



Secure Document Exchange  
Web Services  
6<sup>th</sup> October 2010

To get the very latest information including sample code please visit  
<http://support.secure-dx.com>

## Table of Contents

DCXService .....	7
isRegisteredUser .....	7
Description .....	7
Request Parameters .....	7
Response .....	7
isRegisteredUserEx .....	8
Description .....	8
Response .....	9
areRegisteredUsers .....	13
Description .....	13
Request Parameters .....	13
Response .....	13
areRegisteredUsersEx .....	14
Description .....	14
Additional Request Parameters .....	14
Response .....	14
sendPackage .....	15
Description .....	15
Request Parameters .....	15
Response .....	17
sendPackageEx .....	18
Description .....	18
see <a href="#">sendPackage</a> for all parameters .....	18
Additional Request Parameters .....	18
sendPackageEx2 .....	19
Description .....	19
see <a href="#">sendPackage</a> and <a href="#">sendPackageEx</a> for all parameters .....	19
Additional Request Parameters .....	19
sendPackageEx3 .....	20
Description .....	20
See <a href="#">sendPackage</a> , <a href="#">sendPackageEx</a> and <a href="#">sendPackageEx2</a> for all parameters .....	20
Additional Request Parameters .....	20
sendReply .....	21
Description .....	21
Request Parameters .....	21
Response .....	22
getPackageStatus .....	24
Description .....	24
Request Parameters .....	24
Response .....	24
getExtendedPackageStatus .....	26
Description .....	26
Request Parameters .....	26
Response .....	26
getExtendedPackageStatusEx .....	28
Description .....	28

Additional Request Parameters.....	28
Response .....	28
recallPackage .....	30
Description.....	30
Request Parameters.....	30
Response .....	30
getPackageTrackingIds (UPDATED 6 <sup>th</sup> OCT 2010).....	31
Description.....	31
Request Parameters.....	31
Response .....	31
getPackageTrackingIdsSince (UPDATED 6 <sup>th</sup> OCT 2010).....	33
Description.....	33
Request Parameters.....	33
Response .....	33
getPackageStatusTypes (NEW 6 <sup>th</sup> OCT 2010).....	35
Description.....	35
Request Parameters.....	36
Response .....	37
DCXRecipientService.....	38
getFolderContent.....	38
Description.....	38
Request Parameters.....	38
Response .....	38
getFolderContentEx .....	40
Description.....	40
Request Parameters.....	40
Response .....	41
getPackageContent.....	43
Description.....	43
Request Parameters.....	43
Response .....	43
getPackageAttachment.....	45
Description.....	45
Request Parameters.....	45
Response .....	45
setPackageNote.....	47
Description.....	47
Request Parameters.....	47
Response .....	47
setPackagesNotes.....	48
Description.....	48
Request Parameters.....	48
Response .....	48
setPackageLabel.....	50
Description.....	50
Request Parameters.....	50
Response .....	50
setPackagesLabels.....	51

Description .....	51
Request Parameters .....	51
Response .....	51
setPackageError .....	54
Description .....	54
Request Parameters .....	54
Response .....	54
setPackagePapered .....	56
Description .....	56
Request Parameters .....	56
Response .....	56
setPackagesPapered .....	57
Description .....	57
Request Parameters .....	57
Response .....	57
movePackage .....	58
Description .....	58
Request Parameters .....	58
Response .....	58
movePackages .....	59
Description .....	59
Request Parameters .....	59
Response .....	60

### **Secure Document Exchange Securing Business Communications**

The Secure Document Exchange (SDX) system helps financial institutions increase efficiency and decrease costs while gaining control over content sent via the Internet. A fully collaborative tool, SDX allows information to move between senders and recipients in a secure, easy-to-use environment. Financial institutions can access the benefits of Secure Document Exchange in three integration options, Web-UI, SDX-Onsite, and SDX Web Services.

SDX is a complete solution that provides fast, secure transmission using industry and regulatory-accepted technology. Designed with consideration for legislative requirements, such as Gramm-Leach-Bliley and Sarbanes-Oxley (SOX), SDX provides security to help meet needs that many Internet delivery tools don't address.

### **Savings: SDX Delivers Increased Efficiency and Decreased Costs**

With tangible savings of 65 percent off the cost of traditional overnight shipping, SDX provides clients with a wealth of cost- and time- cutting benefits, such as:

- **Secure Messaging** – Save development time by using a ready-built system to secure all Internet communications.
- **Expedited Delivery** – Securely deliver any type of file or message in seconds.
- **Workflow Improvement** – Automating document delivery drastically increases all types of transactions in the financial services market.
- **Fully Supported** – Wolters Kluwer Financial Services provides extensive, world-class support services for SDX on both the sender and recipient side, including free internal helpdesks for supporting secure document exchange. In a recent survey 100 percent of our customers responded that they would recommend our products based on customer support.

### **Flexibility: Adaptable Control means Reliable Content**

The flexible nature of the SDX solution allows it to be integrated seamlessly with most all systems and networks while providing the control necessary to realize increased efficiencies.

- **Fully Collaborative Exchange** – SDX can be easily configured to allow communication via one-way or two-way channels, which allows documents and data to be exchanged between sender and recipient during the process, for example sending the HUD-1 back and forth for reconciliation.
- **Integration Options** – Unlike other solutions, SDX doesn't require the installation of any hardware or software for users to operate. However, SDX can be integrated into internal systems in a number of ways for maximum flexibility and user preference. The solution can be accessed via a web-based UI, or integrated seamlessly into an existing system via a server application or true web-services.
- **Web Services** - SDX supports web services and allows for third party or custom programming against the SDX service through platforms such as .NET and Java.
- **Optional Electronic Signing** - SDX can be configured to ensure users need to sign legal terms and conditions before accessing a package.
- **Seamlessly Coexist with Corporate Archive and E-Mail Compliance Systems** - SDX can be configured to coexist with any existing archive and e-mail system and conform to existing security policies or compliance content inspection. The technology behind the system's security features was created with Sarbanes-Oxley, SEC 17 a-4, NASD 3010, and HIPAA regulations in mind.

**Security: The Backbone of the System**

SDX employs the highest level of protective measures to keep content safe and secure during transmission to intended recipients. These measures include the following:

- **Tamper Sealing** – All messages and attachments are encrypted and tamper sealed inside a Digital Vault.
- **Digital Rights Management** - Supports ‘Digital Shredding’ of secure messages and communications, print, and copy protection.
- **Enforceable Integrity** - Content is the lifeblood of any organization and its accuracy is paramount. SDX provides features to ensure the integrity of each message and the verification of content.

SDX is the optimal secure Internet document exchange solution for institutions that offer mortgage products because it supports the electronic mortgage (eMortgage) framework and data standards created by the Mortgage Industry Standards Maintenance Organization (MISMO). SDX is built on a true digital vault to provide a powerful and adaptable eMortgage backbone and architecture. SDX offers the ability to add tamper-evident seals to any file type including Real Estate Settlement Procedures Act (RESPA) disclosure documents.

Individual users usually interact with SDX via the Website interface but SDX also offers third party system support via both an email and a Web Services interface. This document describes the Web Services interface.

Web Services allow third party programs to interact with SDX at a programmatic level. Modern development environments such as .NET from Microsoft and Java from Sun offer built-in support for Web Services interfaces. More information on Web Services can be found by following these links <http://www.w3.org/TR/wsdl>, <http://www.w3.org/2002/ws/>, <http://msdn.microsoft.com/webservices>.

## DCXService

### *isRegisteredUser*

#### Description

This method is used to determine whether an intended recipient of a secure mail package is registered with the system prior to sending the package. If a package is intended for multiple recipients, a separate request can be sent for each of the intended recipients, or else the areRegisteredUsers message can be used. Its use is not mandatory, but is recommended.

#### Request Parameters

##### username

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who will send a package

##### password

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who will send a package

##### encodedPassword

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

##### emailID

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The email address of the intended recipient

#### Response

Type: xsd:string

Possible return values:

- “YES” - the intended recipient is registered with the system
- “NO” - the intended recipient is **not** registered with the system
- “ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Empty emailID”)

## ***isRegisteredUserEx***

### **Description**

This method extends `isRegisteredUser` in that allows for automatic creation of users. Unregistered users are created only if the webservice user belongs to a group where “Create Unregistered Users” is enabled. Additionally This method allows for passing an additional if this username known as the `replyTo` user. This user must have “Create Unregistered Users” enabled for his group.

### **username**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who will send a package

### **password**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who will send a package

### **encodedPassword**

Type: `xsd:boolean`

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

### **emailID**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The email address of the intended recipient.

### **replyTo**

Type: `xsd:string`

Requirement: optional – can be null or empty

Description: An optional parameter which, if used in the

`sendPackage` message will denote the user from whom a package is to be sent on behalf of. In This method, the **replyTo** user’s group configuration settings are interrogated to see whether unregistered users should be created. If the **replyTo** user is not a registered user of the system, then the group configuration settings of the user denoted by **username** will be checked to see whether unregistered recipients should be created.

**userInfo**

Type: A UserInfo complex type (see below)

Requirement: optional – may be null or an empty.

Description: Additional user information which is applied to the new user if it created.

Populating this type with data may be used to suppress the user profile update page when a user logs in.

**Response**

Type: xsd:string

Possible return values:

- “YES” - the intended recipient is registered with the system
- “NO” - the intended recipient is **not** registered with the system
- “ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Empty emailID”)

**Complex Type: Attachment****birthDay**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**birthMonth**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**City**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**Company**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**contactLine**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**contactNPA**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**contactNxx**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**Country**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**Department**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**emailAlias1**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**emailAlias2**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**emailAlias3**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**emailAlias4**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**emailAlias5**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**employerName**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**faxPhone**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**firstName**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**forceSetAdditional**

Type: xsd:string  
Requirement: optional – may be null or empty. This value defaults to false  
Description: If this flag is set to false then the User Profile update page which can appear on login is suppressed.

**homePhone**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**lastName**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**middleName**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**operatingSystem**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**otherPhone**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**State**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**streetAddress1**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**streetAddress2**

Type: xsd:string  
Requirement: optional – may be null or empty  
Description:

**Title**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**titlePosition**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**workPhone**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

**zipCode**

Type: xsd:string

Requirement: optional – may be null or empty

Description:

## ***areRegisteredUsers***

### **Description**

This method is used to determine whether one or more intended recipients of a secure mail package are registered with the system prior to sending the package. Its use is not mandatory, but is recommended.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who will send a package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who will send a package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **emailIDs**

Type: array of xsd:string

Requirement: mandatory – cannot be null or empty

Description: An array of email addresses of the intended recipients

### **Response**

Type: xsd:string

Possible return values:

- “YES” - all of the intended recipients are registered with the system
- “NO” - one or more of the intended recipients are **not** registered with the system. This will be followed by a semi-colon separated list of the email IDs of the unregistered users (e.g. “NO : user1@isentry.com;user2@isentry.com”)
- “ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : No emailIDs provided”)

## **areRegisteredUsersEx**

### **Description**

This method is an extension to the areRegisteredUsers message (see the Description). This method works in the same way but implements the following additional functionality: dependent upon configuration of the Secure Document Exchange server, This method can also cause unregistered users to be created. The Request Parameters and Response are identical barring the following addition.

### **Additional Request Parameters**

#### **replyTo**

Type: xsd:string

Requirement: optional – can be null or empty

Description: An optional parameter which, if used in the

sendPackage message will denote the user from whom a package is to be sent on behalf of. In This method, the **replyTo** user's group configuration settings are interrogated to see whether unregistered users should be created. If the **replyTo** user is not a registered user of the system, then the group configuration settings of the user denoted by **username** will be checked to see whether unregistered recipients should be created.

### **Response**

In essence the response can be interpreted in exactly the same way as for areRegisteredUsers i.e. NO means some users are not registered, YES means all users are registered this may mean that one or more users were also created. And ERROR means an error occurred. If you wish to know when users are created then you should call areRegisteredUsers first in order to determine which users need to be created.

Type: xsd:string

Possible return values:

- “YES” - All of the intended recipients are registered with the system. If the SDX server is configured to allow user creation “on the fly” **and** the **replyTo** users group has the right to create unregistered users then any unregistered users were created.
- “NO” - One or more of the intended recipients are **not** registered with the system. This will be followed by a semi-colon separated list of the email IDs of the unregistered users (e.g. “NO : [user1@isentry.com](#);[user2@isentry.com](#)”) If create unregistered is enabled for the **replyTo** user and the email address is not in a valid format (name@domain.topleveldomain).
- “ERROR” - An error occurred. This will be followed by a description of the error (e.g. “ERROR : No emailIDs provided”)

## ***sendPackage***

### **Description**

This method is used to send a secure mail package.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who is sending the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who is sending the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **replyTo**

Type: xsd:string

Requirement: optional – can be null or empty

Description: An optional parameter which denotes the user from whom a package is to be sent on behalf of. If the user exists, then they will become the sender of the package instead of **username**.

#### **recipients**

Type: array of xsd:string

Requirement: mandatory – cannot be null or empty

Description: An array of the usernames of the intended recipients of the package

#### **subject**

Type: xsd:string

Requirement: mandatory – cannot be null **or empty**

Description: The subject of the package

**subjectMetaApplies**

Type: xsd:boolean

Requirement: optional – can be null or empty (indicates false)

Description: Whether the subject is formatted according to meta-data configuration

**body**

Type: xsd:string

Requirement: optional – can be null or empty

Description: The body of the package

**attachments**

Type: Array of the Attachments complex type (see below)

Requirement: optional – may be null or an empty array if no attachments are to be sent

Description: The array of attachments

**settings**

Type: An instance of the Settings complex type (see below)

Requirement: optional – may be null if the sender's default settings are to be used

Description: The settings for the package

**Complex Type: Attachment****filename**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The attachment filename (e.g. mortgage1.doc)

**contentType**

Type: xsd:string

Requirement: optional – can be null or empty

Description: The MIME content type of the attachment (e.g. application/msword)

**content**

Type: array of xsd:base64Binary

Requirement: mandatory – cannot be null **or empty**

Description: This is the content of the attachment

**compresses**

Type: xsd:boolean

Requirement: optional – can be null or empty (indicates false)

Description: Indicates whether the content is gzip compressed

### **Complex Type: Settings**

#### **expirySetting**

Type: xsd:boolean

Requirement: optional – can be null or empty (indicates false)

Description: This specifies whether or not the secure package should expire (i.e. become unavailable to all recipients) after a certain date

#### **expiry**

Type: xsd:dateTime

Requirement: optional – can be null or empty (default setting will apply)

Description: If expirySetting is true, this specified the date after which the package will expire

#### **viewingLimitSetting**

Type: xsd:boolean

Requirement: optional – can be null or empty (indicates false)

Description: This specifies whether or not the secure package should have a viewing limit (i.e. once the viewing limit is reached, the package will become unavailable to that recipient)

#### **viewingLimit**

Type: xsd:short

Requirement: optional – can be null or empty (default setting will apply)

Description: : If viewingLimitSetting is true, this specified the number of times a recipient can open a package after which it will become unavailable for viewing

### **Response**

Type: xsd:string

A successful response will at a minimum, return

<SENT\_ID>11056</SENT\_ID>

Other fields may also be present depending on the skin configuration.

An error will return

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : No recipients provided”)

***sendPackageEx*****Description**

This method is an extension to the

sendPackage message (see the Description). This method works in the same way but takes a further parameter indicating that recipients of the package must provide additional authentication before they are allowed access to the package. So the Request Parameters and Response are identical barring the addition of the additionalAuthReqd parameter below.

see [sendPackage](#) for all parameters.

***Additional Request Parameters*****additionalAuthReqd**

Type: xsd:boolean

Requirement: mandatory – cannot be null or empty

Description: This is used to determine whether the recipient must provide additional authentication before being permitted access to the package.

## ***sendPackageEx2***

### **Description**

This method is an extension to the sendPackageEx message (see the sendPackageEx). This method works in the same way but takes a further parameter indicating whether the recipient of the package will be requested to sign (or reject) the package. This allows you to specify that the package requires none or a combination of:

- Additional Authentication
- Signing.

So the Request Parameters are identical barring the addition of the requestSign parameter below

see [sendPackage](#) and [sendPackageEx](#) for all parameters.

### **Additional Request Parameters**

#### **requestSign**

Type: xsd:boolean

Requirement: mandatory – cannot be null or empty

Description: This is used to determine whether the recipient of the package will be requested to sign (or reject) the package.

## ***sendPackageEx3***

### **Description**

This method is an extension to the sendPackageEx2 message (see the Description). This method works in the same way but takes a further parameter which is used to indicate the **start time** of the package. Typically this is used in a “3day docs” scenario where the application date may precede the time the package was sent. There is only one addition to the parameters specified by sendPackageEx2; applicationTimestamp described below.

See [sendPackage](#), [sendPackageEx](#) and [sendPackageEx2](#) for all parameters.

### **Additional Request Parameters**

#### **applicationTimestamp**

Type: xsd:boolean

Requirement: mandatory – cannot be null or empty

Description: This is used to determine whether the recipient of the package will be requested to sign (or reject) the package.

## ***sendReply***

### **Description**

This method allows a calling process to send a response back to a message back into SDX via web services. The caller has a more limited number of options than when calling one of the sendPackage calls but the benefit is that the response is tied into the original package and includes all the original subject data or meta data as appropriate.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who is sending the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who is sending the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **subject**

Type: xsd:string

Requirement: optional – can be null or empty

Description: The subject of the package.

Note that if the original package was not using meta data then this parameter will be ignored. If the original package was using meta data then anything provided in this field will be appended to the existing meta data. Changing existing meta data is NOT supported.

#### **body**

Type: xsd:string

Requirement: optional – can be null or empty

Description: The body of the package

#### **attachments**

Type: Array of the Attachments complex type (see below)

Requirement: optional – may be null or an empty array if no attachments are to be sent

Description: The array of attachments

## **id**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: This is the tracking id of the original message that is being replied too.

Note that attempts to reply to messages that were not originally sent to the account defined in the <username> parameter will be forcefully rejected.

## **Complex Type: Attachment**

### **filename**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The attachment filename (e.g. mortgage1.doc)

### **contentType**

Type: xsd:string

Requirement: optional – can be null or empty

Description: The MIME content type of the attachment (e.g. application/msword)

### **content**

Type: array of xsd:base64Binary

Requirement: mandatory – cannot be null **or empty**

Description: This is the content of the attachment

### **compresses**

Type: xsd:boolean

Requirement: optional – can be null or empty (indicates false)

Description: Indicates whether the content is gzip compressed

## **Response**

Type: xsd:string

A successful response will at a minimum, return

<SENT\_ID>11056</SENT\_ID>

Other fields may also be present depending on the skin configuration.

An error will return

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : No recipients provided”)

## **getPackageStatus**

### **Description**

This method is used to retrieve the status of a sent package.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID for the sent package (see sendPackage response)

### **Response**

Type: xsd:string

Possible return values:

An xml string – will take the form of

```
<package_statuses>
  <package_status>
    <recipient></recipient>
    <status></status>
  </package_status>
</package_statuses>
```

with a <package\_status> entry for each of the recipients of the package.

Possible contents for the <status> element are

Recalled on *date*

Purged on *date*

Viewed on *date*

Received on *date*

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Package not found”)

## ***getExtendedPackageStatus***

### **Description**

This method is used to retrieve an extended status message for a sent package. This call is usually used where there is a need to provide a status file for a calling system. The resulting XML can be streamed to a file.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID for the sent package (see sendPackage response)

### **Response**

Type: xsd:string

Possible return values:

If successful the response will be an XML format string detailing the entire package and its attachments. The exact format is dependant on how the subject fields are configured on the SDX server. These are set in the **SubjectMetaData** server config item.

For example if the server is configured to expect the fields BorrowerName1, LoanNumber and PropertyAddress the XML will look something like the example below. You will see that below the <Subject> entry there is an entry for each subject field that the SDX server expected ie. <BorrowerName1> <LoanNumber> and <PropertyAddress>

```

<?xml version="1.0" encoding="UTF-8" ?>
<Package>
  <PackageDetail>
    <Id>103003</Id>
    <Sender>loanofficer@lender.com</Sender>
    <SubmittedBy>onsite@isentry.com</SubmittedBy>
    <ReplyTo></ReplyTo>
    <Subject>LOAN NUMBER=12345678:RECIPIENT NAME=John
Smith:BRANCH NUMBER=12:PROPERTY ADDRESS=2 Somewhere, sometown,
somestate:BORROWER NAME1=Alan Borrower</Subject>
    <BorrowerName1>Alan Borrower</BorrowerName1>
    <LoanNumber>12345678</LoanNumber>
    <PropertyAddress>2 Somewhere, sometown,
somestate</PropertyAddress>
    <Comment></Comment>
    <NumberOfAttachments>1</NumberOfAttachments>
    <Attachments>
      <Attachment>
        <Name>loandocs.pcl</Name>
      </Attachment>
    </Attachments>
  </PackageDetail>
  <PackageStatuses>
    <PackageStatus>
      <Recipient>closer@somewhere.com</Recipient>
      <Received>2006-04-06 05:08:50</Received>
      <Viewed></Viewed>
      <Purged></Purged>
      <Recalled></Recalled>
    </PackageStatus>
  </PackageStatuses>
</Package>

```

There will be an <Attachment> entry for each attachment and a <PackageStatus> for each recipient.

The <SubmittedBy> entry will contain the username used when sendPackage was called. If the ReplyTo address supplied in the sendPackage call referred to a registered SDX user then the Sender will be set to that ReplyTo address. If the ReplyTo was not a registered user then the <ReplyTo> entry will contain that address.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Package not found”)

## ***getExtendedPackageStatusEx***

### **Description**

This method is an extension to the `getExtendedPackageStatus` message (see the `getExtendedPackageStatus`). This method works in the same way but takes a further parameter indicating whether the viewed status of attachments (if any) should be included in the response. It also returns `<signed>` and `<rejected>` elements as part of a `<PackageStatus>` section. So the Request Parameters are identical to those of `getExtendedPackageStatus` with the following addition.

### **Additional Request Parameters**

#### **includeAttachments**

Type: `xsd:boolean`

Requirement: mandatory – cannot be null or empty

Description: An optional parameter which is used to denote whether the viewed status of attachments (if any) should be included in the response.

### **Response**

Type: `xsd:string`

Possible return values:

The response is similar to that of `getExtendedPackageStatus` but with additional `<signed>` and `<rejected>` elements and an additional `<AttachmentStatuses>` section as exemplified by the following shaded area:

```
<Package>
  <PackageDetail>
    <Id>226004</Id>
    <Sender>from@isentry.com</Sender>
    <SubmittedBy>mywsclient@isentry.com</SubmittedBy>
    <ReplyTo></ReplyTo>
    <Subject>Loan Number=123456:Comment=Wibble</Subject>
    <LoanNumber>123456</LoanNumber>
    <Comment>Wibble</Comment>
    <NumberOfAttachments>1</NumberOfAttachments>
    <Attachments>
      <Attachment>
        <Name>adresscover.pcl</Name>
      </Attachment>
    </Attachments>
  </PackageDetail>
  <PackageStatuses>
    <PackageStatus>
      <Recipient>to@isentry.com</Recipient>
      <Received>2006-11-07 15:04:00</Received>
      <Viewed>2006-11-07 15:05:31</Viewed>
      <Purged></Purged>
      <Recalled></Recalled>
      <Signed>2006-11-07 15:06:40</Signed>
      <Rejected></Rejected>
      <PaperRequested></PaperRequested>
      <PaperCompleted></PaperCompleted>
      <PaperNote></PaperNote>
      <AttachmentStatuses>
        <AttachmentStatus>
          <Name>adresscover.pcl</Name>
          <Viewed>2006-11-07 15:06:29</Viewed>
        </AttachmentStatus>
      </AttachmentStatuses>
    </PackageStatus>
  </PackageStatuses>
</Package>
```

## ***recallPackage***

### **Description**

This method is used to recall a previously sent package.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID for the sent package (see sendPackage response)

### **Response**

Type: xsd:string

Possible return values:

“YES” - the package was recalled

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Package not found”)

**getPackageTrackingIds (UPDATED 6<sup>th</sup> OCT 2010)****Description**

This method is used to retrieve a list of package tracking ids for a specific sender. It is typically used by a calling system that wants to refresh its own view of all sent messages.

**Request Parameters****username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

**password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

**encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

**trackingFilter**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking filters to be applied. Please see getPackageStatusTypes for information on values that can be sent

**Response**

Possible return values:

Type: xsd:string

```
<trackingids>
  <packageid>
    <id></id>
    <status></status>
    <date></date>
  </packageid>
</trackingids>
```

with a <packageid> entry for each of the package that meets the filtering requirement.

There are many possible contents for the <status> elements depending on the type of packages being requested. More details can be found in the section regarding `getPackageStatusTypes`.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Package not found”)

**getPackageTrackingIdsSince (UPDATED 6<sup>th</sup> OCT 2010)****Description**

This method is used to retrieve a list of package tracking ids for a specific sender for which the tracking filter has changed since the given date. For example in order to find all messages that recipients have signed since yesterday. You would set the tracking filter to SIGNED and the date to yesterdays date.

It is typically used by a calling system that wants to poll the SDX and only wants changes from the last time it polled.

**Request Parameters****username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

**password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

**encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

**trackingFilter**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking filters to be applied. Please see getPackageStatusTypes for information on values that can be sent

**Since**

Type: xsd:dateTime

Requirement: mandatory

Description: This is the date from which the IDs will be retrieved.

**Response**

Possible return values:

Type: xsd:string

```
<trackingids>
  <packageid>
    <id></id>
    <status></status>
    <date></date>

  </packageid>
</trackingids>
```

with a <packageid> entry for each of the package that meets the filtering requirement.

There are many possible contents for the <status> elements depending on the type of packages being requested. More details can be found in the section regarding `getPackageStatusTypes`.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Invalid Filter”)

## **getPackageStatusTypes (NEW 6<sup>th</sup> OCT 2010)**

### **Description**

This method is used to retrieve the various filter types that can be used in getPackageTrackingsIds and getPackageTrackingIds since.

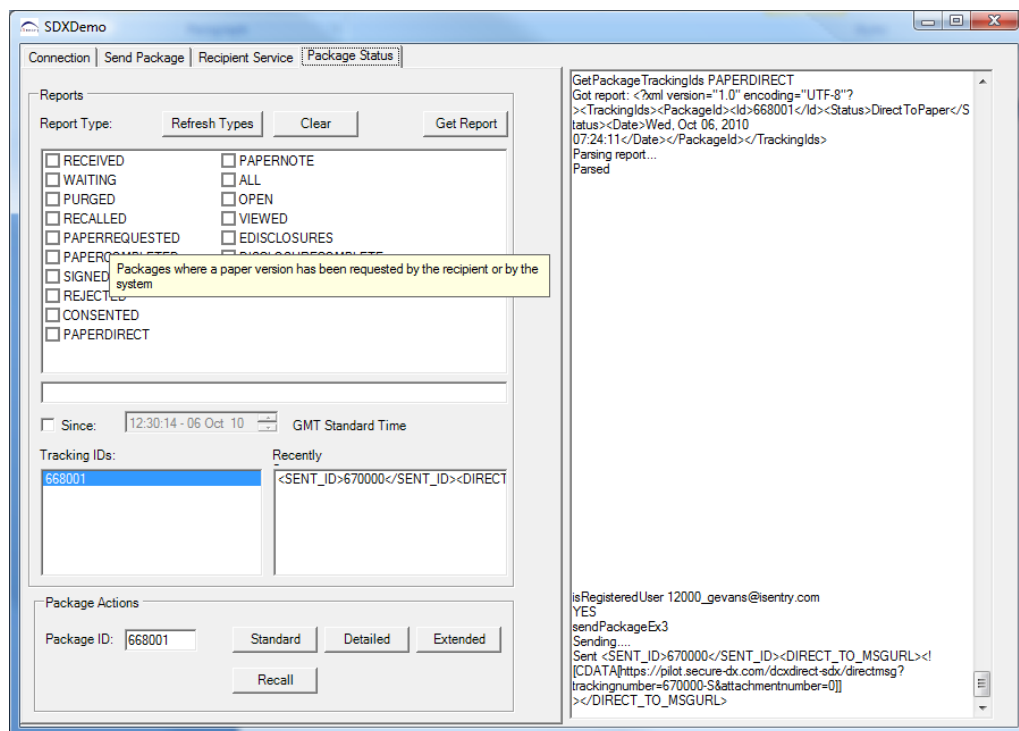
The filter field is a “:” delimited string which defines the types of status messages to be included. For example, to include all signed and rejected packages you could use

“SIGNED:REJECTED”

as the filter.

As the product evolves so do the number of possible status events. This call was introduced as an aid to system developers in that it returns the list of possible filters and includes a descriptive text explain the filter function.

The SDXDemo application has been modified to use this new call as shown below



The current list of filters is included below but this list is subject to change over time. Filters are NOT removed though.

RECEIVED	Packages created but not viewed by recipient
WAITING	Packages created but not viewed by recipient
PURGED	Packages that has been purged
RECALLED	Packages that has been recalled
PAPERREQUESTED	Packages where a paper version has been requested by the recipient or by the system
PAPERCOMPLETED	Packages completed by paper out process
SIGNED	Packages signed by recipient
REJECTED	Packages rejected by recipient
CONSENTED	Packages consented by recipient
PAPERDIRECT	Packages sent direct to paper
PAPERNOTE	Packages with a paper note date set
ALL	Packages received including those purged and recalled
OPEN	Packages received excluding those purged and recalled
VIEWED	Packages viewed by recipient
EDISCLOSURES	Packages signed, rejected, consented, paper requested, sent direct to paper, or paper completed
DISCLOSURECOMPLETE	Packages consented or paper completed by recipient

## Request Parameters

### username

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

### password

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

### encodedPassword

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

## Response

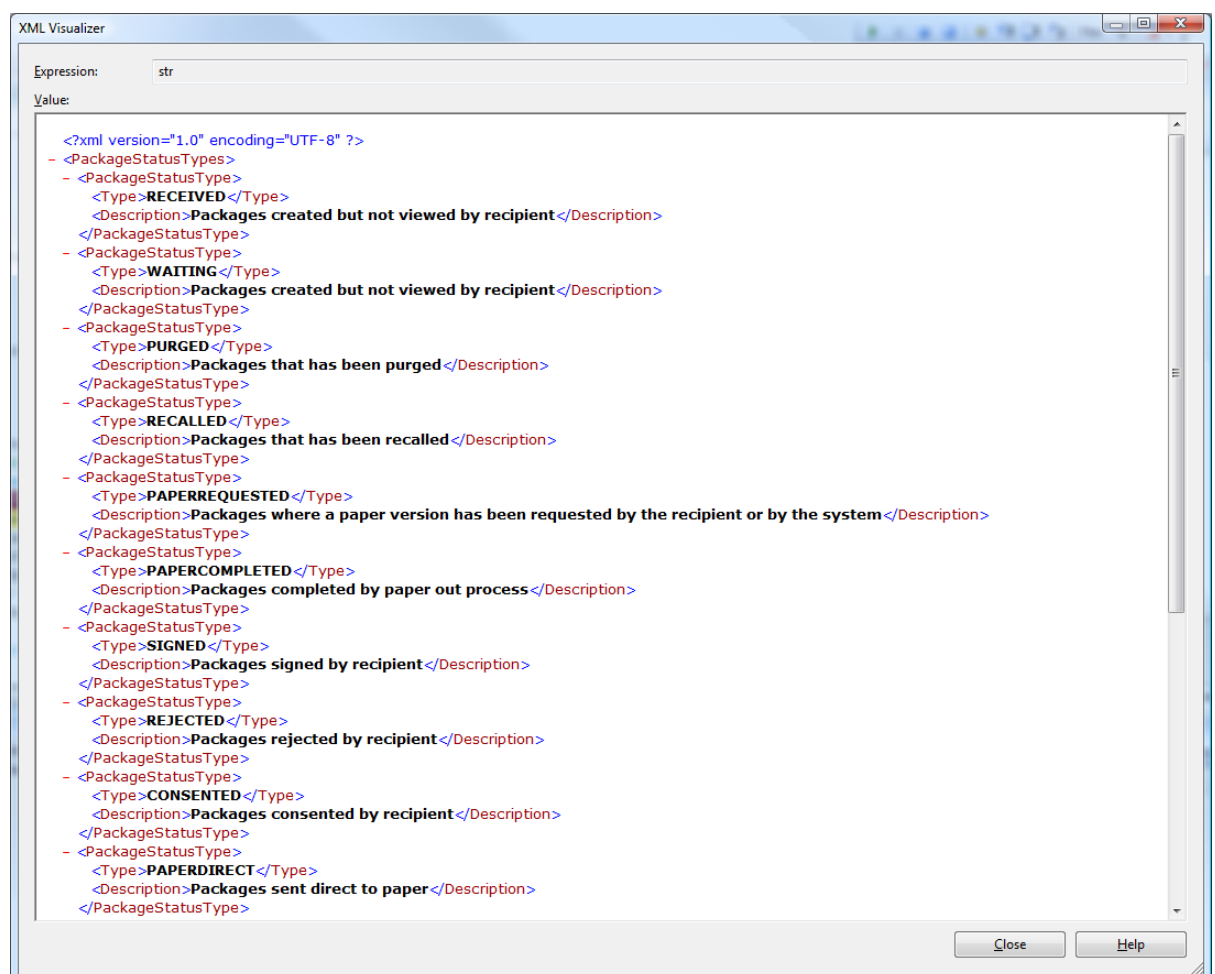
Type: xsd:string

Possible return values:

An xml string – will take the form of

```
<PackageStatusTypes>
  <PackageStatusType>
    <Type>FILTERVALUE</Type>
    <Description>Filter description</Description>
  </PackageStatusType>
</PackageStatusTypes>
```

An example is shown below.



“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Bad username”)

## DCXRecipientService

### *getFolderContent*

#### Description

This method is used to retrieve the contents of a folder.

#### Request Parameters

##### username

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

##### password

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

##### encodedPassword

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

##### folderType

Type: xsd:int

Requirement: mandatory – cannot be null or empty

Description: An integer representing the required folder type. The possible values are:

- 0 – inbox
- 1 – sent items
- 2 – archived items
- 3 – deleted items

#### Response

Type: xsd:string

Possible return values:

An xml string – will take the form of

```
<Folder>
  <Package>
    < PackageDetail>
      <Id></Id>
      <Sender></Sender>
```

```
<Recipient></Recipient>
<Subject></Subject>
<metadatafield></metadatafield>
...
...
<metadatafield></metadatafield>

<NumberOfAttachments></NumberOfAttachments>
<TrackingNumber></TrackingNumber>

    <PackageDetail>
  </Package>
</Folder>
```

The contents of the <Id> element will take the form *digits*-R if the package is a received package or *digits*-S if the package is a sent package.

The metadatafield items will be all those defined within your skin but with whitespace removed, for example

```
<accountname></accountname>
<transactionid></transactionid>
<paperout></paperout>
```

The <subject></subject> entry may not always exist as this can be suppressed by skin settings

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Invalid folderType”)

## ***getFolderContentEx***

### **Description**

This method is used to retrieve a set number of items from a folder. The method is an extension of `getFolderContent` and was implemented to help large volume users who wish to break up their response sizes into more manageable chunks.

### **Request Parameters**

#### **username**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: `xsd:boolean`

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **folderType**

Type: `xsd:int`

Requirement: mandatory – cannot be null or empty

Description: An integer representing the required folder type. The possible values are:

- 0 – inbox
- 1 – sent items
- 2 – archived items
- 3 – deleted items

#### **startPosition**

Type: `xsd:int`

Requirement: mandatory – cannot be null or empty

Description: An integer representing the starting position within the folder

#### **Count**

Type: `xsd:int`

Requirement: mandatory – cannot be null or empty

Description: An integer representing the maximum number of items to return

**bOldestFirst**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the starting position means oldest item.

False will return newest

True will return oldest

**Response**

Type: xsd:string

Possible return values:

An xml string – will take the form of

```

<Folder>
  <Package>
    <PackageDetail>
      <Id></Id>
      <Sender></Sender>
      <Recipient></Recipient>
      <Subject></Subject>
      <metadatafield></metadatafield>
      ...
      ...
      <metadatafield></metadatafield>

      <NumberOfAttachments></NumberOfAttachments>
      <Attachments>
        <Attachment>
          <Name></Name>
          <Number></Number>
        </Attachment>
      </Attachments>

      <TrackingNumber></TrackingNumber>

    </PackageDetail>
  </Package>
</Folder>

```

The contents of the <Id> element will take the form *digits*-R if the package is a received package or *digits*-S if the package is a sent package.

The metadatafield items will be all those defined within your skin but with whitespace removed, for example

<accountname></accountname>

<transactionid></transactionid>

<paperout></paperout>

The <subject></subject> entry may not always exist as this can be suppressed by skin settings

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Invalid folderType”)

## ***getPackageContent***

### **Description**

This method is used to retrieve the contents of a package.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID of the package to retrieve which must be in the same form as that returned by the *<Id>* element in the *Response* to *getFolderContent*.

### **Response**

Type: xsd:string

Possible return values:

An xml string – will take the form of

```
<PackageContent>
  <Sender></Sender>
  <Recipients></Recipients>
  <Subject></Subject>
  <PaperRequested></PaperRequested>
  <PaperCompleted></PaperCompleted>
  <Note> </Note>
  <Body></Body>
  <NumberOfAttachments></NumberOfAttachments>
  <Attachments>
    <Attachment>
```

```
<Name></Name>  
<Number></Number>  
</Attachment>  
</Attachments>  
</PackageContent>
```

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Wrong format for trackingNumber”)

## ***getPackageAttachment***

### **Description**

This method is used to retrieve a package attachment.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID of the package to retrieve which must be in the same form as that returned by the *<Id>* element in the *Response* to *getFolderContent*.

#### **attachmentNumber**

Type: xsd:int

Requirement: mandatory – cannot be null or empty

Description: The number of the attachment to retrieve as returned in the *<Name>* element in the *Response* to *getPackageContent*.

#### **compressAttachment**

Type: xsd:boolean

Requirement: mandatory – cannot be null or empty

Description: Indicates whether the returned attachment content will be compressed.

### **Response**

Type: AttachmentEx complex type (see below).

**Complex Type: AttachmentEx (extends complex type Attachment with additional field)****error**

Type: xsd:string

Requirement: optional – can be null or empty

Description: This field will be set if an error occurs, otherwise the remaining fields will be set.

## **setPackageNote**

### **Description**

This method is used to set the note on a received message, i.e. it is only valid for received messages. If the message is a paper-out request then note will be accessible to the original sender of the message via the `getExtendedPackageStatusEx` method.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID of the package to retrieve which must be in the same form as that returned by the `<Id>` element in the *Response* to *getFolderContent*.

#### **sNote**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The note to set on this message.

### **Response**

Type: xsd:string

Possible return values:

“OK” - The note was set.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Wrong format for trackingNumber”)

## **setPackagesNotes**

### **Description**

This method is used to set the notes on received messages, i.e. it is only valid for received messages. If the message is a paper-out request then note will be accessible to the original sender of the message via the `getExtendedPackageStatusEx` method.

### **Request Parameters**

#### **username**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: `xsd:boolean`

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **sNotes**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: A “;” delimited list of tracking ids and note in the format

“11042-R=Paper note;11045-R=Paper note 2;11046-R=Paper note 3”

The tracking ID of the package to retrieve which must be in the same form as that returned by the `<Id>` element in the *Response* to `getFolderContent`.

The note element is the note to set on each tracking id

### **Response**

Type: `xsd:string`

Possible return values:

A “;” delimited list of status returns in the format

“OK;11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=OK”

Messages that could not have a note set will include an error code, i.e.

“OK:11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=ERROR:  
Invalid ID”

## ***setPackageLabel***

### **Description**

This method is used to set the label on a received message, i.e. it is only valid for received messages.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID of the package to retrieve which must be in the same form as that returned by the *<Id>* element in the *Response* to *getFolderContent*.

#### **sLabel**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The note to set on this message.

### **Response**

Type: xsd:string

Possible return values:

“OK” - The note was set.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Wrong format for trackingNumber”)

## ***setPackagesLabels***

### **Description**

This method is used to set the labels on received messages, i.e. it is only valid for received messages.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **sLabels**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: A “;” delimited list of tracking ids and label in the format

“11042-R=Paper label;11045-R=Paper label 2;11046-R=Paper label 3”

The tracking ID of the package to retrieve which must be in the same form as that returned by the <Id> element in the *Response* to *getFolderContent*.

The label element is the label to set on each tracking id

### **Response**

Type: xsd:string

Possible return values:

A “;” delimited list of status returns in the format

“OK;11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=OK”

Messages that could not have a label set will include an error code, i.e.

“OK:11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=ERROR:  
Invalid ID”



## **setPackageError**

### **Description**

This method is used to set the errors on received messages, i.e. it is only valid for received messages. The error note will be accessible to the original sender of the message via the `getExtendedPackageStatusEx` method.

### **Request Parameters**

#### **username**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: `xsd:boolean`

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **sErrors**

Type: `xsd:string`

Requirement: mandatory – cannot be null or empty

Description: A “;” delimited list of tracking ids and note in the format

“11042-R=Paper Error;11045-R=Paper did not print;11046-R=User was wrong”

The tracking ID of the package to retrieve which must be in the same form as that returned by the `<Id>` element in the *Response* to `getFolderContent`.

The note element is the note to set on each tracking id

### **Response**

Type: `xsd:string`

Possible return values:

A “;” delimited list of status returns in the format

“OK:11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=OK”

Messages that could not have a note set will include an error code, i.e.

“OK:11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=ERROR:  
Invalid ID”

## **setPackagePapered**

### **Description**

This method is used to set the “PaperCompleted” status on the message. This is only valid for received messages that are the result of a paper request.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID of the package to retrieve which must be in the same form as that returned by the <Id> element in the *Response* to *getFolderContent*.

### **Response**

Type: xsd:string

Possible return values:

“OK” - The paper completed date was set.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Wrong format for trackingNumber”)

## ***setPackagesPapered***

### **Description**

This method is used to set the “PaperCompleted” status on a collection of messages. This is only valid for received messages that are the result of a paper request.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumbers**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking IDs of the packages on which the papered out status should be set. These are “;” delimited and must be in the same form as that returned by the <Id> element in the *Response* to *getFolderContent*.

“11042-R;11045-R;11046-R”

### **Response**

Type: xsd:string

Possible return values:

A “;” delimited list of status returns in the format

“OK;11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=OK”

Messages that could not have a note set will include an error code, i.e.

“OK;11042-R=OK;11045-R=OK;11046-R=OK;11047-R=OK;11048-R=ERROR:  
Invalid ID”

## ***movePackage***

### **Description**

This method is used to move a package to the “Deleted” or “Archived” folders.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumber**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking ID of the package to retrieve which must be in the same form as that returned by the *<Id>* element in the *Response* to *getFolderContent*.

#### **folderType**

Type: xsd:int

Requirement: mandatory – cannot be null or empty

Description: An integer representing the required folder type. The possible values are:  
2 – archived items  
3 – deleted items

### **Response**

Type: xsd:string

Possible return values:

This method will move any messages that are not already in the given folder and return the IDs of those messages that have been moved and where they have been moved to. The reason is that this method emulates the WebUI in that if a deleted message is deleted again it is moved out of the deleted folder into the inaccessible purged folder.

“OK: 205003-R=DELETED “ - The message with the id “205003-R” was moved to Deleted folder.

“ERROR” - an error occurred. This will be followed by a description of the error (e.g. “ERROR : Wrong format for trackingNumber”)

## ***movePackages***

### **Description**

This method is used to move packages to the “Deleted” or “Archived” folders.

### **Request Parameters**

#### **username**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The username of the registered user who sent the package

#### **password**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The password of the registered user who sent the package

#### **encodedPassword**

Type: xsd:boolean

Requirement: optional – can be null (indicates false)

Description: Indicates whether the password parameter is MD5 encoded

#### **trackingNumbers**

Type: xsd:string

Requirement: mandatory – cannot be null or empty

Description: The tracking IDs of the packages to move. These must be in the same form as that returned by the <Id> element in the *Response* to *getFolderContent*. Multiple ids can be separated by “;” as in

“205003-R; 205005-R; 205013-R”

#### **folderType**

Type: xsd:int

Requirement: mandatory – cannot be null or empty

Description: An integer representing the required folder type. The possible values are:

2 – archived items

3 – deleted items

**Response**

Type: xsd:string

Possible return values:

“OK: 205003-R=DELETED;205006-R=DELETED “ - The messages were moved to the deleted folder

“OK: 205003-R=ARCHIVED;205006-R=ARCHIVED “ - The messages were moved to the archived folder

“OK: 205003-R=ARCHIVED;205006-R=ERROR:Invalid Id “ - The messages were moved to the archived folder, but some were invalid