belongs to the Network Operator, whereas the application logic represented by AppLogic2 belongs to the ASP. This example uses the same conditions, actions, and rules as the ASP example.



- 1: The creation of the repository by the Network Operator takes place within one transaction.
- 2: The method createRepository is invoked on the IpPolicyManager interface to create a new repository.
- 3: As a result of the createRepository method a new instance of the IpPolicyRepository interface is created. Its interface reference is returned as return parameter of the createRepository method.
- 4: The Network Operator creates an unattached condition in the new repository by invoking the createCondition method. For simplicity reasons, this is the same condition as in sequence 8 of the ASP example. The same condition attributes apply.
- 5: A new instance of the IpPolicyExpressionCondition interface is created.

- 6: The Network Operator creates an unattached action in the repository. Again, this is the same action as in sequence 10 of the ASP example. The same action attributes apply.
- 7: A new instance of the IpPolicyExpressionAction interface is created.
- 8: The Network Operator is finished with creating and populating the repository and closes the transaction.
- 9: Now that a repository exists, the ASP application can open a transaction to start creating a rule based on the conditions and actions stored in the repository.
- 10: The application invokes the getRepository to obtain a reference to the top-level repository. The returned reference in this case is the reference to the new repository just created by the Network Operator.
- 11: The application can invoke the getRepositoryCount method on the IpPolicyRepository interface to check whether there are any sub-repositories available. This is not the case for this example.
- 12: Before trying to obtain all available conditions in this repository the application retrieves the number of conditions by invoking the method getConditionCount.
- 13: The application can now invoke the getConditioniterator method to obtain the reference to an iterator that contains the names of each of the conditions contained by this repository that the application is authorized to see. As the previous method only return one available condition, this would be only one name, i.e. "SufficientCredit".
- 14: A reference to the condition can be obtained by invoking getCondition, with the condition name from the iterator as input parameter.
- 15: Similar to 12.
- 16: Similar to 13.
- 17: Similar to 14.
- 18: At this point in time the application has the names and references to the unattached condition and action from the repository it wants to use to create the rule. First a domain is created by invoking the createDomain method on the IpPolicyManager interface.
- 19: A new instance of the IpPolicyDomain interface is created.
- 20: The application invokes createRule to create a rule within the domain that was just created in flow 18 and 19.
- 21: A new instance of the IpPolicyRule interface is created.
- 22: By invoking the method setConditionList, the application can now apply the condition from the repository to this rule, by passing the condition reference, obtained by getCondition in flow 14, as an input parameter.
- 23: Similarly the application can apply the action to the rule by invoking setActionList.
- 24: Finally, once the rule is created using the condition and action from the policy repository, the transaction can be closed.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

<https://standards.iteh.ai/catalog/standards/sist/ead798bb-ba6e-4d8d-90b3-90c22c8d28e8/sist-es-202-915-13-v1-1-1-2005>

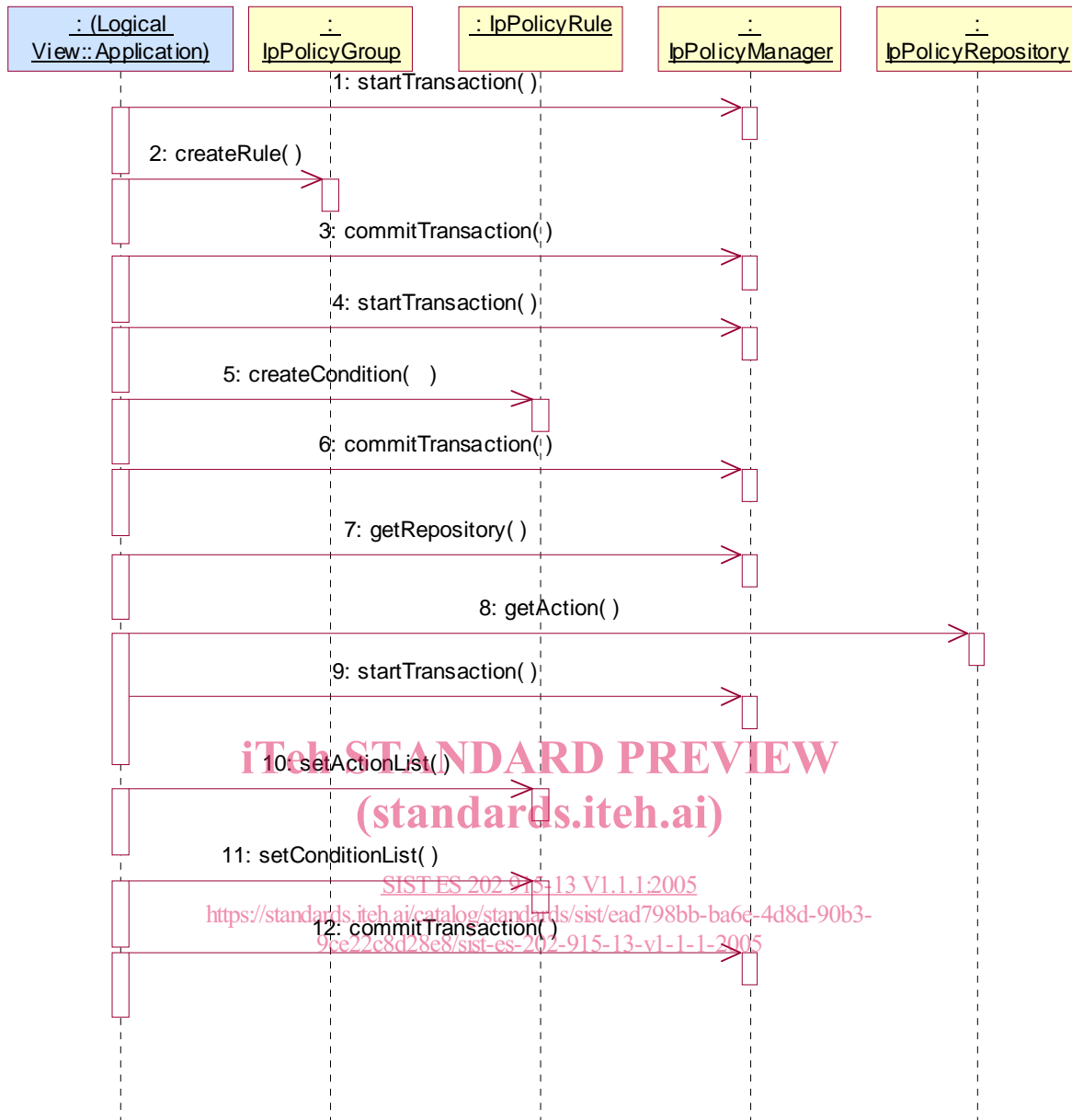
5.2 Introduce condition and action into rule

This sequence diagram describes how a specific policy rule is managed. A rule consists generally of conditions and of actions, the latter being evaluated if all conditions evaluate to true.

This sequence includes:

- creation of a condition and introduction of it into the rule;
- retrieval of an already defined action object from a repository and introduction into the rule;
- establishing a transaction bracket.

Presumption: the Application got a reference to the group, e.g. by having performed the sequence "create&modify" domain.



1: Opens the transaction bracket.

2: creates a rule object in the group by passing the name as parameter. The method returns the reference to the new rule object.

3: Closes the transaction bracket.

4: Opens the transaction bracket.

5: After having created the rule object one can "fill" it with actions and conditions. Here a condition is created on the rule object, thus becoming a part of the rule. Conditions defined in such a way cannot be reused in other rules. For this the repository approach should be used.

Parameters passed are the condition name and the condition type.

Returns a reference to this condition object.

As preliminary to the invocation of "createCondition", the application should perform the following activities:

1) Create a TpAttribute, with AttributeName: "Expression", AttributeType: P_STRING, AttributeValue:

"<the condition expression to be evaluated>"

2) Add the TpAttribute from 1) to a new TpAttributeSet as its sole element

After having performed these steps the application can call the method createCondition() on the appropriate repository or rule, passing in the name of the condition, the type of the condition IpPolicyExpressionCondition, and the TpAttributeSet created in 2). Note that this call may throw an exception if the expression defined in 1) is not parsable according to the published BNF.

Creating IpPolicyExpressionAction is done similarly.

6: Closes the transaction bracket.

7: Now we're using the repository approach, i.e. reusable condition or action objects. In this example we reuse an action.

For that purpose we ask at the IpPolicyManager interface for a reference to a named repository.

The repository name is passed.

Returns the reference to the repository.

8: If we know already the name of the action object one retrieves the action directly by passing the name as parameter. Otherwise one has to retrieve the name first by using an action iterator.

Returns a reference to the action object.

9: Opens the transaction bracket.

10: Now, the action(s) must be assigned to the rule. Furthermore and different to the conditions, one has to assign an ordering number to the action.

Passed parameter is the action list, which is a list of action reference/ sequence pairs.

11: After having created or retrieved all needed conditions they must be assigned to the rule. This is done by passing the list of condition to that method.

This is explicitly done by passing a TpPolicyConditionList again consisting of TpPolicyConditionListElements which contains the reference the IpPolicyCondition object created with message 2.1-2005

If the rule is active, this will then cause the expression defined in the condition to be evaluated (as often as necessary). Note that the binding between the variables referenced in the expression and the instances of the variable available is done each time the expression is evaluated. That is, when evaluating a variable reference, each enclosing domain is searched in order (from closest to farthest) for a matching variable. If one is found, it is used. If no matching variable is set, the expression condition fails (evaluates to FALSE).

Activation of actions is done similarly.

12: Closes the transaction bracket.

5.3 Create and receive an event

This sequence shows how policy events are used.

For clarification we list the different policy related objects used:

- IpPolicyEventDefinition: The "template" used to define allowable events. The template is used to define formally a distinct type of rule condition and rule action, namely, IpPolicyEventCondition and IpPolicyEventAction.
- IpPolicyEventCondition: A special instance of a policy condition used in a rule. The condition evaluates to "True" on the occurrence of the event instance that is formally associated with it.
- IpPolicyEventAction: A special instance of a policy action used in a rule. The action results in the generation of an instance of the formal event associated with it.
- TpPolicyEvent: This data type is passed as a parameter in the formal notification (to a client) of the occurrence of an instance of an event.

Presumption: the reference to a rule has been somehow retrieved.