RDP Packet Handlers

RDP is a flexible and extensible application. This appendix describes the programming methodology in RDP that processes requests received from SSG. It includes the following topics:

- Packet Handlers, page E-1
- RDPPacket Class Description, page E-2
- Processing Requests in Proxy Mode, page E-5

Packet Handlers

This section describes the RDP packet handler class. It includes the following topics:

- Overview, page E-1
- Configuring the Packet Handlers, page E-2
- Adding Additional Packet Handlers, page E-2

Overview

The RDP application is very flexible in the way it handles requests that it receives from SSG. This flexibility is implemented with a number of different packet handlers, each handling a request in a different way. Developers at your site can extend the RDP application with additional packet handlers to provide even more flexibility.

RDP cycles a request from SSG through several levels of packet handlers, each one working to narrow down the type of packet, until a response is generated. The request is initially untyped and is processed by the packet handler for untyped packets. As the request gets processed by various packet handlers, it gets typed several times, each time with a more specific type. RDP creates a new packet object to process each newly assigned packet type.

Configuring the Packet Handlers

The RDPPacketFactoryMBean is the configurable class that specifies the packet handlers to use for each packet type. The rdp.xml file includes the following entries for each packet handler:

```
<Call name="addType">
      <Arg>packetType</Arg>
      <Arg>class</Arg>
    </Call>
```

Each <Call name="addType"> element takes two arguments: a packet type and a class that will handle that packet type. The packetType is a string. The class is a string specifying an RDPPacket derived class. Class parameters follow the class and are separated from it by a semicolon.

The RDPPacketFactoryMBean also accepts entries that set attributes. The attribute entries are used as parameters to the ProfileRequestPacket packet handler to narrow down the packet type.

```
<Call name="setAttribute">
      <Arg>PASSWORD:password</Arg>
      <Arg>packetType</Arg>
    </Call>
```

Each <Call name="setAttribute"> element takes two arguments: a password and a packetType.

There must be a corresponding <Call name="addType"> element for packetType, to specify the packet handler class for that packet type.

Adding Additional Packet Handlers

The packet handling mechanism is extensible. Web developers can write customized or additional packet handlers and map them to specific packet types by making changes or additions in the rdp.xml file.

RDPPacket Class Description

When RDP receives a request, it creates an RDPPacket. The packet handlers in the RDPPacket class have two public methods:

- getType method
- handle method

An RDPPacket derived class either overrides the getType method, in which case it narrows down the type of the packet, or it overrides the handle method, in which case it generates a response. An object calls the handle method first. If the handle method can process the request, it does so, generating the response. Otherwise, the default RDPPacket handle method calls the getType method.

The getType method determines some information about the type of packet. The default handle method uses the returned type to create a new RDPPacket derived packet. The handle method is then called on the new packet, as described in the previous paragraph.

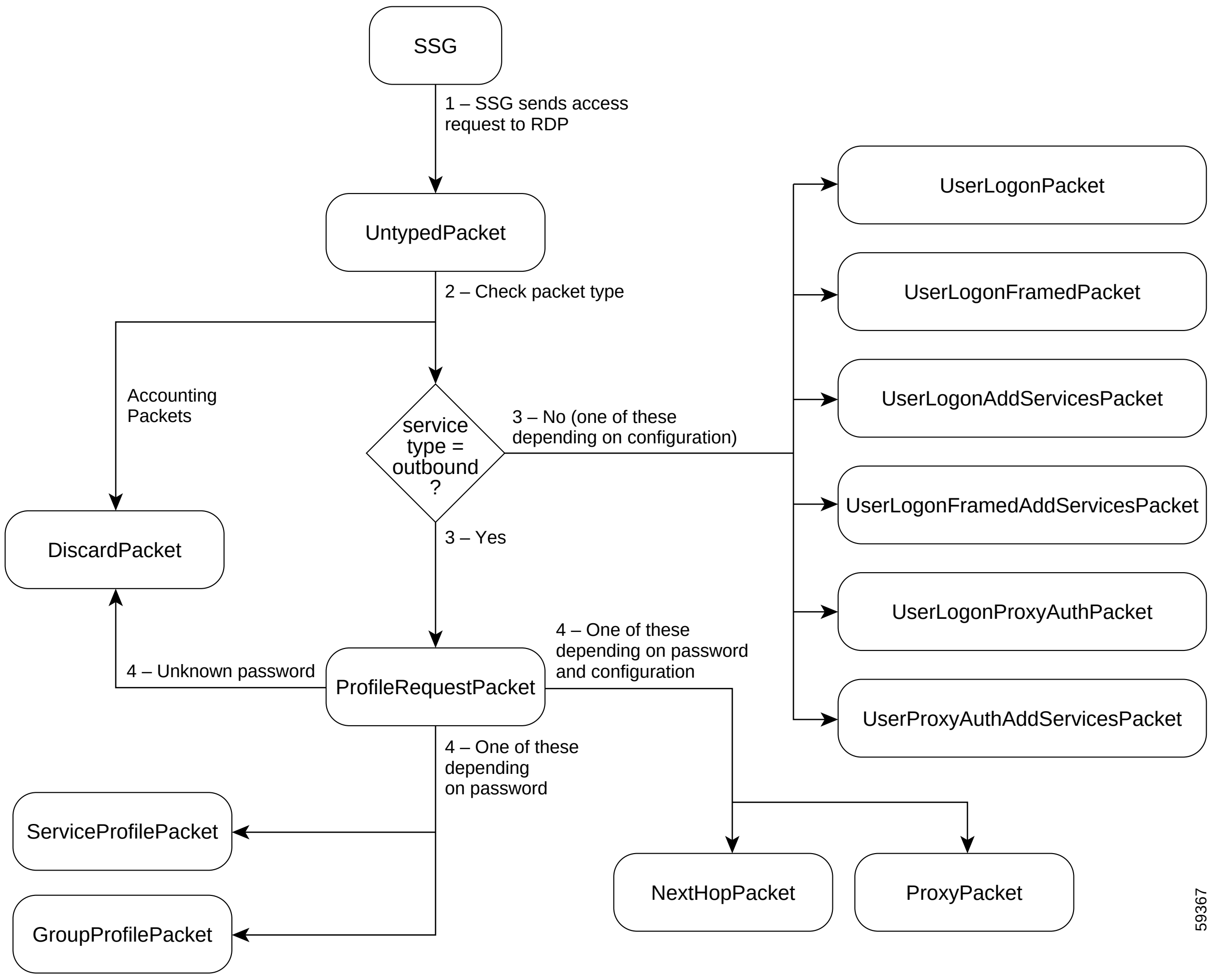
Table E-1 describes the RDPPacket classes included with the installed RDP application.

Table E-1 RDPacket Classes and Methods

Class	Methods
RDPPacket	getType—If the request is an Access Request, this method prompts you with Untyped. Otherwise, the method prompts you with Unknown.
DiscardPacket	handle—Returns null. (That is, it silently discards the request.)
RejectPacket	handle—Returns an Access Reject message.
UntypedPacket	getType—If the request contains the AV Service-Type with the value `Outbound`, then the method ProfileRequest appears. Otherwise, this method prompts you with UserLogon.
ProfileRequestPacket	getType—If the request contains a password that matches a password defined by the `PASSWORD:` attribute, this method displays the attribute's value. Otherwise, this method prompts you with Unknown.
ProxyPacket	handle—Proxies the request to an AAA server. Requires a parameter to define the name of the AAA MBean.
ServiceProfilePacket	handle—Uses the DESS API to create a service profile response.
GroupProfilePacket	handle—Uses the DESS API to create a group profile response.
NextHopPacket	handle—Uses the DESS API to create a next hop gateway response.
UserLogonAddServices Packet	handle—Uses the DESS API to authenticate and authorize a subscriber. All services and groups the subscriber is subscribed to appear.
UserLogonPacket	handle—Uses the DESS API to authenticate a subscriber. If the subscriber is using PPP, the subscriber's auto-logon services appear.
UserProxyAuthAdd ServicePacket	handle—Proxies the request to a AAA server, but uses DESS to add authorization information. Requires a parameter to define the name of an AAA MBean.
UserProxyAuthPacket	handle—Proxies the request to an AAA server, but uses DESS to add authorization information for auto-logon services if the user is a PPP user. Requires a parameter to define the name of an AAA MBean.

Figure E-1 shows how RDP processes a request from SSG. A detailed explanation follows the figure.

Figure E-1 RDP Request Processing


SSG
1 – SSG sends access request to RDP
UntypedPacket
2 – Check packet type
Accounting Packets
service type = outbound ?
3 – No (one of these depending on configuration)
DiscardPacket
3 – Yes
UserLogonPacket
UserLogonFramedPacket
UserLogonAddServicesPacket
UserLogonFramedAddServicesPacket
UserLogonProxyAuthPacket
UserProxyAuthAddServicesPacket
4 – Unknown password
ProfileRequestPacket
4 – One of these depending on password and configuration
4 – One of these depending on password
ServiceProfilePacket
GroupProfilePacket
NextHopPacket
ProxyPacket
59367

A request from SSG is processed in the following way:

1. The initial packet is handled by the base class. The getType method returns Untyped.
2. An Untyped packet is handled by the UntypedPacket class.
3. The getType method returns one of the following types:
   – If the packet contains the AV pair service-type = Outbound, getType returns ProfileRequest packet.
   – Otherwise, getType returns one of the UserLogon request packets, depending on values in the MBean configuration file.

4. A ProfileRequest packet is handled by the ProfileRequestPacket class. This class narrows the type again using the PASSWORD: attributes set in the rdp.xml file. If the password in the request (prepended with the string PASSWORD:) matches any of the password attributes set in the rdp.xml file, the getType method returns the packet type associated with the password in the corresponding <Call name="setAttribute"> element. Password attributes identify the following types of requests:
   - ServiceRequest—The ServiceRequest packet handler uses the DESS API to retrieve a list of services that this subscriber is authorized to access.
   - GroupRequest —The GroupRequest packet handler uses the DESS API to retrieve a list of services that this subscriber is authorized to access through group membership.
   - ProxyNextHop—The ProxyNextHop packet handler passes the request to the RADIUS server identified in the AAA MBean in the rdp.xml file.
   - If the password does not match any of the above, getType returns Unknown. An Unknown packet is handled by the RejectPacket packet handler.

   See the "RDPPacketFactory" section in Table 6-6 on page 6-32 for information about how to set these password values.

Processing Requests in Proxy Mode

When RDP is running in Proxy mode, profile requests are forwarded to a RADIUS server. This section describes the configuration entries in rdp.xml that make this happen. The section discusses the following entries from the installed rdp.xml file.

```
<Call name="setAttribute">
      <Arg>PASSWORD:nexthopcisco</Arg>
      <Arg>ProxyNextHop</Arg>
    </Call>

<Call name="addType">
      <Arg>ProxyNextHop</Arg>
      <Arg>com.cisco.aggbu.rdp.ProxyPacket;NextHop</Arg>
    </Call>

<Configure name="com.cisco.aggbu:name=AAA,connection=NextHop">
```

If a ProfileRequestPacket has the password nexthopcisco (this is an example; your password value might be different), it is typed ProxyNextHop. The `<Call name="addType">` element for ProxyNextHop maps the packet to the ProxyPacket class.

The ProxyPacket class accepts a string in its constructor which identifies the connection object that will handle the request. The string after the class name and semicolon in the `<Call name="addType">` element is passed to the ProxyPacket class constructor. This connection object name matches the connection object configured by the AAA MBean.