

# API for Admission Control Plug-In (ACP) Reference Manual

6.3

Generated by Doxygen 1.3.7

Fri Mar 3 10:40:50 2006



# Contents

<b>1 Admission Control Plug-In (ACP) API</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Creating the Application . . . . .	1
<b>2 API for Admission Control Plug-In (ACP) Namespace Index</b>	<b>3</b>
2.1 API for Admission Control Plug-In (ACP) Package List . . . . .	3
<b>3 API for Admission Control Plug-In (ACP) Class Index</b>	<b>5</b>
3.1 API for Admission Control Plug-In (ACP) Class List . . . . .	5
<b>4 API for Admission Control Plug-In (ACP) File Index</b>	<b>7</b>
4.1 API for Admission Control Plug-In (ACP) File List . . . . .	7
<b>5 API for Admission Control Plug-In (ACP) Namespace Documentation</b>	<b>9</b>
5.1 Package acpAdapter . . . . .	9
<b>6 API for Admission Control Plug-In (ACP) Class Documentation</b>	<b>13</b>
6.1 acpAdapter.AttributeUnion Union Reference . . . . .	13
6.2 acpAdapter.RemoteUpdateInterface Interface Reference . . . . .	15
6.3 acpAdapter.TagValue Struct Reference . . . . .	16
<b>7 API for Admission Control Plug-In (ACP) File Documentation</b>	<b>17</b>
7.1 acpPlugin.idl File Reference . . . . .	17



# Chapter 1

## Admission Control Plug-In (ACP) API

### 1.1 Introduction

The ACP application programming interface (API) is defined by CORBA IDL. The use of CORBA allows for language-independent implementation as well as location-independent deployment.

The ACP API allows dynamic updates of volatile information, that do not reflect static changes, such as the train rate of a DSL modem in a noisy environment. The updates are fast and immediate, but the changes are not persistent.

### 1.2 Creating the Application

You can develop your own application to update information about subscribers and congestion points for ACP. The application can call the `update()` method to interact with ACP. The `update()` method takes a property-value pair and passes the information to ACP.

To create an application that updates ACP remotely:

1. Compile the IDL file, and generate the code in the language in which you want to write the application.
2. Write the application, and include the generated code for the IDL file.
3. Use the CORBA object reference defined in the property `ACP.syncRateAdaptor.ior` to send data from the application to ACP.

For more information about configuring ACP, see the *SDX Application Library Guide*.



## Chapter 2

# API for Admission Control Plug-In (ACP) Namespace Index

### 2.1 API for Admission Control Plug-In (ACP) Package List

Here are the packages with brief descriptions (if available):

[acpAdapter](#) (CORBA-based plug-in API that allows you to pass information  
to ACP in any language that supports CORBA) . . . . . 9



## Chapter 3

# API for Admission Control Plug-In (ACP) Class Index

### 3.1 API for Admission Control Plug-In (ACP) Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">acpAdapter.AttributeUnion</a> (Structure used to pass attribute types to ACP) . .	13
<a href="#">acpAdapter.RemoteUpdateInterface</a> (Remote update interface) . . . . .	15
<a href="#">acpAdapter.TagValue</a> (List of attributes in <a href="#">TagValue</a> format) . . . . .	16



## Chapter 4

# API for Admission Control Plug-In (ACP) File Index

### 4.1 API for Admission Control Plug-In (ACP) File List

Here is a list of all files with brief descriptions:

<a href="#">acpPlugin.idl</a> (API that you can use to implement the Admission Control Plug-In (ACP) ) . . . . .	17
---	----



## Chapter 5

# API for Admission Control Plug-In (ACP) Namespace Documentation

### 5.1 Package `acpAdapter`

#### 5.1.1 Detailed Description

CORBA-based plug-in API that allows you to pass information to ACP in any language that supports CORBA.

#### Classes

- union `acpAdapter.AttributeUnion`  
*Structure used to pass attribute types to ACP.*
- struct `acpAdapter.TagValue`  
*List of attributes in `TagValue` format.*
- interface `acpAdapter.RemoteUpdateInterface`  
*Remote update interface.*

#### Typedefs

- typedef sequence< `TagValue` > `TagValueList`

Attributes are passed to the `update()` method in the form of a *TagValue* list.

## Enumerations

- enum `RemoteUpdateType` {  
    `PE_USER`,  
    `PE_CONGESTIONPOINT` }  
*Types of information that the application is updating.*
- enum `CongestionPointState` {  
    `CP_UP`,  
    `CP_DOWN` }  
*Congestion point state indicates whether the interface is up or down.*
- enum `RemoteUpdateAttribute` {  
    `RU_LOGIN_NAME`,  
    `RU_PHONE`,  
    `RU_INTF_NAME`,  
    `RU_INTF_DN`,  
    `RU_NETWORK_DEVICE`,  
    `RU_INTF_STATE`,  
    `RU_UPSTREAM_SYNC_RATE`,  
    `RU_DOWNSTREAM_SYNC_RATE`,  
    `RU_TIME_STAMP`,  
    `RU_PORT_ID`,  
    `RU_NAS_IP`,  
    `RU_USER_IP_ADDRESS`,  
    `RU_NAS_INET_ADDRESS`,  
    `RU_USER_INET_ADDRESS`,  
    `RU_ROUTER_TYPE`,  
    `RU_last` }  
*Plug-in attributes that the application can pass to ACP.*
- enum `RemoteUpdateAttributeType` {  
    `PAT_LONG_LONG`,  
    `PAT_LONG`,

```
PAT_STRING,  
PAT_STATE,  
PAT_OPAQUE }
```

*Possible encoding types of attribute types.*

## 5.1.2 Typedef Documentation

### 5.1.2.1 typedef sequence<TagValue> `acpAdapter::TagValueList`

Attributes are passed to the `update()` method in the form of a `TagValue` list.

## 5.1.3 Enumeration Type Documentation

### 5.1.3.1 enum `acpAdapter::CongestionPointState`

Congestion point state indicates whether the interface is up or down.

**Enumeration values:**

`CP_UP` Interface is up.

`CP_DOWN` Interface is down.

### 5.1.3.2 enum `acpAdapter::RemoteUpdateAttribute`

Plug-in attributes that the application can pass to ACP.

**Enumeration values:**

`RU_LOGIN_NAME` Subscriber's login name specified as a string, such as `jane@isp1.com`.

`RU_PHONE` Telephone number specified as a string.

`RU_INTF_NAME` Interface name specified as a string, such as `fastEthernet3/1.2`.

`RU_INTF_DN` DN of the backbone congestion point provisioned in the LDAP directory specified as a string.

`RU_NETWORK_DEVICE` Virtual router name specified as a string.

`RU_INTF_STATE` Congestion point state, whether interface is in up or down state.

`RU_UPSTREAM_SYNC_RATE` Upstream bandwidth specified as 32-bit integer value.

***RU\_DOWNSTREAM\_SYNC\_RATE*** Downstream bandwidth specified as 32-bit integer value.

***RU\_TIME\_STAMP*** Timestamp in milliseconds specified as 64-bit integer value.

***RU\_PORT\_ID*** NAS port ID specified as a string.

***RU\_NAS\_IP*** NAS IP address specified as 32-bit integer value.

***RU\_USER\_IP\_ADDRESS*** User IP address specified as 32-bit integer value.

***RU\_NAS\_INET\_ADDRESS***

***RU\_USER\_INET\_ADDRESS*** NAS IP address in InetAddress.

***RU\_ROUTER\_TYPE*** User IP address in InetAddress.

***RU\_last*** type of the router

### 5.1.3.3 enum **acpAdapter::RemoteUpdateAttributeType**

Possible encoding types of attribute types.

#### Enumeration values:

***PAT\_LONG\_LONG*** 64-bit integer

***PAT\_LONG*** 32-bit integer

***PAT\_STRING*** character string

***PAT\_STATE*** CP\_UP or CP\_DOWN.

***PAT\_OPAQUE*** Byte array.

### 5.1.3.4 enum **acpAdapter::RemoteUpdateType**

Types of information that the application is updating.

The information type is passed as the first argument in the update() method.

#### Enumeration values:

***PE\_USER*** Subscriber information.

***PE\_CONGESTIONPOINT*** Congestion point information.

## Chapter 6

# API for Admission Control Plug-In (ACP) Class Documentation

### 6.1 acpAdapter.AttributeUnion Union Reference

```
import "acpPlugin.idl";
```

#### 6.1.1 Detailed Description

Structure used to pass attribute types to ACP.

#### Public Attributes

- long long [longLongVal](#)  
*64-bit integer value*
- string [stringVal](#)  
*character string value*
- [CongestionPointState](#) [congestionPointStateVal](#)  
*CP\_UP or CP\_DOWN.*
- unsigned long [longVal](#)  
*32-bit integer value*

- `sequence< octet > opaqueVal`

*Byte array value.*

## 6.1.2 Member Data Documentation

### 6.1.2.1 `CongestionPointState` `acpAdapter.AttributeUnion.congestionPointStateVal`

CP\_UP or CP\_DOWN.

### 6.1.2.2 `long long` `acpAdapter.AttributeUnion.longLongVal`

64-bit integer value

### 6.1.2.3 `unsigned long` `acpAdapter.AttributeUnion.longVal`

32-bit integer value

### 6.1.2.4 `sequence<octet>` `acpAdapter.AttributeUnion.opaqueVal`

Byte array value.

### 6.1.2.5 `string` `acpAdapter.AttributeUnion.stringVal`

character string value

The documentation for this union was generated from the following file:

- `acpPlugin.idl`

## 6.2 acpAdapter.RemoteUpdateInterface Interface Reference

```
import "acpPlugin.idl";
```

### 6.2.1 Detailed Description

Remote update interface.

The application uses the remote update interface to pass information to ACP.

### Public Member Functions

- void `update` (in [RemoteUpdateType](#) `rut`, in [TagValueList](#) `attrs`)  
*The `update()` method is called when the application passes subscriber or congestion point information to ACP.*

### 6.2.2 Member Function Documentation

#### 6.2.2.1 void `acpAdapter.RemoteUpdateInterface.update` (in [RemoteUpdateType](#) `rut`, in [TagValueList](#) `attrs`)

The `update()` method is called when the application passes subscriber or congestion point information to ACP.

The dynamic updates are not persistent.

#### Parameters:

- ← *`rut`* remote update type; indicates subscriber or congestion point information
- ← *`attrs`* attribute values; sent to ACP as tag value list

The documentation for this interface was generated from the following file:

- [acpPlugin.idl](#)

## 6.3 acpAdapter.TagValue Struct Reference

```
import "acpPlugin.idl";
```

### 6.3.1 Detailed Description

List of attributes in [TagValue](#) format.

- Tag identifies the attribute through its attribute type.
- Value contains the actual value of the attribute.

#### Public Attributes

- [RemoteUpdateAttribute](#) tag  
*Attribute type.*
- [AttributeUnion](#) value  
*Actual value of the attribute.*

### 6.3.2 Member Data Documentation

#### 6.3.2.1 [RemoteUpdateAttribute](#) acpAdapter.TagValue.tag

Attribute type.

#### 6.3.2.2 [AttributeUnion](#) acpAdapter.TagValue.value

Actual value of the attribute.

The documentation for this struct was generated from the following file:

- [acpPlugin.idl](#)

## Chapter 7

# API for Admission Control Plug-In (ACP) File Documentation

### 7.1 `acpPlugin.idl` File Reference

#### 7.1.1 Detailed Description

API that you can use to implement the Admission Control Plug-In (ACP).

#### Namespaces

- namespace [acpAdapter](#)

# Index

- acpAdapter, 9
  - CP\_DOWN, 11
  - CP\_UP, 11
  - PAT\_LONG, 12
  - PAT\_LONG\_LONG, 12
  - PAT\_OPAQUE, 12
  - PAT\_STATE, 12
  - PAT\_STRING, 12
  - PE\_CONGESTIONPOINT, 12
  - PE\_USER, 12
  - RU\_DOWNSTREAM\_SYNC\_RATE, 11
  - RU\_INTF\_DN, 11
  - RU\_INTF\_NAME, 11
  - RU\_INTF\_STATE, 11
  - RU\_last, 12
  - RU\_LOGIN\_NAME, 11
  - RU\_NAS\_INET\_ADDRESS, 12
  - RU\_NAS\_IP, 12
  - RU\_NETWORK\_DEVICE, 11
  - RU\_PHONE, 11
  - RU\_PORT\_ID, 12
  - RU\_ROUTER\_TYPE, 12
  - RU\_TIME\_STAMP, 12
  - RU\_UPSTREAM\_SYNC\_RATE, 11
  - RU\_USER\_INET\_ADDRESS, 12
  - RU\_USER\_IP\_ADDRESS, 12
- acpAdapter
  - CongestionPointState, 11
  - RemoteUpdateAttribute, 11
  - RemoteUpdateAttributeType, 12
  - RemoteUpdateType, 12
  - TagValueList, 11
- acpAdapter::AttributeUnion, 13
- acpAdapter::AttributeUnion
  - congestionPointStateVal, 14
  - longLongVal, 14
  - longVal, 14
  - opaqueVal, 14
  - stringVal, 14
- acpAdapter::RemoteUpdateInterface, 15
- acpAdapter::RemoteUpdateInterface
  - update, 15
- acpAdapter::TagValue, 16
- acpAdapter::TagValue
  - tag, 16
  - value, 16
- acpPlugin.idl, 17
- CongestionPointState
  - acpAdapter, 11
- congestionPointStateVal
  - acpAdapter::AttributeUnion, 14
- CP\_DOWN
  - acpAdapter, 11
- CP\_UP
  - acpAdapter, 11
- longLongVal
  - acpAdapter::AttributeUnion, 14
- longVal
  - acpAdapter::AttributeUnion, 14
- opaqueVal
  - acpAdapter::AttributeUnion, 14
- PAT\_LONG
  - acpAdapter, 12
- PAT\_LONG\_LONG
  - acpAdapter, 12
- PAT\_OPAQUE
  - acpAdapter, 12
- PAT\_STATE
  - acpAdapter, 12

- PAT\_STRING
  - [acpAdapter](#), [12](#)
- PE\_CONGESTIONPOINT
  - [acpAdapter](#), [12](#)
- PE\_USER
  - [acpAdapter](#), [12](#)
- RemoteUpdateAttribute
  - [acpAdapter](#), [11](#)
- RemoteUpdateAttributeType
  - [acpAdapter](#), [12](#)
- RemoteUpdateType
  - [acpAdapter](#), [12](#)
- RU\_DOWNSTREAM\_SYNC\_RATE
  - [acpAdapter](#), [11](#)
- RU\_INTF\_DN
  - [acpAdapter](#), [11](#)
- RU\_INTF\_NAME
  - [acpAdapter](#), [11](#)
- RU\_INTF\_STATE
  - [acpAdapter](#), [11](#)
- RU\_last
  - [acpAdapter](#), [12](#)
- RU\_LOGIN\_NAME
  - [acpAdapter](#), [11](#)
- RU\_NAS\_INET\_ADDRESS
  - [acpAdapter](#), [12](#)
- RU\_NAS\_IP
  - [acpAdapter](#), [12](#)
- RU\_NETWORK\_DEVICE
  - [acpAdapter](#), [11](#)
- RU\_PHONE
  - [acpAdapter](#), [11](#)
- RU\_PORT\_ID
  - [acpAdapter](#), [12](#)
- RU\_ROUTER\_TYPE
  - [acpAdapter](#), [12](#)
- RU\_TIME\_STAMP
  - [acpAdapter](#), [12](#)
- RU\_UPSTREAM\_SYNC\_RATE
  - [acpAdapter](#), [11](#)
- RU\_USER\_INET\_ADDRESS
  - [acpAdapter](#), [12](#)
- RU\_USER\_IP\_ADDRESS
  - [acpAdapter](#), [12](#)
- stringVal
  - [acpAdapter::AttributeUnion](#), [14](#)
- tag
  - [acpAdapter::TagValue](#), [16](#)
- TagValueList
  - [acpAdapter](#), [11](#)
- update
  - [acpAdapter::RemoteUpdateInterface](#), [15](#)
- value
  - [acpAdapter::TagValue](#), [16](#)