Local Messaging

> *Previously AA32-GPayments Card Loader.pdf*

This section details the messaging specification for the ActiveAccess cardholder and user registration application programming interface (API).

Cardholder Registration

The authentication system is responsible for authentication of cardholders during American Express SafeKey, Diners Club International ProtectBuy, JCB J/Secure, Mastercard SecureCode / IDC and Verified by Visa / Visa Secure transactions. As a result, it is necessary that the system stores cardholder related information to the extent that this requirement can be satisfied. Cardholder registration is the process through which an issuer registers this information with ActiveAccess. Depending on the choice of registration model, cardholders may or may not have to complete an enrolment process. Please note, the distinction between 'registration' and 'enrolment,' as used in this document, where registration is carried out by the issuer (3DS1 and 3DS2) and enrolment is performed by the cardholder (3DS1 only).

There are three cardholder registration models for populating data into the authentication system. **Direct entry** (3DS1 only) allows help desk operators to manually enter cardholder registration data through the ActiveAccess administration interface. The **pre-registration** (3DS1 only) and **final registration** models both format cardholder data in an XML message. These messages are either sent directly to the registration server via an API, using the CardLoader application, or uploaded through the ActiveAccess Administration interface. To facilitate these processes and integrate with the authentication system some integration work with the issuer's systems is required.

Info

Refer to New Card for more information on the direct entry cardholder registration model.

The purpose of registration is to upload or supply the information necessary to enable a cardholder to use the authentication protocols. For each registered card, the registration message must include the required fields; card type, card number and card name.

There are four types of XML messages accepted by the system for the purpose of cardholder creation and maintenance. They are:

- **PreReg** (3DS1 only): enables the creation of pre-registered cardholder data in the system. When cards have this status, there is no authentication data available for the card. Authentication data can be assigned to the card by:
  - the cardholder through the Enrolment component (supports static password only)
  - the cardholder via Activation During Shopping
  - admins via MIA > Cards
  - admins via uploading of FinalReg requests.
- **FinalReg**: enables the creation of fully registered cardholder data in the system. When cards have this status, they have one or more authentication data that can be used for authentication.
- **UpdateReg**: enables the alteration of cardholder data in the system
- **CancelReg**: enables the removal of cardholder data from the system.

> **Info**
>
> **Section - Request** provides more detail on the format of these messages.

To perform authenticated transactions, it is essential that the cardholder data elements stored within the authentication system is sufficient to allow this. The essential elements required are:

- Card type
- Card number
- Card name
- Personal assurance message (personal greeting)

Enrolled cards may also use the following elements:

- Password (J/Secure Password, ProtectBuy Password, Mastercard SecureCode, SafeKey Password, or Visa Password).
- Expiry Date
- Device Type
- Device Serial Number

- Hint (used for authorisation retry prompt)
- Hint and Response (question and answer pair used to verify cardholders if they have forgotten their authentication password)

The information required for cardholder authentication is primarily provided to the authentication system by the card issuing member bank. The Registration API provides a flexible data transport and definition mechanism that allows each member bank to build an individual cardholder registration process to meet their specific requirements.

Alternatively, cardholder data can be formatted into XML files and uploaded through the ActiveAccess administration interface. This process reduces the technical requirements for the issuer.

Pre-registration (3DS1 only)

When using the pre-registration model, a member bank only needs to establish the authentication criteria for enrolment of cardholders and provide this information to the authentication system. The actual enrolment process is then handled by the authentication system's enrolment module. The authentication system uses the pre-registration information to verify the identity of the cardholder and set-up the cardholder account during the cardholder enrolment process.

In the pre-registration model, it is essential that a **PreReg** message uploads the card type, card number and card name, regardless of the protocol. It is also required that the issuer supplies a number of other known parameters to the authentication system to verify the cardholder when they come to enrol with the authentication system. The following provides a list of suggested data that can be included in the **PreReg** message to allow the cardholder authentication to occur. Please note that these fields are at the discretion of the issuing bank:

- Date of birth
- Mother's maiden name
- Card verification check number
- Credit limit
- Billing address

An example of pre-registration is when the member bank mails invitations to cardholders and provides them with, say, a registration number. Cardholders can then visit the authentication system's online enrolment website or enrol using the activation during shopping process. Having

entered their card number, card name and registration number and once the cardholder's identity is successfully verified, they can proceed with setting up their account, which includes selection of an American Express SafeKey, Diners Club International ProtectBuy, JCB J/Secure, Mastercard SecureCode or Verified by Visa password. This static or dynamic authentication data will then be used in all subsequent authenticated transactions.

Final Registration

The final registration model allows issuers to control the cardholder enrolment process. It requires the member bank to develop its own enrolment process to collect cardholder enrolment information. Once the cardholder enrolment process is complete, the information is sent to the authentication system in the form of a final registration message.

There are two types of final registration, final registration for traditional 3-D Secure authentication and final registration for two-factor authentication over 3-D Secure.

In the final registration model for traditional 3-D Secure, it is essential that a **FinalReg** message uploads the card type, card number and card name, regardless of the protocol. It is also required that the issuer supplies the cardholder authentication fields:

- SafeKey, J/Secure Password, ProtectBuy Password, Mastercard Identity Check / SecureCode, or Visa Password
- Personal Assurance Message

In the final registration model for two-factor authentication over 3-D Secure it is also essential that the issuer supplies the device information which will be used for generating tokens at authentication time in addition to above fields:

- Device Type
- Device Serial Number

An example of final registration is enrolment of cardholders through the issuer's Internet banking site. A possible scenario is for the cardholder to log into the Internet banking site and enable American Express SafeKey, Diners Club International ProtectBuy, JCB J/Secure, Mastercard SecureCode or Verified by Visa for their credit card. The cardholder will then be prompted to choose a SafeKey, J/Secure password, ProtectBuy Password, Mastercard SecureCode or a Verified by Visa password and personal assurance message for their card. If the issuer wants to enable two-factor authentication over 3-D Secure for the card then the cardholder will be prompted to select a device from a list of supported device types and enter its serial number.

The issuer then formats a final registration message, which is sent to the authentication system in order to complete cardholder enrolment.

The format of the registration request is the same for real-time and batch uploads, except that the batch upload may contain multiple blocks of cardholder data.

The registration API uses XML as the message format and HTTP as the message transport. API calls (requests) can be made to the registration server of the authentication system. Issuers can use the registration API to automate their cardholder registration process.

XML Message Format

XML messages must be produced in accordance with the Messaging Requirements Cardholder Registration DTD, as specified in Cardholder Registration DTD.

All request and response messages are enclosed by the `<Message>` element. A message can contain either a **Request and a Signature** or a **Response**.

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456">

<!--the request content here-->

</request>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<!--the request signature here-->

</Signature>

</Message>
```

<Message>		
Attributes	**Description**	**Usage**
<Request>	Used in the request message, which is sent by the issuer to register one or more cards.	Required for request messages

<Message>		
<Signature>	Issuer signature is used in the request message to prove the identity of the issuer and to validate that the message content has not been altered.	Required for request message
<Response>	Cardholder registration response sent back in response to a cardholder registration request.	Required for response message

Note

The XML response starts with the XML declaration `(<?xml version="1.0" encoding="UTF-8"?>)`. However a request does not need to start with the XML declaration. Request content must be sent with UTF-8 encoding.

Request

A Request is sent by the issuer to perform various cardholder registration tasks. A request can be sent to perform pre-registration, final registration, cancel registration or update registration for one or more cards. A request may only contain one <PreReg>, <FinalReg>, <CancelReg> or <UpdateReg>.

<Request>		
Attributes	**Description**	**Usage**
Id	An arbitrary identifier, which the issuer defines and can be used to refer to the request element as part of a standard URI. Also referenced by the Signature element. XML signature requires the element, which is being signed, to be identified by a unique Id. The value entered should start with an alphabetic character.	Required Max 28 char
IssuerId	A unique identifier for the issuer. Created when the issuer first signs up with the system and supplied during the issuer registration process. Typically an 18 digit numeric value.	Either IssuerId or GroupId is required
GroupId	A unique identifier for the group of issuers. Created when the group is introduced to the system and supplied during the registration process. Typically an 18 digit numeric value. Using a GroupId, cards can be registered for different issuer members within the group in an identical request.	Either IssuerId or GroupId is required

<Request>		
EncVectorIV	If an encryption keystore has been defined for the issuer, or a group of issuers, critical card data must be encrypted using it. The critical data is encrypted using AES/CBC/PKCS5Padding mode, which requires an IV, including 16 random bytes, as an input parameter for encryption and decryption. The IV should be sent to the server to indicate that the card data in the response should be encrypted in CBC mode. To do this, the IV itself must be encrypted in AES/ECB/PKCS5Padding mode, using an encryption key, then base64 encoded and set as EncVectorIV in the request.	Optional, if present, it means that the client has generated an IV parameter and critical card information has been encrypted using the CBC mode and the generated IV, otherwise ECB or plain mode has been used instead.
Elements	**Description**	**Usage**
<PreReg>	Used for pre-registration of one or more cards. Pre-registered Cardholders will need to go through the enrolment process to finalise their registration.	Required for a pre-registration request
<FinalReg>	Used for registration of one or more cards. Registered Cardholders can start making authenticated transactions without the need to go through the enrolment process.	Required for a final registration request
<CancelReg>	Used to remove one or more cards from the system.	Required for a cancel registration request
<UpdateReg>	Used to update the information for one or more cards. For example to change the card number. Cardholder registration status is left unchanged.	Required for an update registration request

Response

A response message is sent back for each request. The response message provides the result of the request message with details of errors, if any. Issuers must process the response message and should correct and replay their request if there is an error.

<Response>		
Attributes	**Description**	**Usage**

<Response>		
EncVectorIV	If an encryption keystore has been defined for the issuer or group of issuers, depending on the encryption key algorithm, the server decrypts critical card data using either AES/ECB/PKCS5Padding or DESede/ECB/PKCS5Padding mode. If the EncVectorIV attribute is set to protect the data using the AES/CBC/PKCS5Padding or DESede/CBC/PKCS5Padding mode, card data is decrypted using the CBC encryption mode and the IV is sent as an input parameter. During processing on the server side, a new random IV is generated and used to encrypt critical card data in AES/CBC/PKCS5Padding or DESede/CBC/PKCS5Padding mode. When the process is complete, the IV itself is encrypted in AES/ECB/PKCS5Padding or DESede/ECB/PKCS5Padding mode using the same key depending on the key algorithm, and then base64 encoded and set as EncVectorIV in the response.	Required, if the client has set this attribute for the request. Server generates a new IV parameter and encrypts critical card information in the response using CBC mode and the new IV, otherwise no attribute will be set and ECB or plain mode will be used instead.
Elements	**Description**	**Usage**
<Code>	Response code. 0 if the request was successful. 1 if the request has been successfully processed but there are warnings. Any other value denotes an error in processing the request.	Required
<ErrorMessage>	A descriptive message that identifies the category of the error.	Required
<ErrorDetail>	A more detailed description of the error.	Required
<Warning>	A warning is issued to provide information on an unexpected situation that does not prevent the request from being successfully processed.	Conditional. Required only if response code is 1.

Note

A message with response code **1** denotes that the request has been successfully processed but yet the registration server has not been able to comply with certain instructions. For example a cancel registration attempt on an already cancelled card is not processed and as such a warning message is reported. An issuer does not need to take any further action in this case.
Warning messages may be logged at the issuer's end for later reference.

> **Note**
>
> Multiple warning messages may be included in a single response message.

Requests

Pre-registration Request

The pre-registration request is typically used by the members who wish to leave cardholder enrolment to the enrolment module. A pre-registered card cannot be used for authenticated transactions. Cardholders are required to finalise their registration by going through the standard enrolment process, which is offered by the enrolment module or through the cardholder activation during shopping process. Pre-registration data is used to verify the identity of cardholders during the enrolment process.

<PreReg>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**

<PreReg>		
<DataFormat>	Defines a data type, which can be associated with card data. This issuer-defined field has a maximum field length of 1024 characters. You can define the following attributes with the DataFormat: **Name:** The name as used by the program to refer to this data format. Also used by the data element in order to refer to this particular type, e.g.: pass or password **Label:** A short description to appear before the data elements of this type when displayed to the cardholder, e.g.: SecureCode: **Description:** A longer description to appear after the data elements of this type when displayed to the cardholder (optional), e.g.: Please enter your SecureCode **MaxLen:** The maximum length that can be stored in the data elements of this type (optional). **Type:** Can be set to date, string, number or hidden values. When set to date, number, the application performs type validation for the content of the data elements. If not specified the string type is assumed (no validation). If the data format type is set to hidden, this excludes the data format as an authentication data element. Hidden data types are not displayed to the cardholder and can be used as internal field in the XSL pages per bank to achieve certain customised features. You should not use this type unless specifically advised by GPayments to do so **Format:** Additional formatting information can be set to YYYYMMDD or YYYYMM. This is only meaningful when you have set the data format type to date. **Mask:** Can be set to Yes or No. Determines whether the user input for this data format should be masked or not. Set to Yes for password type fields. **DataMode:** This is an optional attribute. Its value can be **Identity**, **identity**, **Auth**, **auth**, **Extension**, or **extension**. Data which has Identity as DataMode will be displayed on the Registration page and may be displayed on the Forgot Password and Hint/Response pages to identify the cardholder during changing or resetting the password. Data which has Auth as DataMode will be displayed during authentication, including the Authentication and Reactivation pages. Data which has Extension as Datamode will be checked by extensions. If it is not set, the default value will be used. In the PreReg file, the default value is Identity.	Required A valid data format must be defined for each type, which is referenced by a <Data> element.
<Card>	Card related data including card number, name on card, expiry date and issuer defined data. Card numbers up to 19 digits are accepted and the name on card field has a maximum length of 128 characters.	Required At least one <Card> must be present

Note

If a data has a 'discard=no' attribute, it will be kept after cardholder registration via enrolment or Activation During Shopping has completed. Otherwise, it will be removed from card data.

Note

If an encryption key is defined for the card issuer/group:

Card Number, **Name**, **PAM**, **HINT**, **HINT Response** and **Data.Value** should be sent encrypted and the Registration Server will need to decrypt these fields before using them.

Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

The pre-registration data may include existing customer information such as date of birth, mother's maiden name, card verification check, credit limit, billing address, etc or a registration number distributed by mail (or in some cases a combination of both). The requirements for pre-registration information are at the discretion of member banks. The pre-registration request is dynamic enough to meet the authentication requirements of each individual member bank. However, the enrolment module determines the presentation of authentication data to the cardholder during the enrolment process.

Member banks, which require greater flexibility in the presentation or control of the enrolment process, should use the final registration model.

The pre-registration data may be removed once a cardholder has successfully completed the enrolment process.

Multiple unsuccessful enrolment attempts may cause the cardholder to be locked. This limit can be set on a per issuer basis and through the ActiveAccess administration interface.

To disable a cardholder registration, a cancel registration request should be used. To un-register finally registered cardholder, the cardholder account should be cancelled first and a new pre-registration message should be sent. This will require the cardholder to repeat the enrolment process.

A sample pre-registration request is displayed below. The following request will provide pre-registration information, which is required for enrolment of Mr. Joe Citizen for American Express SafeKey, Mastercard SecureCode and Verified by Visa.

SAMPLE PRE-REGISTRATION REQUEST

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456789">

    <PreReg>
```

```
        <DataFormat Name="birthdate" Type="date" Format="YYYYMMDD" Label="Date of
        Birth:"/>

        <DataFormat Name="credit" Type="number" Label="Credit Limit:"
Desc="Please
        enter your credit card limit"/>

        <DataFormat Name="regpassword" Type="string" Label="Registration
Password:"
        Desc="Please enter your registration password" Mask="Yes"/>

        <Card Type="SPA" Number="5012345678901234" Name="Joe Citizen">

            <ClientId>819737457046382</ClientId>

            <ExpDate>202204</ExpDate>

            <Data Name="birthdate" Value="19730201"/>

            <Data Name="credit" Value="10000"/>

            <Data Name="regpassword" Value="pro2345"/>

        </Card>

        <Card Type="VbV" Number="4012345678901234" Name="Joe Citizen">

            <ClientId>819737457046382</ClientId>

            <ExpDate>202202</ExpDate>

            <Data Name="birthdate" Value="19730201"/>

            <Data Name="credit" Value="3000"/>

            <Data Name="regpassword" Value="pro2345"/>

        </Card>

        <Card Type="SK" Number="373700000000000" Name="Joe Citizen">

            <ClientId>819737457046382</ClientId>

            <ExpDate>202204</ExpDate>

            <Data Name="birthdate" Value="19730201"/>

            <Data Name="credit" Value="10000"/>

            <Data Name="regpassword" Value="pro2345"/>
```

```
        </Card>
    </PreReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

Final Registration Request

Members who wish to have greater control over the cardholder enrolment process typically use the final registration request. In this case the member bank itself handles the enrolment process. The registration process should result in the selection of an authentication password between the member bank and the cardholder. A personal assurance message (PAM) should also be set for 3-D Secure cards. If the issuer also supports two-factor authentication for 3-D Secure, device information will need to be set. The member bank will then format a final registration request and provide the agreed authentication information to the enrolment module for storage.

<FinalReg>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**

<FinalReg>		
<DataFormat>	Defines a data type, which can be associated with card data. This issuer-defined field has a maximum field length of 1024 characters. You can define the following attributes with the DataFormat: **Name:** The name as used by the program to refer to this data format. Also used by the data element in order to refer to this particular type, e.g.: pass or password **Label:** A short description to appear before the data elements of this type when displayed to the cardholder, e.g.: SecureCode: **Description:** A longer description to appear after the data elements of this type when displayed to the cardholder (optional), e.g.: Please enter your SecureCode **MaxLen:** The maximum length that can be stored in the data elements of this type (optional). **Type:** Can be set to date, string, number or hidden values. When set to date, number, the application performs type validation for the content of the data elements. If not specified the string type is assumed (no validation). If the data format type is set to hidden, this excludes the data format as an authentication data element. Hidden data types are not displayed to the cardholder and can be used as internal field in the XSL pages per bank to achieve certain customised features. You should not use this type unless specifically advised by GPayments to do so **Format:** Additional formatting information can be set to YYYYMMDD or YYYYMM. This is only meaningful when you have set the data format type to date. **Mask:** Can be set to Yes or No. Determines whether the user input for this data format should be masked or not. Set to Yes for password type fields. **DataMode:** This is an optional attribute. Its value can be **Identity**, **identity**, **Auth**, **auth**, **Extension**, or **extension**. Data which has Identity as DataMode will be displayed on the Registration page and may be displayed on the Forgot Password and Hint/Response pages to identify the cardholder during changing or resetting the password. Data which has Auth as DataMode will be displayed during authentication, including the Authentication and Reactivation pages. Data which has Extension as Datamode will be checked by extensions. If it is not set, the default value will be used. In the PreReg file, the default value is Identity.	Required A valid data format must be defined for each type, which is referenced by a element.

<FinalReg>		
<Card>	Card related data including card number, name on card, expiry date and issuer defined data. Card data may also include device information for two-factor authentication enabled cards. Card numbers of 19 digits are accepted and the name on card field has a maximum length of 128 characters. Device information includes, Device type {1: VASCO, 3: SMS, 7: OOB, 6: Email, 8: Decoupled Authentication} and for tokens: serial no which is a unique number assigned to each device by the manufacturer; for SMS: mobile number; and for Email: email address.	Required At least one <Card> must be present

Note

If an encryption key is defined for the card issuer/group:

Card Number, **Name**, **PAM**, **HINT**, **HINT Response**, **Data.Value** and Device.SerialNo should be sent encrypted and the Registration Server will need to decrypt these fields before using them.

Refer to section **Critical Card Data Encryption and Decryption** for further details of the requirements for the encryption/decryption process.

Cardholders registered using the final registration method can perform authenticated transactions as soon as the registration request is processed, without the need for any direct interaction with the enrolment module.

This method also allows issuers to enrol cardholders by using existing authentication data. For example, a member bank may choose to send the Internet banking username and password for authentication. Any change in username or password of the Internet banking site can also be reflected on the enrolment module by sending a new **FinalReg** request to keep the two systems synchronised.

The following request will provide final registration information, which is required for Mr. Joe Citizen to use his cards with traditional American Express SafeKey, Mastercard SecureCode and Verified by Visa.

SAMPLE FINAL REGISTRATION REQUEST FOR TRADITIONAL 3-D SECURE

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456789">

    <FinalReg>

        <DataFormat Name="Password" Type="string" Label="Password:" Desc="Please
enter
        your authentication password " Mask="Yes"/>
```

```
            <DataFormat Name="Country" Type="singleSelect" Label="Please choose
the where you live in: " Desc="Please enter your country " Mask="No" MaxLen="64">

            <Option Label="Australia" Value="100"/>

            <Option Label="Germany" Value="200"/>

        </DataFormat>

            <DataFormat Name="City" Type="multiSelect" Label="Please choose the
cities you have been to: " Desc="Please select the cities " Mask="No"
MaxLen="64">

            <Option Label="Sydney" Value="11"/>

            <Option Label="Melbourne" Value="12"/>

            <Option Label="Brisbane" Value="13"/>

            <Option Label="Gold Coast" Value="14"/>

        </DataFormat>

        <Card Type="SPA" Number="5012345678901234" Name="Joe Citizen">

            <ClientId>819737457046382</ClientId>

            <ExpDate>202204</ExpDate>

            <PAM>I am sure that this is my bank</PAM>

            <Data Name="Password" Value="409634"/>

        </Card>

        <Card Type="VbV" Number="4012345678901234" Name="Joe Citizen">

            <ExpDate>202202</ExpDate>

            <PAM>I am sure that this is my bank</PAM>

            <HINT>Your childhood hero</HINT>

            <HINTResponse>Superman</HINTResponse>

            <Data name="Password" Value="354607"/>

        </Card>

        <Card Type="SK" Number="373700000000000" Name="Joe Citizen">
```

```
            <ExpDate>202204</ExpDate>
            <PAM>I am sure that this is my bank</PAM>
            <Data Name="Password" Value="409634"/>
        </Card>
        <Card Name="Joe Citizen" Number=" 3528457610673260" Type="JCB">
            <PAM>My personal message</PAM>
            <HINT>You know</HINT>
            <HINTResponse>Dreams come true</HINTResponse>
            <Data Name="Password" Value="123456">
            </Data>
            <Data Name="Country" Value="100">
            </Data>
            <Data Name="City">
                <SelectedOption Value="11"/>
                <SelectedOption Value="13"/>
            </Data>
        </Card>
    </FinalReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

And this request will provide final registration information, which is required for Mr. Joe Citizen to use his cards with two-factor authentication over American Express SafeKey, Mastercard SecureCode and Verified by Visa.

SAMPLE FINAL REGISTRATION REQUEST FOR TWO-FACTOR AUTHENTICATION OVER 3-D SECURE

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="123456789012345678">

    <FinalReg>

        <DataFormat Name="Password" Type="string" Label="Password:" Desc="Please
enter
        your authentication password " Mask="Yes"/>

        <Card Type="SPA" Number="5012345678901234" Name="Joe Citizen">

            <ClientId>819737457046382</ClientId>

            <ExpDate>202204</ExpDate>

            <PAM>I am sure that this is my bank</PAM>

            <Data Name="Password" Value="409634"/>

            <Device>

                <DeviceType>1</DeviceType>

                <SerialNo>0097123456</SerialNo>

            </Device>

        </Card>

        <Card Type="VbV" Number="4012345678901234" Name="Joe Citizen">

            <ExpDate>202202</ExpDate>

            <PAM>I am sure that this is my bank</PAM>

            <HINT>Your childhood hero</HINT>

            <HINTResponse>Superman</HINTResponse>

            <Data name="Password" Value="354607"/>
```

```
        <Device>
            <DeviceType>2</DeviceType>
            <SerialNo>79722647</SerialNo>
        </Device>
    </Card>
    <Card Type="SK" Number="3712345678901234" Name="Joe Citizen">
        <ExpDate>202204</ExpDate>
        <PAM>I am sure that this is my bank</PAM>
        <HINT>Your childhood hero</HINT>
        <HINTResponse>Superman</HINTResponse>
        <Data name="Password" Value="465809"/>
        <Device>
            <DeviceType>3</DeviceType>
            <SerialNo>+61400000000</SerialNo>
            <Param Name="SMSC">SMSGateWay</Param>
        </Device>
        <Device>
            <DeviceType>6</DeviceType>
            <SerialNo>username@domain.com</SerialNo>
        </Device>
        <Device>
            <DeviceType>7</DeviceType>
            <Param Name="OOBAdapter">sample-oob-adapter</Param>
        </Device>
        <Device>
```

```
                <DeviceType>7</DeviceType>
                <Param Name="OOBAdapter">restful-adapter</Param>
                <Param Name="OOBDeviceId">97d77002-b0b0-494e-922e-9e4cddfc6db9</
Param>
            </Device>
        </Card>
    </FinalReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

Cancel Registration Request

Cancel registration can be used to remove cardholder data (either pre-registration or full registration data). A card that has been cancelled can no longer be used in authenticated payments.

Cancel registration is typically used when a cardholder's account is closed or cancelled. A **CancelReg** request should consist of simple blocks of card data. The registration module ignores the content of the element, as it only requires the card number (and optionally name on card) for cancelling the cardholder's registration.

The following listing cancels the American Express SafeKey, Mastercard SecureCode and Verified by Visa registration of Mr. Joe Citizen.

> **Note**
>
> If an encryption key is defined for the card issuer/group:
>
> **Card** Number and **Name** should be sent encrypted and the Registration Server will need to decrypt these fields before using them.
>
> Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

SAMPLE CANCEL REGISTRATION REQUEST

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="123456789012345678">

    <CancelReg>

        <Card Type="SPA" Number="5012345678901234" Name="Joe Citizen"></Card>

        <Card Type="VbV" Number="4012345678901234"></Card>

        <Card Type="SK" Number="3712345678901234" Name="Joe Citizen"></Card>

    </CancelReg>

</Request>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<!--the request signature here-->

</Signature>

</Message>
```

Update Registration Request

An update registration is used to update card information, such as card number, name on card, expiry date, PAM, Hint/Response and status (`Enabled` / `Disabled`). It is used to update cardholder data without affecting the cardholder's enrolment status. For example, if the cardholder's card number has changed, it is possible to save the cardholder from going through the enrolment process again by simply updating the card number.

The following listing shows how Mr. Joe Citizen's card number can be updated. Mr. Citizen will be able to continue making authenticated payments without the need to re-enrol.

SAMPLE UPDATE REGISTRATION REQUEST

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="123456789012345678">
```

```
    <UpdateReg>
        <CardUpdate Number="5012345678901234" Name="Joe Citizen">
            <Number>5012345678943210</Number>
            <ClientId>819737457046383</ClientId>
            <ExpDate>202304</ExpDate>
            <Status>Enabled</Status>
        </CardUpdate>
        <CardUpdate Number="5123456789012353" Name="Joe Citizen">
            <ClientId>819737457046384</ClientId>
            <ExpDate>202304</ExpDate>
            <Status>Disabled</Status>
        </CardUpdate>
    </UpdateReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

In cases where a cardholder's account number changes, either to another card number of the same card scheme or to another card scheme, an update registration can be used to change these details. By using a CardCopy, the account details on the existing card will be transferred to a new card, with both accounts being enabled at the same time. The issuer must then disable or cancel the original account. CardCopy allows any of the card data format parameters and card attributes to be changed.

> **Note**
>
> If the Issuer is configured to use Mastercard Identity Check, the registered Mastercard cards of the issuer cannot be copied. This is because devices are not copied during CardCopy and Mastercard Identity Check requires at least one device to be assigned to the card.

> **Note**
>
> If an encryption key is defined for the card issuer/group:
>
> **Card** Number, **Name**, **PAM**, **HINT**, **HINT Response** and **Data.Value** should be sent encrypted and the Registration Server will need to decrypt these fields before using them.
>
> Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

Card Device Update Request

SAMPLE CARD DEVICE UPDATE REQUEST

```
<Message>

<Request Id="request1" IssuerId="123456789012345678">

    <DeviceUpdateReg>

        <CardDeviceUpdate Number="5012345678901234" Name="Joe Citizen">

            <DeviceUpdate Operation="Add">

                <Device>

                    <DeviceType>7</DeviceType>

                    <Param Name="OOBAdapter">restful-adapter</Param>

                    <Param
Name="OOBDeviceId">123456123456123456123456123456123477</Param>

                </Device>

                <Device>

                    <DeviceType>7</DeviceType>

                    <Param Name="OOBAdapter">restful-adapter</Param>
```

```
                    <Param
Name="OOBDeviceId">777666666666666666666666666666666799</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">sample-oob-adapter</Param>
                    <Param
Name="OOBDeviceId">888666666666666666666666666666666899</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">sample-oob-adapter</Param>
                    <Param
Name="OOBDeviceId">888666666666666666666666666666666899</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">sample-oob-adapter</Param>
                </Device>
            </DeviceUpdate>
            <DeviceUpdate Operation="Delete">
                <Device>
                    <DeviceType>3</DeviceType>
                    <SerialNo>+4111112323</SerialNo>
                    <Param Name="SMSC">SMSC</Param>
                </Device>
                <Device>
```

```
                    <DeviceType>3</DeviceType>
                    <SerialNo>+641111</SerialNo>
                    <Param Name="SMSC">SMSC</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">restful-adapter1</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">restful-adapter</Param>
                    <Param
Name="OOBDeviceId">66666666666666666666666666666666777</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">restful-adapter</Param>
                    <Param
Name="OOBDeviceId">12345612345612345612345612345612345</Param>
                </Device>
            </DeviceUpdate>
        </CardDeviceUpdate>
        <CardDeviceUpdate ClientId="123456789012345">
            <DeviceUpdate Operation="Add">
                <Device>
                    <DeviceType>3</DeviceType>
                    <SerialNo>+4411111</SerialNo>
```

```
                    <Param Name="SMSC">SMSC</Param>
                </Device>
                <Device>
                    <DeviceType>7</DeviceType>
                    <Param Name="OOBAdapter">restful-adapter</Param>
                </Device>
            </DeviceUpdate>
            <DeviceUpdate Operation="Delete">
                <Device>
                    <DeviceType>3</DeviceType>
                    <SerialNo>+6411112323</SerialNo>
                </Device>
            </DeviceUpdate>
        </CardDeviceUpdate>
    </DeviceUpdateReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

Notification

A Notification is a record of a single cardholder event. Each event is stored in ActiveAccess and a record is logged in the following events:

- Cardholder completes their registration

- Cardholder re-activates its account by registering a new or an existing device during authentication
- Administrator registers a new or an existing device for a specified cardholder via MIA
- Administrator unregisters a device from a specified cardholder
- Administrator removes a device from a specified cardholder
- Cardholder opts-out of Activation During Shopping
- Cardholder locks their account.

At regular intervals, ActiveAccess runs a procedure to collect all notification events, storing them in separate files based on the event type. The files are stored on the server and made available for a short period. To download notifications, an issuer should send a Notification Request, requesting the notification period and the type of notification event. ActiveAccess will return the data in the form of a Notification Response message as outlined in the following sections. If the request is unable to be processed entirely, an error code, error message and error details should be sent back in the response message.

Events that can be retrieved by this method are:

- Card registration
- Card device update
- Card lock and unlock
- ADS Opt-out

XML Message Format

XML messages must be produced in accordance with the Messaging Requirements, as specified in Section **Notification DTD**.

All request and response messages are enclosed by the **<Message>** element. A message can contain either a **Request and a Signature** or a **Response**.

A message can contain either a NotificationRequest and a Signature or a NotificationResponse

```
<?xml version="1.0"?>

<Message>

<NotificationRequest Id="Notification1" IssuerId="1234567890123456789"
```

```
Type="<!–Type of the query here (must be either cardreg, carddeviceupdate,
cardoptout or cardlock) -->">

<StartDate><!-- the query start date here. For example
200901010000--></StartDate>

<EndDate><!-- the query end date here. For example 200901150000
--></EndDate>

</NotificationRequest>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<!--the request signature here-->

</Signature>

</Message>
```

<Message>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**
<NotificationRequest>	Used in the request message sent by the issuer to query cardholder events.	Required for request messages
<Signature>	Issuer signature is used in the request message to prove the identity of the issuer and to validate that the message content has not been altered.	Required for request message
<NotificationResponse>	The query response sent back in response to a notification request.	Required for response message

Request

A Request is sent by the issuer to query various cardholder events. A request can be sent to query Card Registration, Opt-out or Card Lock notifications.

The issuer sends a NotificationRequest requesting the period of time and the type of notification event data they require. A request may be of the type: CardReg, CardLock or CardOptOut.

<NotificationRequest>		
Attributes	**Description**	**Usage**
Id	An arbitrary identifier, which the issuer defines and can be used to refer to the request element as part of a standard URI. Also referenced by the Signature element. XML signature requires the element, which is being signed, to be identified by a unique The value entered should start with an alphabetic character.	Required
IssuerId	A unique identifier for the issuer. Created when the issuer first signs up with the system and supplied during the issuer registration process. Typically an 18 digit numeric value.	Required if GroupId not provided
GroupId	A unique identifier for the group of issuers. Created when the group is first created with the system upon system admin request. Typically an 18 digit numeric value.	Required if IssuerId not provided
Type	Determines the type of notification.	Can either be"cardreg", "carddeviceupdate", "cardlock" or "cardoptout".
EncVectorIV	If an encryption keystore has been defined for the issuer, or a group of issuers, critical card data must be encrypted using it. The critical data is encrypted using AES/CBC/PKCS5Padding mode, which requires an IV, including 16 random bytes, as an input parameter for encryption and decryption. The IV should be sent to the server to indicate that the card data in the response should be encrypted in CBC mode. To do this, the IV itself must be encrypted in AES/ECB/ PKCS5Padding mode, using an encryption key, then base64 encoded and set as EncVectorIV in the request.	Optional, if present, the server must encrypt critical card information in CBC mode, otherwise ECB or plain mode will be used instead.
Elements	**Description**	**Usage**

<NotificationRequest>		
<StartDate>	The start date of the period of interest with a format of YYYYMMDDHHMM in GMT.	Required
<EndDate>	The end date of the period of interest with a format of YYYYMMDDHHMM in GMT.	Required

Response

ActiveAccess returns the data in the form of a NotificationResponse message as outlined in the following sections. If the request is unable to be processed entirely, an error code, error message and error details are sent back in the response message.

<NotificationResponse>		
Attributes	**Description**	**Usage**
Id	The same NotificationRequest ID is returned	Required
IssuerId	Specifies the issuer ID that this report is provided for. Should be the same as the IssuerId of the request.	Required if GroupId not provided
GroupId	Specifies the group ID that this report is provided for. Should be the same as the GroupId of the request.	Required if IssuerId not provided
Type	Determines the type of response.	Can be cardreg", "carddeviceupdate", "cardlock" or "cardoptout.

<NotificationResponse>		
EncVectorIV	If the encryption keystore has been defined for the issuer or group of issuerscritical card data is encrypted using AES/ECB/PKCS5Padding or DESede/ECB/PKCS5Padding mode depending on encryption key algorithm. If the EncVectorIV attribute is set to protect the data using AES/CBC/PKCS5Padding or DESede/CBC/PKCS5Padding mode, a new random IV is generated and used by the server to encrypt critical card data in AES/CBC/PKCS5Padding or DESede/CBC/PKCS5Padding mode. When the process is finished, the IV itself is encrypted in AES/ECB/PKCS5Padding or DESede/ECB/PKCS5Padding mode using same encryption key, then base64 encoded and set as EncVectorIV in the response.	Required, if this attribute is set for the request. Server generates a new IV parameter and encrypts critical card information in the response using the CBC mode and the new IV, otherwise no attribute is set and ECB or plain mode will be used instead.
Elements	**Description**	**Usage**
<CardReg>	Contains the list of the cards, which have had registration activity within the specified period.	Required for a card registration notification request
<CardDeviceUpdate>	Contains the list of the cards, which have had device update activity, within the specified period.	Required for a card device update notification request
<CardOptOut>	Contains the list of the cards, which have opted out of their ADS within the specified period.	Required for a card opt-out notification request
<CardLock>	Contains the list of the cards, which have been locked or unlocked within the specified period.	Required for a card lock notification request
<Code>	Response code. Denotes an error has occurred in processing the request.	Required if error occurs
<ErrorMessage>	A descriptive message that identifies the category of the error.	Required if error occurs
<ErrorDetail>	A more detailed description of the error.	Required if error occurs

> **Note**
>
> All dates in notification reports will be in GMT and clients can convert them to their local time if they required.

Notifications

Card Registration Notification

The Card Registration Notification is created when the cardholder changes their registration status to confirmed or registered. A status of confirmed is attained if the issuer is using the confirmation method and the cardholder updates their default authentication password. A status of registered is achieved if the issuer chooses not to use the confirmation method and the cardholder finalises their registration through the enrolment process, Final registration or ADS. ActiveAccess should record the method of registration, time of registration and details of probable registered authentication device.

<CardReg>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**
<Card>	Card related data used to uniquely identify the cardholder including Card Number, Name on Card and Card Type.	Optional, If any related activity reported in the requested period.

> **Note**
>
> If an encryption key is defined for the card issuer/group:
>
> **Card** Number, **Card Name** and Device.**SerialId** of probable registered authentication device are encrypted in response and the client will need to decrypt these fields before using them.
>
> Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

SAMPLE REGISTRATION NOTIFICATION

```
<?xml version="1.0"?>
```

```
<Message>

<NotificationResponse Type="cardreg" Id="notification1"
IssuerId="123456789012345678">

<CardReg>

<Card Type="VbV" Number="4012345678901234" Name="Joe Citizen"
ClientId="123456789012345">

<Device DeviceType="3" SerialId="+61411000001"/>

<Register>20090113080000</Register>

</Card>

</CardReg>

</Notification>

</Message>
```

Card Device Update Notification

The Card Device Update Notification is created when any of the following scenarios occur and the registered devices of the cardholder's account are changed:

- Administrator registers a new device for the specified cardholder via MIA
- Administrator registers an existing device for the specified cardholder via MIA
- Cardholder re-activates its account by registering a new device during authentication
- Cardholder re-activates its account by registering an existing device during authentication
- Administrator unregisters a device of the specified cardholder
- Administrator removes a device of the specified cardholder

<CardDeviceUpdate>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**

<CardDeviceUpdate>		
<Card>	Card related data used to uniquely identify the cardholder including Card Number, Name on Card and Card Type.	Optional, If any related activity reported in the requested period.

Note

If an encryption key is defined for the card issuer/group:

Card Number, **Card Name** and **DeviceUpdate.SerialId** of the device involved, are encrypted in the response and will need to be decrypted before they can be used.

Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

SAMPLE DEVICE UPDATE NOTIFICATION

```
<?xml version="1.0" encoding="UTF-8"?>

<Message>

<NotificationResponse Id="Notification1" IssuerId="123456789012345678"
Type="carddeviceupdate">

<CardDeviceUpdate>

<Card Name="CARD19" Number="4123455000000000009" Type="VbV"
ClientId="123456789012345">

<DeviceUpdate Action="RemoveDevice" Date="20140214052643" DeviceType="SMS"
SerialId="+355665544332211"/>

<DeviceUpdate Action="UnregisterDevice" Date="20140214044825" DeviceType="SMS"
SerialId="+355112233445566"/>

<DeviceUpdate Action="RegisterExistingDevice" Date="20140214044255"
DeviceType="SMS" SerialId="+355665544332211"/>

<DeviceUpdate Action="RegisterNewDevice" Date="20140214044202" DeviceType="SMS"
SerialId="+355112233445566"/>

</Card>

</CardDeviceUpdate>

</NotificationResponse>
```

```
</Message>
```

Card Opt-Out Notification

The Card Opt-Out Notification is created when the cardholder opts-out of the ADS. ActiveAccess should record the date and time of each opt-out. The response should only list the date/time of the most recent opt-out and the sequence number of this opt-out within the requested period and should not list any previous opt-outs which may have occurred during the period.

<CardOptOut>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**
<Card>	Card related data used to uniquely identify the cardholder including Card Number, Name on Card and Card Type.	Optional, If any related activity reported in the requested period.

Note

If an encryption key is defined for the card issuer/group:

Card Number and **Name** are encrypted in response and the client will need to decrypt these fields before using them.

Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

SAMPLE OPT-OUT NOTIFICATION

```
<?xml version="1.0"?>

<Message>

<NotificationResponse Type="cardoptout" Id="notification2"
IssuerId="123456789012345678">

<CardOptOut>

<Card Type="VbV" Number="4012345678901234" Name="Joe Citizen"
```

```
ClientId="123456789012345">
<Optout Number="2">20090131083025</Optout>
</Card>
</CardOptOut>
</Notification>
</Message>
```

Card Lock Notification

The Card Lock Notification is created when the cardholder locks their account during the registration or authentication process. ActiveAccess should record the date and time when a cardholder locks their account and where it is locked (enrolment website or ACS). This should be made available for querying, as with the other notifications.

<CardLock>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**
<Card>	Card related data used to uniquely identify the cardholder including Card Number, Name on Card and Card Type.	Optional, If any related activity reported in the requested period.

Note

If an encryption key is defined for the card issuer/group:

Card Number and **Name** are encrypted in response and the client will need to decrypt these fields before using them.

Refer to section **Cardholder Registration DTD** for further details of the requirements for the encryption/decryption process.

SAMPLE LOCK NOTIFICATION

```
<?xml version="1.0"?>
```

```
<Message>

<NotificationResponse Type="cardlock" Id="notification1"
IssuerId="123456789012345678">

<CardLock>

<Card Type="VbV" Number="4012345678901234" Name="Joe Citizen"
ClientId="123456789012345">

<Lock>20090114093015</Lock>

<Unlock>20090114095015</Unlock>

</Card>

</CardLock>

</Notification>

</Message>
```

User Registration

The authentication system is also responsible for authentication of users during any transactions, commercial or otherwise using two-factor authentication. As a result, it is necessary that the system stores user related information to the extent that this requirement can be satisfied.

There are three user registration models for populating data into the authentication system. Direct entry allows help desk operators to manually enter user registration data through the ActiveAccess administration interface. The final registration and pre-registration models both format user data in an XML message. These messages are either sent directly to the registration server via an API or uploaded through the ActiveAccess administration interface. To facilitate these processes and integrate with the authentication system some integration work with the issuer's systems is required.

Info

See the document, **ActiveAccess Member Bank Administration**, for more information on the Direct Entry user registration model.

The purpose of registration is to upload or supply the information necessary to enable a user to use their two-factor authentication device to gain access to a secured resource such as the issuer's Internet site. For each registered user, the registration message must include a username and the authentication device information.

There are four types of XML messages accepted by the system for the purpose of user creation and maintenance. They are:

- **PreReg**: enables the creation of pre-registered user data in the system
- **FinalReg**: enables the creation of fully registered user data in the system
- **UpdateReg**: enables the alteration of user data in the system
- **CancelReg**: enables the removal of user data from the system

> **Info**
>
> **Section Request** provides more detail on the format of these messages.

To perform authenticated transactions, it is essential that the user data elements stored within the authentication system are sufficient to allow this. The essential elements required are:

- Username
- Device

The information required for user authentication is primarily provided to the authentication system by the member bank. The Registration API provides a flexible data transport and definition mechanism that allows each member bank to build an individual user registration process to meet their specific requirements.

Alternatively, user data can be formatted into XML files and uploaded through the ActiveAccess administration interface. This process reduces the technical requirements for the issuer.

Pre-registration

When using the pre-registration model, a member bank only needs to establish the authentication criteria for enrolment of users and provide this information to the authentication system. The actual enrolment process is then handled by the authentication system's activation during authentication. The authentication system uses the pre-registration information to verify the identity of the user and set-up the user account during the user enrolment process.

In the pre-registration model, it is essential that a **PreReg** message uploads Username. It is also required that the issuer supplies a number of other known parameters to the authentication system to verify the user when they come to enrol with the authentication system. The following provides a list of suggested data that can be included in the **PreReg** message to allow the user authentication to occur. Please note that these fields are at the discretion of the issuing bank:

- Date of birth
- Mother's maiden name
- Internet Banking username
- Information about accounts held by the user (e.g. account types, credit limit, billing address, etc)

An example of pre-registration is when the member bank mails invitations to users and provides them with, say, a registration number. Users can then enrol using the activation during shopping process. Having entered their registration number and once the user's identity is successfully verified, they can proceed with setting up their account, which includes selection of device specification. Once the enrolment process is complete, the tokens generated by the enrolled authentication device will then be used in all subsequent authenticated transactions.

Final Registration

The final registration model allows issuers to control the user's enrolment process. It requires the member bank to develop their own enrolment process to collect user enrolment information. Once the user enrolment process is complete, the information is sent to the authentication system in the form of a final registration message.

In the final registration model, it is essential that a **FinalReg** message uploads the Username. It is also required that the issuer supplies the specifications of device which is used at authentication time for generating tokens:

- Device Type
- Device Serial No

An example of final registration is enrolment of users through the issuer's Internet banking site. A possible scenario is for the user to log into the Internet banking site and enable two-factor authentication for their account. The user will then be prompted to choose device type and its serial number for their account. The issuer then formats a final registration message, which is sent to the authentication system in order to complete the user enrolment.

The format of the registration request is the same for real-time and batch uploads, except that the batch upload may contain multiple blocks of user data.

The registration API uses XML as the message format and HTTP as the message transport. API calls (requests) can be made to the registration server of the authentication system. Issuers can use the registration API to automate their user registration process.

XML Message Format

XML messages must be produced in accordance with the Messaging Requirements, as specified in section User Registration DTD .

All request and response messages are enclosed by the <Message> element. A message can contain either a Request and a Signature or a Response.

A message can contain either a Request and a Signature or a Response

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456789">

<!--the request content here-->

</request>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<!--the request signature here-->

</Signature>

</Message>
```

<Message>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**

<Message>		
<Request>	Used in the request message, which is sent by the issuer to register one or more users.	Required for request messages
<Signature>	Issuer signature is used in the request message to prove the identity of the issuer and to validate that the message content has not been altered.	Required for request message
<Response>	User registration response sent back in response to a user registration request.	Required for response message

Note

The XML response starts with the XML declaration (<?xml version="1.0" encoding="UTF-8"?>). However a request does not need to start with the XML declaration. Request content must be sent with UTF-8 encoding.

Request

A Request is sent by the issuer to perform various user registration tasks. A request can be sent to perform pre-registration, final registration, cancel registration or update registration for one or more users. A request may only contain one <PreReg>, <FinalReg>, <CancelReg> or <UpdateReg>.

<Request>		
Attributes	**Description**	**Usage**
Id	An arbitrary identifier, which the issuer defines and can be used to refer to the request element as part of a standard URI. Also referenced by the Signature element. XML signature requires the element, which is being signed, to be identified by a unique The value entered should start with an alphabetic character.	Required
IssuerId	A unique identifier for the issuer. Created when the issuer first signs up with the system and supplied during the issuer registration process. Typically an 18 digit numeric value.	Required
Elements	**Description**	**Usage**

<Request>		
<PreReg>	Used for pre-registration of one or more users. Pre-registered Users will need to go through the enrolment process to finalise their registration.	Required for a pre-registration request
<FinalReg>	Used for registration of one or more users. Registered Users can start making authenticated transactions without the need to go through the enrolment process.	Required for a final registration request
<CancelReg>	Used to remove one or more users from the system.	Required for a cancel registration request
<UpdateReg>	Used to update the information for one or more users. For example to change the Username. User's registration status is left unchanged.	Required for an update registration request

Response

A response message is sent back for each request. The response message provides the result of the request message with details of errors, if any. Issuers must process the response message and should correct and replay their request if there is an error.

<Response>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**
<Code>	Response code. 0 if the request was successful. 1 if the request has been successfully processed but there are warnings. Any other value denotes an error in processing the request.	Required
<ErrorMessage>	A descriptive message that identifies the category of the error.	Required
<ErrorDetail>	A more detailed description of the error.	Required

<Response>		
<Warning>	A warning is issued to provide information on an unexpected situation that does not prevent the request from being successfully processed.	Conditional. Required only if response code is 1.

Note

A message with response code **1** denotes that the request has been successfully processed but yet the registration server has not been able to comply with certain instructions. For example a cancel registration attempt on an already cancelled user is not processed and as such a warning message is reported. An issuer does not need to take any further action in this case. Warning messages may be logged at the issuer's end for later reference.

Note

Multiple warning messages may be included in a single response message.

Requests

Pre-registration Request

The pre-registration request is typically used by the members who wish to leave user enrolment to authentication time. A pre-registered user cannot perform authenticated transactions until they finalise the registration by going through the standard enrolment process, which is offered by the user's activation during authentication process. Pre-registration data is used to verify the identity of users during the enrolment process.

<PreReg>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**

<PreReg>		
<DataFormat>	Defines a data type, which can be associated with user data. This issuer-defined field has a maximum field length of 1024 characters. You can define the following attributes with the DataFormat: **Name:** The name as used by the program to refer to this data format. Also used by the data element in order to refer to this particular type, e.g.: pass or password **Label:** A short description to appear before the data elements of this type when displayed to the cardholder, e.g.: Birth Date: **Description:** A longer description to appear after the data elements of this type when displayed to the cardholder (optional), e.g.: Please enter your birth date **MaxLen:** The maximum length that can be stored in the data elements of this type (optional). **Type:** Can be set to date, string, number or hidden values. When set to date, number, the application performs type validation for the content of the data elements. If not specified the string type is assumed (no validation). If the data format type is set to hidden, this excludes the data format as an authentication data element. Hidden data types are not displayed to the users and can be used as internal field in the XSL pages per bank to achieve certain customised features. You should not use this type unless specifically advised by GPayments to do so **Format:** Additional formatting information can be set to YYYYDDMM or YYYYMM. This is only meaningful when you have set the data format type to date. **Mask:** Can be set to Yes or No. Determines whether the user input for this data format should be masked or not. Set to Yes for password type fields.	Required A valid data format must be defined for each type, which is referenced by a <Data> element.
<UserReg>	User related data including username, name, password and issuer defined data. Usernames up to 128 characters are accepted and the name field has a maximum length of 256 characters.	Required At least one <UserReg> must be present

The pre-registration data may include existing customer information such as date of birth, mother's maiden name, card verification check, credit limit, billing address, etc or a registration number distributed by mail (or in some cases a combination of both). The requirements for pre-registration information are at the discretion of member banks. The pre-registration request is dynamic enough to meet the authentication requirements of each individual member bank.

Member banks, which require greater flexibility in the presentation or control of the enrolment process, should use the final registration model.

The pre-registration data may be removed once a user has successfully completed the enrolment process.

Multiple unsuccessful enrolment attempts may cause the user to be locked. This limit can be set on a per issuer basis and through the ActiveAccess administration interface.

To disable a user registration, a cancel registration request should be used. To un-register finally registered users, the user account should be cancelled first and a new pre-registration message should be sent. This will require the user to repeat the enrolment process.

A sample pre-registration request is displayed below. The following request will provide pre-registration information, which is required for enrolment of Mr. Joe Citizen:

SAMPLE PRE-REGISTRATION REQUEST

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456">

<PreReg>

<DataFormat Name="birthdate" Type="date" Format="YYYYMMDD" Label="Date of
Birth:"/>

<DataFormat Name="regpassword" Type="string" Label="Registration Password:"
Desc="Please enter your registration password" Mask="Yes"/>

<UserReg Username="citizenjoe">

<Name>Mr. Joe Citizen</Name>

<Password>123456</Password>

<Data Name="birthdate" Value="19730201"/>

<Data Name="regpassword" Value="pro2345"/>

</UserReg>

</PreReg>

</Request>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<!--the request signature here-->
```

```
</Signature>

</Message>
```

Final Registration Request

Members who wish to have greater control over the user enrolment process typically use the final registration request. In this case the member bank itself handles the enrolment process. The registration process should result in the setup of a device between the member bank and the user. The member bank will then format a final registration request and provide the agreed authentication information to the enrolment module for storage.

<FinalReg>		
Attributes	**Description**	**Usage**
None	N/A	N/A
Elements	**Description**	**Usage**
<UserReg>	User related data including username, name, an optional password and device information. Usernames with maximum 128 characters and names that has a maximum length of 256 characters are accepted. Device information includes, Device type {1: VASCO, 3: SMS, 7: OOB, 6: Email, 8: Decoupled Authentication} specifies type of device which will be used for generating tokens and serial no which is a unique identifier assigned by the device manufacturer.	Required At least one <UserReg> must be present

Users registered using the final registration method can perform authenticated transactions as soon as the registration request is processed, without the need to go through activation during authentication process.

Note that the 'Password' field is optional and is only used when both the first and the second factor of authentication is to be handled by ActiveAccess. Typically an issuer would use their existing first factor authentication (e.g. Internet banking password) to authenticate the user themselves and would only rely on ActiveAccess for second factor authentication.

The following request will provide final registration information, which is required for Mr. Joe Citizen.

SAMPLE FINAL REGISTRATION REQUEST

```
<?xml version="1.0"?>
<Message>
<Request Id="request1" IssuerId="1234567890123456789">
<FinalReg>
<UserReg Username="citizenjoe">
<Name>Mr. Joe Citizen</Name>
<Password>123456</Password>
<Device>
<DeviceType>1</DeviceType>
<SerialNo>0097123456</SerialNo>
</Device>
</UserReg>
</FinalReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

Cancel Registration Request

Cancel registration can be used to remove user data (either pre-registration or full registration data). A user that has been cancelled can no longer be used in authentication process.

Cancel registration is typically used when a user's account is closed or cancelled. A **CancelReg** request should consist of simple blocks of user data. The enrolment module ignores the content of the <User> element, as it only requires the Username for cancelling the user's registration.

The following listing cancels the registration of Mr. Joe Citizen.

SAMPLE CANCEL REGISTRATION REQUEST

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456789">

<CancelReg>

<User Username="citizenjoe"></User>

</CancelReg>

</Request>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<!--the request signature here-->

</Signature>

</Message>
```

Update Registration Request

An update registration is used to update user information such as username, name and password. It is used to update user data without affecting the user's enrolment status. For example if the user's username has changed it is possible to save the user from going through the enrolment process again by simply updating the username.

To update user authentication data, use pre-registration or final registration request.

The following listing shows how Mr. Joe Citizen's username and name can be updated. Mr. Citizen will be able to continue making authenticated logins without the need to re-enrol.

SAMPLE UPDATE REGISTRATION REQUEST

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="1234567890123456789">

<UpdateReg>
```

```
<UserUpdate Username="Joe Citizen">
<Name>Mr. Ko Citizen</Name>
<Username>citizenco</Username>
<Password>123456</Password>
</UserUpdate>
</UpdateReg>
</Request>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--the request signature here-->
</Signature>
</Message>
```

Messaging Requirements

Signing the Message

A request must be always accompanied by a Signature element to determine the authenticity of the message. Issuers are required to sign the <Request>….</Request> element of their message. As the registration server is not a general XML signature verifier tool, it does not resolve the specified URI in <Reference> of the signed information and always assumes that the client has signed the <Request>….</Request> element.

The following table summarises the algorithms that should be used for signing the XML message and their reference.

Algorithm	Reference
Canonicalization	http://www.w3.org/TR/2001/REC-xml-c14n-20010315
Message Digest	http://www.w3.org/2000/09/xmldsig#sha1
Encoding	http://www.w3.org/2000/09/xmldsig#base64

Algorithm	Reference
Signature	http://www.w3.org/2000/09/xmldsig#rsa-sha1
MAC	http://www.w3.org/2000/09/xmldsig#hmac-sha1

Info

Please refer to XML-Signature Syntax and Processing, W3C Proposed Recommendation, dated 20 August 2001 at http://www.w3.org/TR/2001/PR-xmldsig-core-20010820 for further information.

Example of a message with standard XML signature

```
<?xml version="1.0"?>

<Message>

<Request Id="request1" IssuerId="123456789012345678">

<!--the request content here-->

</Request>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>

<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315">

</CanonicalizationMethod>

<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">

</SignatureMethod>

<Reference URI="#request1">

<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">

</DigestMethod>

<DigestValue>pHfyjnLJ2LK0Vc4cLgYSFp8gGhM=</DigestValue>

</Reference>
```

```
</SignedInfo>

<SignatureValue>

eHNXORO0egBEFqYt16zOtXG4FaraIEfCxM5cZ2QYCCl3tqbx9ynF6DmOdlLymaR0kBdkIAWS6uYC3Tg3Z8t
9i+ze4veCZLfHsXbJsvHxcAsF/kRJWmDCfpSrApbKqIkmAPWpw3W8hxF950gggYWkXOCSUIw4C1Vc=

</SignatureValue>

<KeyInfo>

<X509Data>

<X509Certificate>

MIIDwjCCA2ygAwIBAgIIGtU11QAAAGEwDQYJKoZIhvcNAQEEBQAwfjELMAkGA1UEBhMCQVUxGDAWBgNVBAg
V0aCBXYWxlczEPMA0GA1UEBxMGU3lkbmV5MRowGAYDVQQKExFHUGF5bWVudHMgUHR5IEx0ZDEUMBIGA1UEC
dmljZXMxEjAQBgNVBAMTCUdQYXltZW50czAeFw0wMjEyMDUyMzU5MTJaFw0wNjExMjkwNTM1NDRaMHYxCzA
FVMRgwFgYDVQQIEw9OZXcgU291dGggV2FsZXMxDzANBgNVBAcTBlN5ZG5leTESMBAGA1UEChMJR1BheW1lb
VQQLEwtJVCBTZXJ2aWNlczESMBAGA1UEAxMJMTI3LjAuMC4xMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQK
+EfXoR
8yljNopWKm/nnEqyBkghJc9xuO+uVkYqPTuuIK6KFVxHGU+BT3+SAtP2K5MQIUCpi9c6/
yc6wrYfFvHyW9nf6LixAsAeZ2
DiMCZfD1TUwoA0FOjwEcBollj4SkqOnZia9kuHVpkhLiOxxHJMgXuIymyfWDChikH+/
4LwIDAQABo4IBkDCCAYwwgbkGA1
UdIwSBsTCBroAUqG0JqSygT3QCMlmFHp+x41MjfzChgYOkgYAwfjELMAkGA1UEBhMCQVUxGDAWBgNVBAgTD
aCBXYWxlczEPMA0GA1UEBxMGU3lkbmV5MRowGAYDVQQKExFHUGF5bWVudHMgUHR5IEx0ZDEUMBIGA1UECxM
ljZXMxEjAQBgNVBAMTCUdQYXltZW50c4IQazV34VzY/rxAdU/
vkRbzrjB5BgNVHR8EcjBwMDWgM6Axhi9odHRwOi8vVklT
QURJUi9DZXJ0U3J2L0NlcnRFbnJvbGwvR1BheW1lbnRzLmNybDA3oDWgM4YxZmlsZTovL1xcVklTQURJUlx
NlcnRFbnJvbGxcR1BheW1lbnRzLmNybDBTBggrBgEFBQcBAQRHMEUwQwYIKwYBBQUHMAKGN2h0dHA6Ly9WS
cnRTcnYvQ2VydEVucm9sbC9WSVNBRElSX0dQYXltZW50cy5jcnQwDQYJKoZIhvcNAQEEBQADQQCrPtHfPVp
5
EXYiwDxJg1HSDrxi7Kgvf0jQD18uz6m48BQ2Pb/wQX/eMkXcQlOIAit/K7tHD8A4wG

</X509Certificate>

</X509Data>

</KeyInfo>

</Signature>

</Message>
```

The registration API ignores the content of <X509Certificate> and does not use this element for verification of the <SignatureValue>. The DTD requires at least one of the many <X509Data> elements such as <X509Certificate> to be specified however, you may leave the content of <X509Certificate> empty

(<X509Certificate> </X509Certificate>).

XML Signing Certificate

The verification of the message signature is based on the XML signing certificate. The member bank submits this certificate as part of the banks testing and enrolment process. After it is received, it is stored in the issuer's profile in the authentication system database. This is a two-party scenario whereby the authentication system operator verifies the identity of the member bank before accepting the certificate. As such there is no need for the XML signing certificate to be signed by a third party and member banks can submit a self-signed certificate.

Critical Card Data Encryption and Decryption

The key, which is used for encrypting/decrypting the critical card data, must be a 128 bit AES key. A KeyStore with the following details should be prepared for the encryption key that is to be uploaded, through MIA, for the specified issuer or group of issuers:

KeyStore type/format: JCEKS

KeyStore provider: SunJCE

Key algorithm: AES

Key size: 128 bit

Key name: can be any

No of keys in the KeyStore: Only one key must be populated in the KeyStore

Such KeyStores can be easily created by the Java keytool utility using the following command:

```
keytool -genseckey -alias enckey128 -keypass 123456 -keyalg AES -keysize 128
-keystore enc-key.JKS -storepass 123456 -storetype JCEKS
```

In order to facilitate a smooth transition to the latest version of CardLoader, keys with the following specifications are also supported:

KeyStore type/format: JCEKS

KeyStore provider: SunJCE

Key algorithm: DESede

Key size: 112 or 168 bit

Key name: can be any

If EncVectorIV is set for the request/response, the registration server/client needs to get the IV by base64 decoding and decrypting the EncVectorIV using the encryption key in DESede/ECB/PKCS5Padding or AES/ECB/PKCS5Padding mode, before decrypting the critical card data in DESede/CBC/PKCS5Padding or AES/CBC/PKCS5Padding mode using the obtained EncVectorIV from the request/response.

Cardholder Registration

In cardholder registration messages, critical cardholder information must be encrypted using AES/ECB/PKCS5Padding or AES/CBC/PKCS5Padding mode (if EncVectorIV is used for the request) in ActiveAccess v8+, and using DESede/ECB/PKCS5Padding or DESede/CBC/PKCS5Padding mode (if EncVectorIV is used for the request), and critical cardholder information must be encrypted in older versions of ActiveAccess. The output must then be base64 encoded and included in the message.

If a card encounters a warning during the process, its **Card Number**, **Name** and **Device.SerialNo** (if final registration request contains device information) would be encrypted in the response. In this case, if EncVectorIV has been set for the request, the server generates a new IV, encrypts it using DESede/ECB/PKCS5Padding or AES/ECB/PKCS5Padding mode, base64 encodes it and then sets it in the response. If the server sets EncVectorIV for the response, the IV must be obtained by base64 decoding and decrypting the EncVectorIV using the encryption key in DESede/ECB/PKCS5Padding or AES/ECB/PKCS5Padding mode before decrypting the critical card data in the response. Critical cardholder information must then be decrypted using DESede/ECB/PKCS5Padding or DESede/CBC/PKCS5Padding and AES/ECB/PKCS5Padding or AES/CBC/PKCS5Padding (If EncVectorIV has been set for the response) mode, depending on encryption key algorithm.

Notification

In notification responses, critical cardholder information must be encrypted using DESede/ECB/PKCS5Padding or DESede/CBC/PKCS5Padding and AES/ECB/PKCS5Padding or AES/CBC/PKCS5Padding (If EncVectorIV is used for the request) mode, depending on encryption key algorithm. The output must then be base64 encoded and included in the message.

Card Number, **Name**, **Device.SerialId** and **DeviceUpdate.SerialId** would be encrypted in the response. In this case, if EncVectorIV has been set for the request, the server generates a new IV,

encrypts it using DESede/ECB/PKCS5 Padding or AES/ECB/PKCS5Padding mode, base64 encodes it and then sets it in the response. If the server sets EncVectorIV for the response, the IV must be obtained by base64 decoding and decrypting the EncVectorIV, using the encryption key in DESede/ECB/PKCS5 Padding or AES/ECB/PKCS5Padding mode, before decrypting the critical card data in the response. Critical cardholder information must then be decrypted using DESede/ECB/PKCS5 Padding or DESede/CBC/PKCS5Padding and AES/ECB/PKCS5Padding or AES/CBC/PKCS5Padding (If EncVectorIV has been set for the response) mode, depending on encryption key algorithm.

Calling Convention

Requests must be sent using HTTPS POST. The HTTP header must have a Content-Length, which should be set to the body length of the request.

Registration API DTD

Issuers must make sure that their XML request conforms to the requirements of the registration API by validating the request against the appropriate DTD.

Cardholder Registration DTD

Validate the request against the following DTD.

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT Message ((Request, Signature) | Response | (GetResponse, Signature) |
GetProgress | Progress)>
<!ATTLIST Message
    Id CDATA #IMPLIED
>
<!ELEMENT Request (PreReg | FinalReg | CancelReg | UpdateReg | DeviceUpdateReg)>
<!ATTLIST Request
    IssuerId NMTOKEN #IMPLIED
    GroupId NMTOKEN #IMPLIED
    Id ID #REQUIRED
    EncVectorIV CDATA #IMPLIED
>
<!ATTLIST GetResponse
    Id ID #REQUIRED
    EncVectorIV CDATA #IMPLIED
>
<!ATTLIST Response
```

```
    EncVectorIV CDATA #IMPLIED
>
<!ELEMENT Response (Code, ErrorMessage, ErrorDetail, Warning*)>
<!ELEMENT GetResponse (IssuerId?, GroupId?)>
<!ELEMENT GetProgress (IssuerId?, GroupId?)>
<!ELEMENT PreReg (DataFormat*, Card+)>
<!ELEMENT FinalReg (DataFormat*, Card+)>
<!ELEMENT CancelReg (Card+)>
<!ELEMENT UpdateReg (CardUpdate*,CardCopy*)>
<!ELEMENT Card (ClientId*, ExpDate?, PAM?, HINT?, HINTResponse?, Data*, Device*)>
<!ELEMENT Device (DeviceType, SerialNo?, Param*)>
<!ATTLIST Card
    Type (SPA | VbV | JCB | SK | DC) #REQUIRED
    Number CDATA #REQUIRED
    Name CDATA #IMPLIED
>
<!ELEMENT CardUpdate (Number?, Name?, ClientId*, ExpDate?, PAM?, HINT?,
HINTResponse?, Status?, Data*)>
<!ATTLIST CardUpdate
    Number CDATA #REQUIRED
    Name CDATA #REQUIRED
>
<!ELEMENT CardCopy (Type?, Number?, Name?, ClientId*, ExpDate?, PAM?, HINT?,
HINTResponse?, Status?, Data*)>
<!ATTLIST CardCopy
    Number CDATA #REQUIRED
    Name CDATA #REQUIRED
>
<!ELEMENT DeviceUpdateReg (CardDeviceUpdate+)>
<!ELEMENT CardDeviceUpdate (DeviceUpdate+)>
<!ATTLIST CardDeviceUpdate
    Number CDATA #IMPLIED
    Name CDATA #IMPLIED
    ClientId CDATA #IMPLIED
>
<!ELEMENT DeviceUpdate (Device+)>
<!ATTLIST DeviceUpdate
    Operation (Add | Delete) #REQUIRED
>
<!ATTLIST Param
    Name NMTOKEN #REQUIRED
>
<!ELEMENT IssuerId (#PCDATA)>
<!ELEMENT GroupId (#PCDATA)>
<!ELEMENT Progress (#PCDATA)>
<!ELEMENT Code (#PCDATA)>
<!ELEMENT ErrorMessage (#PCDATA)>
<!ELEMENT ErrorDetail (#PCDATA)>
<!ELEMENT Warning (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT ExpDate (#PCDATA)>
```

```
<!ELEMENT ClientId (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Number (#PCDATA)>
<!ELEMENT PAM (#PCDATA)>
<!ELEMENT HINT (#PCDATA)>
<!ELEMENT HINTResponse (#PCDATA)>
<!ELEMENT DeviceType (#PCDATA)>
<!ELEMENT Status (#PCDATA)>
<!ELEMENT SerialNo (#PCDATA)>
<!ELEMENT Param (#PCDATA)>
<!ELEMENT SelectedOption EMPTY>
<!ATTLIST SelectedOption
    Value NMTOKEN #REQUIRED
>
<!ELEMENT Data (SelectedOption*)>
<!ATTLIST Data
    Name NMTOKEN #REQUIRED
    Value CDATA #IMPLIED
    Discard (Yes | yes | No | no) #IMPLIED
>
<!ELEMENT Option EMPTY>
<!ATTLIST Option
    Label CDATA #REQUIRED
    Value NMTOKEN #REQUIRED
>
<!ELEMENT DataFormat (Option*)>
<!ATTLIST DataFormat
    Name NMTOKEN #REQUIRED
    Label CDATA #REQUIRED
    Desc CDATA #IMPLIED
    MaxLen NMTOKEN #IMPLIED
    Type (date | string | number | hidden | singleSelect | multiSelect) #IMPLIED
    Format (YYYYMMDD | YYYYMM) #IMPLIED
    Mask (Yes | yes | No | no) #IMPLIED
    DataMode (Auth | Identity | Extension | auth | identity | extension) #IMPLIED
>
<!-- DTD for XML Signatures http://www.w3.org/2000/09/xmldsig# -->
<!ENTITY % Object.ANY ''>
<!ENTITY % Method.ANY ''>
<!ENTITY % Transform.ANY ''>
<!ENTITY % SignatureProperty.ANY ''>
<!ENTITY % KeyInfo.ANY ''>
<!ENTITY % KeyValue.ANY ''>
<!ENTITY % PGPData.ANY ''>
<!ENTITY % X509Data.ANY ''>
<!ENTITY % SPKIData.ANY ''>
<!-- Start Core Signature declarations, these should NOT be altered -->
<!ELEMENT Signature (SignedInfo, SignatureValue, KeyInfo?, Object*)>
<!ATTLIST Signature
    xmlns CDATA #FIXED "http://www.w3.org/2000/09/xmldsig#"
    Id ID #IMPLIED
```

```
>
<!ELEMENT SignatureValue (#PCDATA)>
<!ATTLIST SignatureValue
    Id ID #IMPLIED
>
<!ELEMENT SignedInfo (CanonicalizationMethod, SignatureMethod, Reference+)>
<!ATTLIST SignedInfo
    Id ID #IMPLIED
>
<!ELEMENT CanonicalizationMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST CanonicalizationMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT SignatureMethod (#PCDATA|HMACOutputLength %Method.ANY;)* >
<!ATTLIST SignatureMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT Reference (Transforms?, DigestMethod, DigestValue)>
<!ATTLIST Reference
    Id ID #IMPLIED
    URI CDATA #IMPLIED
    Type CDATA #IMPLIED
>
<!ELEMENT Transforms (Transform+)>
<!ELEMENT Transform (#PCDATA|XPath %Transform.ANY;)* >
<!ATTLIST Transform
    Algorithm CDATA #REQUIRED
>
<!ELEMENT XPath (#PCDATA)>
<!ELEMENT DigestMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST DigestMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT DigestValue (#PCDATA)>
<!ELEMENT KeyInfo (#PCDATA|KeyName|KeyValue|RetrievalMethod|
X509Data|PGPData|SPKIData|MgmtData %KeyInfo.ANY;)* >
<!ATTLIST KeyInfo
    Id ID #IMPLIED
>
<!-- Key Information -->
<!ELEMENT KeyName (#PCDATA)>
<!ELEMENT KeyValue (#PCDATA|DSAKeyValue|RSAKeyValue %KeyValue.ANY;)* >
<!ELEMENT MgmtData (#PCDATA)>
<!ELEMENT RetrievalMethod (Transforms?)>
<!ATTLIST RetrievalMethod
    URI CDATA #REQUIRED
    Type CDATA #IMPLIED
>
<!-- X.509 Data -->
<!ELEMENT X509Data ((X509IssuerSerial | X509SKI | X509SubjectName |
X509Certificate | X509CRL )+ %X509Data.ANY;)>
```

```
<!ELEMENT X509IssuerSerial (X509IssuerName, X509SerialNumber)>
<!ELEMENT X509IssuerName (#PCDATA)>
<!ELEMENT X509SubjectName (#PCDATA)>
<!ELEMENT X509SerialNumber (#PCDATA)>
<!ELEMENT X509SKI (#PCDATA)>
<!ELEMENT X509Certificate (#PCDATA)>
<!ELEMENT X509CRL (#PCDATA)>
<!-- PGPData -->
<!ELEMENT PGPData ((PGPKeyID, PGPKeyPacket?) | (PGPKeyPacket) %PGPData.ANY;)
>
<!ELEMENT PGPKeyPacket (#PCDATA)>
<!ELEMENT PGPKeyID (#PCDATA)>
<!-- SPKI Data -->
<!ELEMENT SPKIData (SPKISexp %SPKIData.ANY;) >
<!ELEMENT SPKISexp (#PCDATA)>
<!-- Extensible Content -->
<!ELEMENT Object (#PCDATA|Signature|SignatureProperties|Manifest
%Object.ANY;)* >
<!ATTLIST Object
    Id ID #IMPLIED
    MimeType CDATA #IMPLIED
    Encoding CDATA #IMPLIED
>
<!ELEMENT Manifest (Reference+)>
<!ATTLIST Manifest
    Id ID #IMPLIED
>
<!ELEMENT SignatureProperties (SignatureProperty+)>
<!ATTLIST SignatureProperties
    Id ID #IMPLIED
>
<!ELEMENT SignatureProperty (#PCDATA %SignatureProperty.ANY;)* >
<!ATTLIST SignatureProperty
Target CDATA #REQUIRED
    Id ID #IMPLIED
>
<!-- Algorithm Parameters -->
<!ELEMENT HMACOutputLength (#PCDATA)>
<!ELEMENT DSAKeyValue ((P, Q)?, G?, Y, J?, (Seed, PgenCounter)?)>
<!ELEMENT P (#PCDATA)>
<!ELEMENT Q (#PCDATA)>
<!ELEMENT G (#PCDATA)>
<!ELEMENT Y (#PCDATA)>
<!ELEMENT J (#PCDATA)>
<!ELEMENT Seed (#PCDATA)>
<!ELEMENT PgenCounter (#PCDATA)>
<!ELEMENT RSAKeyValue (Modulus, Exponent)>
<!ELEMENT Modulus (#PCDATA)>
<!ELEMENT Exponent (#PCDATA)>
```

User Registration DTD

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT Message ((Request, Signature) | Response | (GetResponse, Signature)
| GetProgress | Progress)>
<!ATTLIST Message
    Id CDATA #IMPLIED
>
<!ELEMENT Request (FinalReg | PreReg | CancelReg | UpdateReg)>
<!ATTLIST Request
    IssuerId NMTOKEN #REQUIRED
    Id ID #REQUIRED
>
<!ELEMENT Response (Code, ErrorMessage, ErrorDetail, Warning*)>
<!ELEMENT GetResponse (IssuerId)>
<!ELEMENT GetProgress (IssuerId)>
<!ELEMENT PreReg (DataFormat*, UserReg+)>
<!ELEMENT FinalReg (UserReg+)>
<!ELEMENT UpdateReg (UserUpdate+)>
<!ELEMENT CancelReg (User+)>
<!ELEMENT UserReg (Name?, Password?, Data*, Device*)>
<!ATTLIST UserReg
    Username CDATA #REQUIRED
>
<!ELEMENT UserUpdate (Name?, Username?, Password?)>
<!ELEMENT Device (DeviceType, SerialNo?, Param*)>
<!ATTLIST UserUpdate
    Username CDATA #REQUIRED
>
<!ATTLIST User
    Username CDATA #REQUIRED
>
<!ATTLIST Data
    Name NMTOKEN #REQUIRED
    Value CDATA #REQUIRED
>
<!ATTLIST DataFormat
    Name NMTOKEN #REQUIRED
    Label CDATA #REQUIRED
    Desc CDATA #IMPLIED
    MaxLen NMTOKEN #IMPLIED
    Type (date | string | number | hidden) #IMPLIED
    Format (YYYYMMDD | YYYYMM) #IMPLIED
    Mask (Yes | yes | No | no) #IMPLIED
>
<!ATTLIST Param
    Name NMTOKEN #REQUIRED
>
```

```
<!ELEMENT IssuerId (#PCDATA)>
<!ELEMENT Progress (#PCDATA)>
<!ELEMENT Code (#PCDATA)>
<!ELEMENT ErrorDetail (#PCDATA)>
<!ELEMENT ErrorMessage (#PCDATA)>
<!ELEMENT Warning (#PCDATA)>
<!ELEMENT User EMPTY>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Username (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!ELEMENT Data EMPTY>
<!ELEMENT DataFormat EMPTY>
<!ELEMENT DeviceType (#PCDATA)>
<!ELEMENT SerialNo (#PCDATA)>
<!ELEMENT Param (#PCDATA)>
<!-- DTD for XML Signatures http://www.w3.org/2000/09/xmldsig# -->
<!ENTITY % Object.ANY ''>
<!ENTITY % Method.ANY ''>
<!ENTITY % Transform.ANY ''>
<!ENTITY % SignatureProperty.ANY ''>
<!ENTITY % KeyInfo.ANY ''>
<!ENTITY % KeyValue.ANY ''>
<!ENTITY % PGPData.ANY ''>
<!ENTITY % X509Data.ANY ''>
<!ENTITY % SPKIData.ANY ''>
<!-- Start Core Signature declarations, these should NOT be altered -->
<!ELEMENT Signature (SignedInfo, SignatureValue, KeyInfo?, Object*)>
<!ATTLIST Signature
    xmlns CDATA #FIXED "http://www.w3.org/2000/09/xmldsig#"
    Id ID #IMPLIED
>
<!ELEMENT SignatureValue (#PCDATA)>
<!ATTLIST SignatureValue
    Id ID #IMPLIED
>
<!ELEMENT SignedInfo (CanonicalizationMethod, SignatureMethod, Reference+)>
<!ATTLIST SignedInfo
    Id ID #IMPLIED
>
<!ELEMENT CanonicalizationMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST CanonicalizationMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT SignatureMethod (#PCDATA|HMACOutputLength %Method.ANY;)* >
<!ATTLIST SignatureMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT Reference (Transforms?, DigestMethod, DigestValue)>
<!ATTLIST Reference
    Id ID #IMPLIED
    URI CDATA #IMPLIED
```

```
    Type CDATA #IMPLIED
>
<!ELEMENT Transforms (Transform+)>
<!ELEMENT Transform (#PCDATA|XPath %Transform.ANY;)* >
<!ATTLIST Transform
    Algorithm CDATA #REQUIRED
>
<!ELEMENT XPath (#PCDATA)>
<!ELEMENT DigestMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST DigestMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT DigestValue (#PCDATA)>
<!ELEMENT KeyInfo (#PCDATA|KeyName|KeyValue|RetrievalMethod|
X509Data|PGPData|SPKIData|MgmtData %KeyInfo.ANY;)* >
<!ATTLIST KeyInfo
    Id ID #IMPLIED
>
<!-- Key Information -->
<!ELEMENT KeyName (#PCDATA)>
<!ELEMENT KeyValue (#PCDATA|DSAKeyValue|RSAKeyValue %KeyValue.ANY;)* >
<!ELEMENT MgmtData (#PCDATA)>
<!ELEMENT RetrievalMethod (Transforms?)>
<!ATTLIST RetrievalMethod
    URI CDATA #REQUIRED
    Type CDATA #IMPLIED
>
<!-- X.509 Data -->
<!ELEMENT X509Data ((X509IssuerSerial | X509SKI | X509SubjectName |
X509Certificate | X509CRL )+ %X509Data.ANY;)>
<!ELEMENT X509IssuerSerial (X509IssuerName, X509SerialNumber)>
<!ELEMENT X509IssuerName (#PCDATA)>
<!ELEMENT X509SubjectName (#PCDATA)>
<!ELEMENT X509SerialNumber (#PCDATA)>
<!ELEMENT X509SKI (#PCDATA)>
<!ELEMENT X509Certificate (#PCDATA)>
<!ELEMENT X509CRL (#PCDATA)>
<!-- PGPData -->
<!ELEMENT PGPData ((PGPKeyID, PGPKeyPacket?) | (PGPKeyPacket) %PGPData.ANY;)
>
<!ELEMENT PGPKeyPacket (#PCDATA)>
<!ELEMENT PGPKeyID (#PCDATA)>
<!-- SPKI Data -->
<!ELEMENT SPKIData (SPKISexp %SPKIData.ANY;) >
<!ELEMENT SPKISexp (#PCDATA)>
<!-- Extensible Content -->
<!ELEMENT Object (#PCDATA|Signature|SignatureProperties|Manifest
%Object.ANY;)* >
<!ATTLIST Object
    Id ID #IMPLIED
    MimeType CDATA #IMPLIED
```

```
    Encoding CDATA #IMPLIED
>
<!ELEMENT Manifest (Reference+)>
<!ATTLIST Manifest
    Id ID #IMPLIED
>
<!ELEMENT SignatureProperties (SignatureProperty+)>
<!ATTLIST SignatureProperties
    Id ID #IMPLIED
>
<!ELEMENT SignatureProperty (#PCDATA %SignatureProperty.ANY;)* >
<!ATTLIST SignatureProperty
    Target CDATA #REQUIRED
    Id ID #IMPLIED
>
<!-- Algorithm Parameters -->
<!ELEMENT HMACOutputLength (#PCDATA)>
<!ELEMENT DSAKeyValue ((P, Q)?, G?, Y, J?, (Seed, PgenCounter)?)>
<!ELEMENT P (#PCDATA)>
<!ELEMENT Q (#PCDATA)>
<!ELEMENT G (#PCDATA)>
<!ELEMENT Y (#PCDATA)>
<!ELEMENT J (#PCDATA)>
<!ELEMENT Seed (#PCDATA)>
<!ELEMENT PgenCounter (#PCDATA)>
<!ELEMENT RSAKeyValue (Modulus, Exponent)>
<!ELEMENT Modulus (#PCDATA)>
<!ELEMENT Exponent (#PCD)>
```

Notification DTD

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT Message ((NotificationRequest, Signature) | (NotificationResponse |
Error))>
<!ATTLIST Message
    Id ID #IMPLIED
>
<!ELEMENT NotificationRequest (StartDate?, EndDate?)>
<!ATTLIST NotificationRequest
    Id ID #REQUIRED
    IssuerId NMTOKEN #IMPLIED
    GroupId NMTOKEN #IMPLIED
    Type (cardreg | cardlock | cardoptout | carddeviceupdate) #REQUIRED
    EncVectorIV CDATA #IMPLIED
>
<!ELEMENT StartDate (#PCDATA)>
<!ELEMENT EndDate (#PCDATA)>
```

```
<!ELEMENT NotificationResponse ((CardLock*) | (CardOptOut*) | (CardReg*)
| (CardDeviceUpdate))>
<!ATTLIST NotificationResponse
    Id ID #REQUIRED
    IssuerId NMTOKEN #IMPLIED
    GroupId NMTOKEN #IMPLIED
    Type (cardreg | cardlock | cardoptout | carddeviceupdate) #REQUIRED
    EncVectorIV CDATA #IMPLIED
>
<!ELEMENT Error (Code, ErrorMessage, ErrorDetail)>
<!ELEMENT Code (#PCDATA)>
<!ELEMENT ErrorMessage (#PCDATA)>
<!ELEMENT ErrorDetail (#PCDATA)>
<!ELEMENT CardReg (Card+)>
<!ELEMENT Card ((Data*, Lock*, Unlock*) | (Data*, Unlock*, Lock*) |
(Data*, Device*, Register?, Confirm?) | (Data*, Device*, Confirm?,
Register?) | (Data*, Optout+) | (Data*, DeviceUpdate+))>
<!ELEMENT CardDeviceUpdate (Card+)>
<!ELEMENT DeviceUpdate EMPTY>
<!ATTLIST DeviceUpdate
    SerialId CDATA #REQUIRED
    DeviceType (VASCO | SMS | EMAIL | OOB | DECOUPLED)
    #REQUIRED
    Action (RegisterNewDevice | RegisterExistingDevice | ActivateByNewDevice |
    ActivateByExistingDevice | UnregisterDevice | RemoveDevice ) #REQUIRED
    Date NMTOKEN #REQUIRED
>
<!ATTLIST Card
    Type (VbV | SPA | JCB | SK | DC) #REQUIRED
    Number NMTOKEN #REQUIRED
    Name CDATA #REQUIRED
>
<!ELEMENT Optout (#PCDATA)>
<!ATTLIST Optout
    Number NMTOKEN #REQUIRED
>
<!ELEMENT Lock (#PCDATA)>
<!ELEMENT Unlock (#PCDATA)>
<!ELEMENT Register (#PCDATA)>
<!ELEMENT Confirm (#PCDATA)>
<!ELEMENT CardLock (Card+)>
<!ELEMENT CardOptOut (Card+)>
<!ELEMENT Data EMPTY>
<!ATTLIST Data
    Name NMTOKEN #REQUIRED
    Value CDATA #REQUIRED
>
<!ELEMENT Device (#PCDATA)>
<!ATTLIST Device
    DeviceType NMTOKEN #REQUIRED
    SerialId CDATA #REQUIRED
```

```
>
<!-- DTD for XML Signatures http://www.w3.org/2000/09/xmldsig# -->
<!ENTITY % Object.ANY ''>
<!ENTITY % Method.ANY ''>
<!ENTITY % Transform.ANY ''>
<!ENTITY % SignatureProperty.ANY ''>
<!ENTITY % KeyInfo.ANY ''>
<!ENTITY % KeyValue.ANY ''>
<!ENTITY % PGPData.ANY ''>
<!ENTITY % X509Data.ANY ''>
<!ENTITY % SPKIData.ANY ''>
<!-- Start Core Signature declarations, these should NOT be altered -->
<!ELEMENT Signature (SignedInfo, SignatureValue, KeyInfo?, Object*)>
<!ATTLIST Signature
    xmlns CDATA #FIXED "http://www.w3.org/2000/09/xmldsig#"
    Id ID #IMPLIED
>
<!ELEMENT SignatureValue (#PCDATA)>
<!ATTLIST SignatureValue
    Id ID #IMPLIED
>
<!ELEMENT SignedInfo (CanonicalizationMethod, SignatureMethod, Reference+)>
<!ATTLIST SignedInfo
    Id ID #IMPLIED
>
<!ELEMENT CanonicalizationMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST CanonicalizationMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT SignatureMethod (#PCDATA|HMACOutputLength %Method.ANY;)* >
<!ATTLIST SignatureMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT Reference (Transforms?, DigestMethod, DigestValue)>
<!ATTLIST Reference
    Id ID #IMPLIED
    URI CDATA #IMPLIED
    Type CDATA #IMPLIED
>
<!ELEMENT Transforms (Transform+)>
<!ELEMENT Transform (#PCDATA|XPath %Transform.ANY;)* >
<!ATTLIST Transform
    Algorithm CDATA #REQUIRED
>
<!ELEMENT XPath (#PCDATA)>
<!ELEMENT DigestMethod (#PCDATA %Method.ANY;)* >
<!ATTLIST DigestMethod
    Algorithm CDATA #REQUIRED
>
<!ELEMENT DigestValue (#PCDATA)>
<!ELEMENT KeyInfo (#PCDATA|KeyName|KeyValue|RetrievalMethod|
```

```
X509Data|PGPData|SPKIData|MgmtData %KeyInfo.ANY;)* >
<!ATTLIST KeyInfo
    Id ID #IMPLIED
>
<!-- Key Information -->
<!ELEMENT KeyName (#PCDATA)>
<!ELEMENT KeyValue (#PCDATA|DSAKeyValue|RSAKeyValue %KeyValue.ANY;)* >
<!ELEMENT MgmtData (#PCDATA)>
<!ELEMENT RetrievalMethod (Transforms?)>
<!ATTLIST RetrievalMethod
    URI CDATA #REQUIRED
    Type CDATA #IMPLIED
>
<!-- X.509 Data -->
<!ELEMENT X509Data ((X509IssuerSerial | X509SKI | X509SubjectName |
X509Certificate | X509CRL )+ %X509Data.ANY;)>
<!ELEMENT X509IssuerSerial (X509IssuerName, X509SerialNumber)>
<!ELEMENT X509IssuerName (#PCDATA)>
<!ELEMENT X509SubjectName (#PCDATA)>
<!ELEMENT X509SerialNumber (#PCDATA)>
<!ELEMENT X509SKI (#PCDATA)>
<!ELEMENT X509Certificate (#PCDATA)>
<!ELEMENT X509CRL (#PCDATA)>
<!-- PGPData -->
<!ELEMENT PGPData ((PGPKeyID, PGPKeyPacket?) | (PGPKeyPacket) %PGPData.ANY;)
>
<!ELEMENT PGPKeyPacket (#PCDATA)>
<!ELEMENT PGPKeyID (#PCDATA)>
<!-- SPKI Data -->
<!ELEMENT SPKIData (SPKISexp %SPKIData.ANY;) >
<!ELEMENT SPKISexp (#PCDATA)>
<!-- Extensible Content -->
<!ELEMENT Object (#PCDATA|Signature|SignatureProperties|Manifest
%Object.ANY;)* >
<!ATTLIST Object
    Id ID #IMPLIED
    MimeType CDATA #IMPLIED
    Encoding CDATA #IMPLIED
>
<!ELEMENT Manifest (Reference+)>
<!ATTLIST Manifest
    Id ID #IMPLIED
>
<!ELEMENT SignatureProperties (SignatureProperty+)>
<!ATTLIST SignatureProperties
    Id ID #IMPLIED
>
<!ELEMENT SignatureProperty (#PCDATA %SignatureProperty.ANY;)* >
<!ATTLIST SignatureProperty
    Target CDATA #REQUIRED
    Id ID #IMPLIED
```

```
>
<!-- Algorithm Parameters -->
<!ELEMENT HMACOutputLength (#PCDATA)>
<!ELEMENT DSAKeyValue ((P, Q)?, G?, Y, J?, (Seed, PgenCounter)?)>
<!ELEMENT P (#PCDATA)>
<!ELEMENT Q (#PCDATA)>
<!ELEMENT G (#PCDATA)>
<!ELEMENT Y (#PCDATA)>
<!ELEMENT J (#PCDATA)>
<!ELEMENT Seed (#PCDATA)>
<!ELEMENT PgenCounter (#PCDATA)>
<!ELEMENT RSAKeyValue (Modulus, Exponent)>
<!ELEMENT Modulus (#PCDATA)>
<!ELEMENT Exponent (#PCDATA)>
```