PHIN EXCHANGE

DEVELOPER GUIDE

Version 1.0

April 30, 2010

VERSION HISTORY

Version #	Implemented By	Revision Date	Reason
0.1	Tony Pruitt	2/8/2010	Draft specification format – Copied from PCA Wiki
0.2	Tony Pruitt	2/11/2010	Complete Draft
0.3	Tony Pruitt	2/17/2010	Edits – Change links to internal document sections
0.4	Tony Pruitt	4/15/2010	Add section of Cascade Mapper and File Exchange services. Update Data Objects and Enumeration values.
0.5	Tony Pruitt	4/21/2010	Final edits to draft for review
1.0	Tony Pruitt	4/26/2010	Final

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

1.1 OBJECTIVES

The objective of the PHIN Exchange (Exchange) Developer Guide document is to provide a detailed description of the Exchange web services and data models to assist developers at state, local, and other public health organizations integrate their local alert communications system with the Exchange in order to send and received PHIN Cascade Alerts.

1.2 SCOPE

The Developer Guide describes the implementation specifications of the PHIN Exchange.

1.3 REFERENCES

[1] PHIN Cascade Alerting Developer Wiki, Version 2.0, January 29, 2010

[2] PHIN Communication and Alerting Implementation Guide, Version 1.3, April 30, 2010

[3] OASIS Common Alerting Protocol, v1.1, October 2005

[4] OASIS Emergency Data Exchange Language (EDXL) Distribution Element, v1.0, May 2006

2 OVERVIEW

Welcome to the PHIN Exchange Developer Guide, a guide for implementing a PHIN Communication and Alerting (PCA) compliant Cascade Alerting system. The guide targets the State and Local PHIN partner managers and developers who will plan, manage, and implement the integration of the partners' local alerting application with the PHIN Exchange services.

2.1 WHAT IS PHIN COMMUNICATION AND ALERTING

PCA is a technical specification for the exchange of public health alert messages between public health organizations. The specification defines the capabilities, technical standards, data standards, and interoperability requirements in order to implement a PHIN compliant PCA system. PCA was designed as a solution for the following opposing emergency alert and response use cases:

- *Need for rapid, comprehensive distribution of information to health workers and emergency responders across multiple jurisdictions and organizations;*

  V**ersus**

- *Respect for autonomous authority of each agency to control the flow of information within its jurisdiction of responsibility and among its workforce.*

2.2 CROSS-JURISDICTIONAL ALERT SCENARIO

This map illustrates a common public health event scenario requiring a jurisdiction to alert bordering jurisdictions. In this scenario the Georgia public health officials have detected a possible E. coli outbreak around a lake in southwest Georgia. Since the source of the outbreak is unknown and the lake borders the states of Alabama and Florida, the Georgia officials must alert the Alabama and Florida public health officials of the possible outbreak. To compound problems, the outbreak occurs over a summer holiday and the officials on duty in each state are minimal.

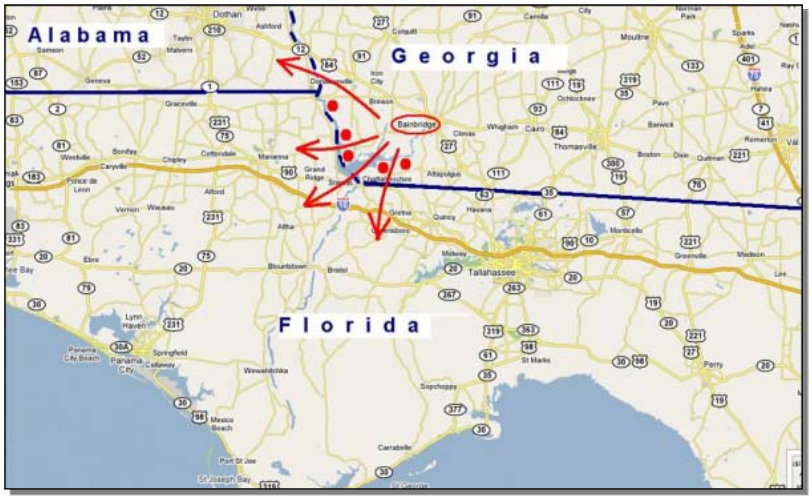

Figure 2-1: Cross-Jurisdictional Alert Scenario

Georgia officials have two choices:

- Georgia can contact the Alabama and Florida officials directly by email or phone; or
- Georgia can send a cascade alert message to both states identifying the public health roles and jurisdictions to alert.

2.3 DIRECT ALERTING VS. CASCADE ALERTING

Direct Alerting

This diagram represents direct alerting from one jurisdiction to persons associated with other jurisdictions. While this may seem like the quickest way to alert public health officials, there are issues and challenges with this method.

Direct Alerting

- Does not respect autonomous authority of local jurisdictions;
- By-passes local emergency response protocols;
- Contact profiles are difficult to keep current; and
- Officials assigned to each public health role are constantly changing.

Cascade Alerting

This diagram represents the cascade alerting solution that achieves the opposing use cases. A jurisdiction's alerting system now sends system-to-system cascade alert messages to other jurisdiction. The receiving jurisdictions processes and sends the alerts to public health officials within their jurisdiction. This method eliminates the issues and challenges with Direct Alerting.

Cascade Alerting

- Local jurisdictional authority is respected;
- Local officials are better adapt to implement response protocols;
- Local directories of officials and role assignments are current; and
- Local alerting application can automatically send alert messages based on local emergency response protocols.

Figure 2-2: Direct vs. Cascade Alerting

2.4 CASCADE ALERTING REQUIREMENTS

In order to exchange messages between disparate systems and technologies, all public health alerting systems must be consistent in:

- the type of information sent to recipients;
- use of standard attributes and vocabularies;
- how alert attributes correspond to system behavior and human processes; and
- logging contact information for historical and reporting purposes.

The PCA specification defines:

- a technical specification for cross-jurisdictional alerting;
- a common set of standard PCA attributes and vocabularies;
- a standard definition for alert message content and distribution information;
- the correspondence of PCA attributes to system functionality;
- the requirements for persistent storage of information about alerts;
- the composition and interpretation of system-to-system (Cascade Alert) messages; and
- the "industry standard" emergency communication message schema (XML).[1]
    - OASIS Common Alerting Protocol (CAP) [3]
    - OASIS Emergency Data Exchange Language (EDXL) [4]

At the same time, each partner must have the maximum possible leeway in choosing an alerting solution that works for their particular circumstance and environment. While any partner can implement their own local solution, CDC has created a new Secure SOA Web Service called the **PHIN Exchange** service to simplify integration of partners alerting system into the Public Health Information Network.

2.5 PHIN EXCHANGE AND CASCADE ALERTING

As a Public Health Information Network (PHIN) solution to provide alert communications between CDC and its PHIN partners, all PCA implementations must integrate with the an approved service to securely transport the Cascade Alert messages between PHIN partners. For smaller message types, such as Cascade Alerts, CDC has created the PHIN Exchange, a CDC hosted Secure Web Service to exchange messages between participating PHIN partners. PHIN partners will no longer need to install PHINMS or the Cascade Lite Utility within their networks in our new **zero-footprint** solution.

PHIN Exchange provides a standardized communication service to route any type of small public health message between multiple partners. Here is a list of features and capabilities PHIN Exchange provides to Cascade Alerting. These are the same features as PHINMS without the installation requirements.

- Route a message to multiple partners;
- Guarantee of delivery;
- Message payload agnostic;

[1] *CAP and EDXL were developed by consortium of emergency communication groups.*

- PKI / Digit certificate authentication;
- Authorizes each message; and
- Automatically encrypts messages during transport.

The following diagram represents the conceptual architecture of the new PHIN Exchange. The PHIN Exchange is a centralized service hosted by CDC. The model shows two PHIN partners (X and Y) and CDC exchanging cascade alert XML payloads. Any PHIN partner can send a cascade alert to other PHIN partners connected to the Exchange. PHIN Partner X utilizes the Cascade Exchange services while PHIN Partner Y utilizes the File Exchange services. Partners have a choice to either create the XML payload files and use the File Exchange to upload and download payload files or use Cascade Exchange to handle all the XML file marshalling

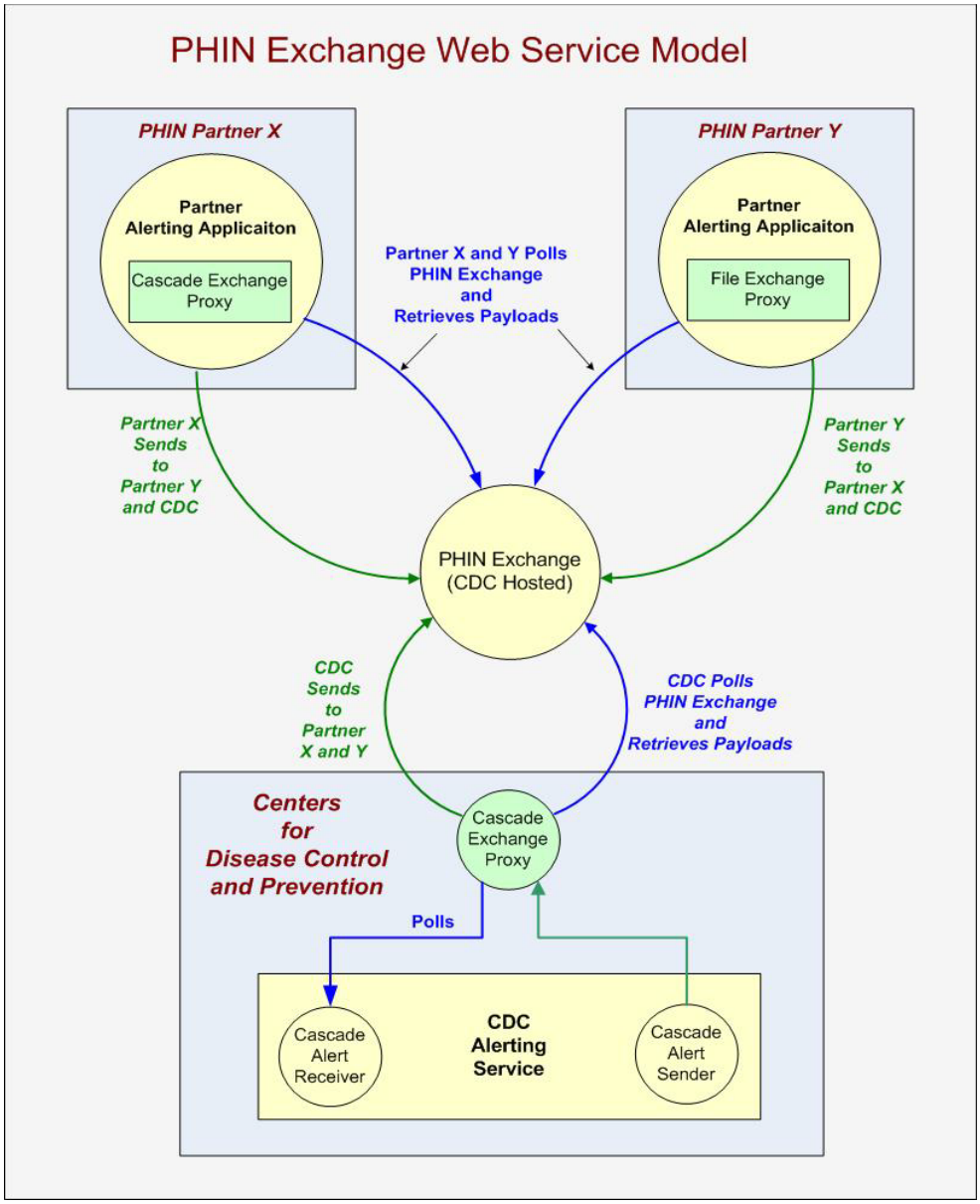

Figure 2-3: PHIN Exchange Model

Each PHIN partner only has to create a Cascade Exchange or File Exchange web service proxy and install a CDC Secure Data Network (SDN) issued client digital certificate to access the service.[2] If you have already integrated with Cascade Lite, there are only minor changes to interface with the new **Cascade Exchange** service.

2.6 WHAT IS THE CASCADE EXCHANGE SERVICE

CDC developed the PHIN Exchange services to simplify and standardize integration with PHIN partner systems and to eliminate any need to install CDC supplied software and utilities within partner networks. The **Cascade Exchange** is a component of the PHIN Exchange that does all the heavy lifting of packaging and sending cascade alerts to PHIN partners and receiving cascade alerts from PHIN partners. This strongly typed web service is almost identical to the Cascade Lite web service interface definition.

Cascade Exchange handles the following tasks:

- Parsing of Cascade Distribution data objects to and from XML files based on the EDXL and CAP XML schemas;
- Post the cascade alert messages and reports to centralized service for delivery to each partner destination;
- Retrieves cascade messages and reports from centralized service sent to your organization
- Queries the centralized service queue for any alert or report sent by or sent to your organization;
- Provides interoperability with any language that support SOAP including .Net and Java; and
- Provides a standard interface (WSDL) for commercial vendors to add cascade alerting to their products.

2.7 HOW CASCADE EXCHANGE WORK

The Cascade Exchange is designed to simplify PHIN partner's implementation of the PCA requirements. This section demonstrates "how" the Cascade Exchange service sends and receives cascade alert messages. Before we discuss the send and receive use cases, let's review the following list of components represented on *Figure 2.3* and their roles in the process.

Component	Role
Partner alerting application	The PHIN partner's alerting system invokes the Cascade Exchange web service methods required to send and receive cascade alerts and acknowledgments.

[2] Additional security information and guidance will be available at time of implementation for Java, Microsoft .Net, PHP languages.

Component	Role
Cascade Exchange Proxy	The client stub is a web service proxy class that performs the low-level processes of initializing and invoking the Cascade Exchange web service methods. This can easily be created by .NET or Java tools.
PHIN Exchange	This is the CDC hosted secure, centralized web service application.
Payload	The cascade alert message is represented by an XML file implementing the EDXL and CAP XML Schema. The PCA Implementation Guide [2] defines the data requirements of cascade alert messages.
SDN Digital Certificate (Not Shown)	Your alerting application will use SDN Digital Certificates to access the PHIN Exchange services. All message requests and responses are SSL encrypted.

2.7.1 Send a Cascade Alert Message

This use case defines the process of sending a Cascade Alert message.

1. Alerting application creates a cascade distribution object with alert message.
2. Alerting application defines the destinations of the message. The routing destinations are maintained and accessible on the PHIN Exchange.
3. Alerting application invokes Cascade Exchange Send method to send the cascade distribution object.
4. Cascade Exchange marshals (parses) the cascade distribution object into the EDXL / CAP XML file.
5. Cascade Exchange stores the message in centralized repository for retrieval by each partner the alert was distributed to.
6. Alerting application invokes Cascade Exchange Query method to get status of message to determine if the recipient partner(s) acknowledge receipt.

2.7.2 Receive a Cascade Alert Message

This use case defines the process of receiving a Cascade Alert message.

1. A partner alerting application or CDC Alert Services sends a cascade alert to you. The alert is stored in your PHIN Exchange repository awaiting your retrieval.
2. Your alerting application polls Cascade Exchange on a regular schedule requesting any new cascade alerts in your repository.
3. Your alerting application request to retrieve a Cascade Alert from the Cascade Exchange.
4. Cascade Exchange marshals (parses) the EDXL/CAP XML file into the cascade distribution object.
5. Cascade Exchange returns the cascade distribution object to your alerting application for processing and distribution.

6. Your alerting application creates an acknowledgment message and calls Cascade Exchange to send acknowledgement to the sender.

2.8 FILE EXCHANGE

A PHIN partner may choose to create and read the EDXL/CAP XML payload files and simply upload to download these files to the PHIN Exchange. The File Exchange services was created for just this purpose. Section 5 of the document defines the specification for the File Exchange Web Service.

2.9 CASCADE MAPPER

The PHIN Exchange also includes a utility called the Cascade Mapper service to create a XML document from the CascadeDistribution data object utilized by the Cascade Exchange or create a CascadeDistribution data object from a properly formed XML document. Section 6 of the document defines the specification for the File Exchange Web Service.

3 DEVELOPER TUTORIAL

3.1 REQUIREMENTS

You must have coding knowledge of either Java or Microsoft .NET in order to use the examples in the tutorial. While having knowledge of web services would be preferable, a strong knowledge of object oriented programming is all that is really needed.

Java

If your development environment is Java, the CDC Cascade Alert team can provide you with a web service client jar to use in your application to access the PHIN Exchange web services. There are tools that you can use to generate a web service client but those are beyond the scope of this document.

Microsoft .Net

Visual Studio .Net can be used to create a web service client by simply accessing the PHIN Exchange web services. Please refer to your Visual Studio documentation for more information.

3.2 CREATING AN INSTANCE OF THE CASCADE EXCHANGE SERVICE PROXY

Before you can make any calls to Exchange methods, you must create an instance of the web service proxy client in your application. The exact technique will vary for each type of programming languages, and in many cases, the version of the language.

Refer to the Security guide section on the PCA Wiki for information on setting up client security for the Cascade Exchange.

Java Example

```
CascadeExchange cascadeExchange =
CascadeExchangeFactory.getCascadeExchangeProxy();
```

3.3 SEND CASCADE ALERT

The following steps define how to create a cascade alert message, identify partners to distribute to and send the message using the PHIN Cascade Exchange Service.

1. Create a cascade distribution object that contains the alert message and cascade alert attributes, public health roles, jurisdictions and individual email addresses.
2. Identify the partners to send the cascade alert.
3. Send the alert.

3.3.1 How to create a cascade alert object?

This code shows you how to create a fully loaded CascadeDistribution object that represents the cascade alert message you wish to send along with distribution information.

Java Example

The following method will create a CascadeDistribution object and call other methods listed below to load portions of the object graph.

```
private CascadeDistribution getEDXL() {
     CascadeDistribution dist = new CascadeDistribution();
     dist.setDistributionID(this.getUniqueId());
     dist.setDateTimeSent(Calendar.getInstance());
     dist.setDistributionStatus(DistributionStatus.TEST);
     dist.setDistributionType(MessageType.ALERT);
     // call getRole to load set of public health roles...
     dist.setRecipientRole(this.getRole());
     String[] mJuri = {"State", "Local"};
     dist.setJurisdictionLevel(JurisdictionLevel.LOCAL);
     dist.setDeliveryTime(DeliveryTime.WITHIN_24_HOURS);
     // call getAddress to load email addresses...
     dist.setExplicitAddress(this.getAddress());
     // call getAread to load jurisdictional information
     dist.setTargetArea(this.getArea);
     // call getRoutes to set the partner destinations
     dist.setRoutes(getDestinations().getRoutes());
     // call getCAP to load the message information that maps to the CAP
schema...
     dist.setAlert(this.getCAP(dist.getDistributionID()));
     return dist;
 }
```

The following method loads the public health roles into a string array.

```
private String[] getRole() {
     String[] roles = new String[5];
     roles[0] = "Public Health Administrator";
     roles[1] = "Emergency Management Coordinator";
     roles[2] = "Chief Epidemiologists";
     roles[3] = "Communicable/Infectious Disease Coordinators";
     roles[4] = "Environmental Health Directors";
     return roles;
 }
```

This method loads fictitious email addresses.

```
private String[] getAddress() {
     String[] addrs = new String[2];
     addrs[0] = "johnsmith@healthdept.gov";
     addrs[1] = "maryjones@healthdept.gov";
     return addrs;
 }
```

The following two methods load the jurisdictional information for Michigan and Indiana counties (FIPS).

```
private CascadeArea[] getArea() {
     CascadeArea[] areas = new CascadeArea[2];
     String[] mifips = {"26027", "26159", "26027"};
     String[] infips = {"18039", "18087", "18151"};
     areas[0] = this.setArea("US", "MI", mifips);
     areas[1] = this.setArea("US", "IN", infips);
     return areas;
 }
```

```
private CascadeArea setArea(String country, String subdivision, String[]
fips) {
     CascadeArea area = new CascadeArea();
     area.setCountry(country);
     area.setSubdivision(subdivision);
     area.setLocCodeUN(fips);
     return area;
 }
```

This method loads the CAP (Message information).

```
 private CascadeAlert getCAP(String distID) {
     CascadeAlert alert = new CascadeAlert();
     alert.setIdentifier("TEST-2008-" + distID);
     alert.setMsgType(MessageType.ALERT);
     alert.setProgram("TEST");
     alert.setConfidentiality(Confidentiality.NOT_SENSITIVE);
     alert.setRestriction("State, Local,and Tribal Public Health
Officials");
     alert.setSender("2.16.840.1.114222.4.20.1.1@cdc.gov");
     alert.setSent(Calendar.getInstance());
     alert.setStatus(MessageStatus.TEST);
     alert.setEventType("Secure Acknowledge");
     alert.setResponseType(ResponseType.NONE);
     alert.setUrgency(Urgency.EXPECTED);
     alert.setSeverity(Severity.SEVERE);
     alert.setCertainty(Certainty.LIKELY);
     alert.setHeadline("PHIN Cascade Alert Message Title/Subject");
     alert.setDescription("Your alert message would go here.");
     alert.setInstruction("Supply any distribution instructions here.");
     alert.setWeb("https://alert.cdc.gov");
     return alert;
 }
```

3.3.2 How to select partner distribution list?

The Exchange will return a list of distribution routes that your application can use to select the partners you wish to distribute the message to. The following code shows you how to retrieve the distribution list.

Java Example

```
RouteList routeList = cascadeExchange.getRoutes();
```

3.3.3 How to send a cascade alert?

As you can see in the examples here, the majority of your effort will be to map the data from your application to the CascadeDistribution object.

To send an alert is easy once your data is mapped. The first step is to create a CascadeDistribution object using the proceeding example.

Java Example

```
CascadeDistribution distribution = getEDXL();
```

Now we need to define the destinations the alert is distributed to. In this example we will send to every route defined in the routing table. Normally your application would select destinations from the route list.

Java Example

```
RouteList routeList = cascadeExchange.getRoutes();
distribution.setRoutes(routeList.getRoutes());
```

One last step to send the alert.

Java Example

```
CascadeDistribution dist = cascadeExchange.sendAlert(distribution);
```

3.3.4 How do I know the cascade alert was successfully sent?

When a CascadeDistribution object is submitted by the sendAlert method, if returns a CascadeDistribution object. If the result field equals FAILURE, the there is a problem with the alert and it was not sent. To determine the cause of the errors, the CascadeDistribution.getErrors() element will return an array of AlertError object listing all errors.

Java Example

```
 if (dist.isValid() == false || dist.getErrors().length > 0) {
     for (int i = 0; i < dist.getErrors().length; i++) {
      AlertError err = dist.getErrors()[i];
      log.debug("Error: " + err.getErrorCode() +", "+ err.getErrorElement()
+", "+ err.getErrorMessage());
     }
 }
```

3.4 READ ACKNOWLEDGMENTS FROM PARTNERS YOU HAVE ALERTED

When you send a cascade alert to partner, the partner's alerting system will acknowledge receipt of message. Your alerting application will have to poll this service periodically to receive the acknowledgment and update your system. The following two steps will query the receiver queue and retrieve the acknowledgment object.

1. Query receiver queue for new acknowledgment messages.
2. Retrieve acknowledgment message

3.4.1 How to retrieve acknowledgment messages from partners?

Java Example

```
private void processAcks() {

    CascadeAlertSearchForm searchForm = new CascadeAlertSearchForm();
    searchForm.setDirection(Direction.IN);
    searchForm.setType(PayloadType.REPORT);
    searchForm.setSeen(Boolean.FALSE);

    CascadeQueueList list = cascadeExchange.query(searchForm);
```

```
    for (int i = 0; i < list.getItem().length; i++) {
        CascadeReport report =
cascadeExchange.getReport(list.getItem()[i].getQueueId);
        if (report.getDistributionType().equals(DistributionType.ACK)) {
            handleAck(report);
        }
    }
}
```

3.5 READ CASCADE ALERT MESSAGES SENT TO YOU BY PHIN PARTNERS

To get and handle messages from PHIN partners, you need to poll the receiver queue for new messages. This can be accomplished by setting the direction on the search form to IN and the type to ALERT. The entries in the results will only be the Exchange transit layer headers, so to get the entire message, you need to call getAlert().

3.5.1 How to query receiver queue for alert messages from partners?

```
private void getNewAlerts() {
    CascadeAlertSearchForm searchForm = new CascadeAlertSearchForm();
    searchForm.setDirection(Direction.IN);
    searchForm.setType(PayloadType.ALERT);
    searchForm.setSeen(Boolean.FALSE);
    searchForm.setDays(10);

    CascadeQueueList list = cascadeExchange.query(searchForm);

    for (int i = 0; i < list.length; i++) {
        handleIncomingAlert(list.getList()[i];
    }
}
```

3.5.2 How to retrieve an alert message from receiver queue?

```
private void handleIncomingAlert(CascadeQueue item) {
    CascadeDistribution distribution =
cascadeExchange.getAlert(item.getQueueId());

    //Your code for handing a cascade alert

   sendAcknowledgement(distribution);
   cascadeExchange.markRead(item.getQueueId());
}
```

3.5.3 How to create and send an acknowledgment message to sender?

```
private void sendAcknowledgement(CascadeDistribution distribution) {
    CascadeReport report = new CascadeReport();
    report.setDistributionID(this.getUniqueID());
    report.setDistributionReference(distribution.getDistributionID());
    report.setDateTimeSent(Calendar.getInstance());
    report.setDistributionStatus(distribution.getDistributionStatus());
    report.setDistributionType(DistributionType.ACK);
    report.setRoute(getRoute(distribution.getSenderID()));
```

```
report.setConfidentiality(distribution.getAlert().getConfidentiality());
    cascadeExchange.sendReport(report);
}
```

3.5.4 How to mark Alerts and Reports as seen?

For the alerts and reports to not show up in the "unseen" queries, they must be marked as seen with the service.

```
private void markSeen(long queueID) {
    cascadeExchange.markRead(queueID);
}
```

4 CASCADE EXCHANGE WEB SERVICE

The PHIN Cascade Exchange Service is a hosted message exchange service for sending and receiving PCA (Cascade Alert) messages and reports. The service is centrally hosted and managed. Authorization and authentication is provided through SSL Certificates issued by the host of the exchange service. The service is implemented as a SOAP-based Web Service running over HTTPS.

4.1 SERVICE DATA MODEL

The Cascade Exchange Data Model objects are listed in the following table. The details for each object can be found in Appendix A and AppendixB. Click the object name to jump to the appendix details.

Object	Description
CascadeDistribution	Primary container for Cascade Alerts
CascadeAlert	Alert and message data for Cascade Alerts
CascadeArea	Targeting and geographical information for Cascade Alerts
CascadeContent	Binary attachments and encapsulated content for Cascade Alerts
DestinationRoute	Routing information for sending Cascade Alerts
RouteList	Container for lists of DestinationRoute data objects
CascadeReport	Report and Acknowledgement data for Cascade Alerts
CascadeQueue	Metadata for listed and queried Cascade Alerts & Reports
CascadeQueueList	Container for lists of CascadeQueues
CascadeValidation	Validation information for Cascade Alerts
CascadeAlertError	Error object for CascadeValidations
CascadeAlertSearchForm	Search for for finding Cascade Alerts & Reports.

4.2 SERVICE METHODS

4.2.1 getDestinations

RouteList getDestinations()

Lists the destinations your client can currently send to. Any destination that supports alerts and reports will be listed. Your own client (destination) will not be listed in this list. Returns a RouteList.

4.2.2 query

CascadeQueueList query(CascadeAlertSearchForm searchForm)

Queries the Cascade Alerts and Reports that you've sent or received. Any alert or report that matches the query parameters will be returned. Returns a CascadeQueueList.

Argument	Type	Description
seachForm	CascadeAlertSearchForm	Search parameters for finding alerts and reports

4.2.3 getAlert

CascadeDistribution getAlert(Integer alertID)

Gets the cascade alert represented by the ID passed in. You can only retrieve alerts that have been sent to you with this method. Returns a CascadeDistribution.

Argument	Type	Description
alertID	Integer	Queue ID of the alert to get.

4.2.4 sendAlert

CascadeDistribution sendAlert(CascadeDistribution cascadeDistribution)

Sends the alert (as a CascadeDistribution) if it is valid. Multiple routes to send the alert to can be submitted. A separate copy will be delivered to each recipient. If the alert is invalid (missing required data, etc.), the CascadeValidation contained in the CascadeDistribution will be marked 'FAILURE' and contain CascadeAlertErrors that describe why the alert couldn't be sent. The submitted CascadeDistribution with the validation information will be returned

Argument	Type	Description
cascadeDistribution	CascadeDistribution	The Cascade Alert to send.

4.2.5 getReport

CascadeReport getReport(Integer reportID)

Gets the cascade report represented by the ID passed in. You can only retrieve reports that have been sent to you with this method. Returns a CascadeReport.

Argument	Type	Description
reportID	Integer	Queue ID of the report to get.

4.2.6 sendReport

CascadeReport sendReport(CascadeReport cascadeReport)

Sends the report if it is valid. If the report is invalid (missing required data, etc.), the CascadeValidation contained in the CascadeReport will be marked ‘FAILURE’ and contain CascadeAlertErrors that describe why the alert couldn’t be sent. The submitted CascadeReport with the validation information will be returned

Argument	Type	Description

cascadeReport	CascadeReport	The Cascade Alert to send.

4.2.7 markRead

CascadeValidation markRead(long queueID)

This method marks the alert or report represented by the queueID as read. This will affect how it shows up in searches done by the query method. If there are any errors marking the alert or report as read, they will be returned and described in the CascadeValidation object.

Argument	Type	Description
queueID	Integer	Queue ID of the alert or report to mark as read.

4.2.8 getXML

String getXML(long queueID)

Gets the raw XML source for the alert or report represented by the queueID. This is useful for testing, or if you want to completely store all alerts on your own system.

Argument	Type	Description
queueID	Integer	Queue ID of the alert or report to get the source XML for.

5 FILE EXCHANGE WEB SERVICE

The File Exchange Service is a hosted payload exchange service for sending and receiving PCA (Cascade Alert) messages and reports and other types of payloads. This service allows PHIN partners who create the PHIN Cascade Alert XML files to submit these files and payloads and to download payloads files they have received from other partners.

The Cascade Exchange utilizes the File Exchange to store and retrieve XML payloads from the strongly type CascadeDistribution object. The service is centrally hosted and managed. Authorization and authentication is provided through SSL Certificates issued by the host of the exchange service. The service is implemented as a Secure SOAP Web Service running over HTTPS.

5.1 SERVICE DATA MODEL

Object	Description
Destination	Defines the destination of the payload sent.
ExchangeQuery	Contains criteria to query the PHIN Exchange repository.
ExchangeQueryOrder	Defines the order of the results of a query.
Payload	Defines the meta-data of a XML payload.
PayloadFile	Binary (Base64) format of the payload.
PayloadValidation	List of PayloadError(s) associated with a failed submission of a XML payload.
User	Defines the user who submitted the payload.

5.2 SERVICE METHODS

5.2.1 getMyDestination

Destination getMyDestination()

Use this method to retrieve your Destination data.

5.2.2 getDestinations

Destination[] getDestinations(string[] activities)

This method will return a complete list of the destinations associated with the list of activities submitted.

Argument	Type	Description
activities	String []	List of PHIN Exchange / SDN activities.

5.2.3 query

ExchangeQuery query(ExchangeQuery query)

The query method returns an ExchangeQuery object that includes a list of payloads sent to your organization and/or received by your organization. The same Exchange Query object is used to submit the query criteria. This method includes the ability to define the sort order of the results and supports pagination control.

Argument	Type	Description
query	ExchangeQuery	The EDXL to transform into a CascadeDistribution.

5.2.4 send

PayloadValidation send(Payload payload)

The send method will validate the Payload object, and if valid, send the payload to the defined destination. If the validation fails, the PayloadValidation object includes a list of validation errors and the payload in not sent.

Argument	Type	Description
payload	Payload	Payload object defines the payload file to send and target destination.

5.2.5 receive

Payload receive(Long payloadID)

The receive method returns a payload object from the PHIN Exchange repository.

Argument	Type	Description
payloadID	Long	The payloadID of the payload file to retrieve. Use the query method to find the payloadID of the payload file.

5.2.6 markRead

PayloadValidation markRead(Long payloadID)

The markRead method will mark a new payload as Read (see PayloadStatus). This allows your system to tag each payload you process as Read so they will be excluded from future queries of Sent payloads.

Argument	Type	Description
payloadID	Long	The payloadID of the payload file to retrieve. Use the query method to find the payloadID of the payload file.

5.2.7 validatePayload

PayloadValidation validatePayload(Payload payload)

This method validates the Payload object and returns a PayloadValidation object. The PayloadValidation object includes a list of validation errors if the validation failed.

Argument	Type	Description

payload	Payload	Payload object defines the payload file to validate.

5.2.8 updateStatus

PayloadValidation updateStatus(Long payloadID, String applicationStatus)

The updateStatus method updates the application status and returns a PayloadValidation object. The validation will fail if the payloadID is invalid. Note that the application status is for your use to correlate with your direct alerting application.

Argument	**Type**	**Description**
payloadID	Long	The payloadID of the payload file to update. Use the query method to find the payloadID of the payload file.
applicationStatus	String	Updates the application status to any value you provide.

6 CASCADE MAPPER WEB SERVICE

The PHIN Cascade Mapper Service is a hosted marshalling and EDXL/CAP validation service for The PHIN Cascade Exchange Service. It contains methods to transform between the CascadeDistribution object and EDXL xml formats and validate CascadeDistribution and CascadeReport objects.

This service does not save or send any data.

6.1 SERVICE DATA MODEL

Object	Description
CascadeDistribution	Primary container for Cascade Alerts
CascadeAlert	Alert and message data for Cascade Alerts
CascadeArea	Targeting and geographical information for Cascade Alerts
CascadeContent	Binary attachments and encapsulated content for Cascade Alerts
CascadeReport	Report and Acknowledgement data for Cascade Alerts
CascadeValidation	Validation information for Cascade Alerts
CascadeAlertError	Error object for CascadeValidations

6.2 SERVICE METHODS

6.2.1 marshalAlert

String marshalAlert(CascadeDistribution distribution)

Transforms the passed in CascadeDistribution into EDXL xml returned as a String. If the CascadeDistribution is invalid (and can't be transformed), an exception will be thrown.

Argument	Type	Description
cascadeDistribution	CascadeDistribution	The CascadeDistribution to transform into EDXL.

6.2.2 marshalReport

String marshalReport(CascadeReport report)

Transforms the passed in CascadeReport into EDXL xml returned as a String. If the CascadeReport is invalid (and can't be transformed), an exception will be thrown.

Argument	Type	Description
report	CascadeReport	The CascadeReport to transform into EDXL.

6.2.3 unmarshalDistribution

CascadeDistribution unmarshalDistribution(String xml)

Transforms the passed in xml String into a CascadeDistribution object. If the xml is too invalid to create a CascadeDistribution, an exception will be thrown.

Argument	Type	Description
xml	String	The EDXL to transform into a CascadeDistribution.

6.2.4 unmarshalReport

CascadeReport unmarshalReport(String xml)

Transforms the passed in xml String into a CascadeReport object. If the xml is too invalid to create a CascadeReport, an exception will be thrown.

Argument	Type	Description
xml	String	The EDXL to transform into a CascadeReport.

6.2.5 validateAlert

CascadeValidation validateAlert(CascadeDistribution distribution)

Performs a validation on the CascadeDistribution object passed in. The validation will be in the CascadeValidation object returned, along with any CascadeAlertErrors.

Argument	Type	Description
cascadeDistribution	CascadeDistribution	The CascadeDistribution to validate.

6.2.6 validateReport

CascadeValidation validateReport(CascadeReport report)

Performs a validation on the CascadeReport object passed in. The validation will be in the CascadeValidation object returned, along with any CascadeAlertErrors.

Argument	Type	Description
report	CascadeReport	The CascadeReport to validate.

APPENDIX A – DATA MODEL OBJECTS

Each of the Cascade Exchange data objects are detailed in this section. Data objects are submitted to and returned by the Cascade Exchange web service methods. Below is a list of the data element attributes.

- Name Name of data element
- XML Attribute Name of EDXL or CAP data element.[3]
- Type The data type of the element. The data types represent both primitive and object data types.
- Notes Identifies required data elements and other important information.

The PCA Implementation Guide contains more information on the XML Attributes.

ACTIVITY

The Activity object identifies the program activity the partner has rights to send payloads to and receive payloads from. This is only used by the File Exchange web service. While the current version only support the CascadeAlert activity. Other activities, such as DirectoryExchange, could be added in the future.

Name	Type	Notes
id	String	Id of activity is Required.
name	String	Name of activity is Required.

CASCADEALERT

The CascadeAlert contains the main alert and message information for the CascadeDistribution object.

Name	XML Attribute	Type	Notes
identifier	CAP.identifier	String	Required.
sent	CAP.sent	Date	*Read-Only*. Populated by the Cascade Exchange Service.

[3] Not all data object list an XML Attributes. These objects are not part of the XML document defining the Cascade Alert content.

confidentiality	EDXLDistribution.contentObject.confidentiality	Confidentiality	Required.
status	CAP.status	MessageStatus	Required.
msgType	CAP.msgType	MessageType	Required.
scope	CAP.scope	Scope	Required.
restriction	CAP.Info.restriction	String	
eventType	CAP.Info.eventCode["Event Type"]	String	
responseType	CAP.Info.responseType	ResponseType	
urgency	CAP.Info.urgency	Urgency	Required.
severity	CAP.Info.severity	Severity	Required.
certainty	CAP.Info.certainty	Certainty	Required.
acknowledge	CAP.Info.parameter["Acknowledge"]	Boolean	
program	CAP.Info.event	String	Required.
senderName	CAP.sender	String	*Read-Only*, populated by the Cascade Exchange.
headline	CAP.Info.headline	String	Required.
description	CAP.Info.description	String	Required.
instruction	CAP.Info.instruction	String	
web	CAP.Info.web	String	
contact	CAP.Info.contact	String	

CascadeAlertSearchForm

The CascadeAlertSearchForm contains the search parameters for the query method.

Name	Type	Notes
direction	Direction	The direction of messages to include in the search.
type	PayloadType	The type of messages to search for.
seen	Boolean	Search for messages that have been marked read or not.
days	Integer	Number of days in the past to search for messages.
startSentTime	Date	Start time of sent time search.
endSentTime	Date	End time of sent time search.

CascadeArea

Describes the area associated with the Cascade Alert.

Name	XML Attribute	Type	Notes
circle	EDXLDistribution.targetArea.circle	String	
polygon	EDXLDistribution.targetArea.polygon	String	
country	EDXLDistribution.targetArea.country	String	
subdivision	EDXLDistribution.targetArea.subdivision	String	
locCodeUN	EDXLDistribution.targetArea.locCodeUN	String[]	FIPS codes. Any populated values must be non-empty.

CascadeDistribution

The CascadeDistribution is the container object for CascadeAlerts. The routing, targeting, and delivery information is contained in the CascadeDistribution.

Name	XML Attribute	Type	Notes
senderID	EDXLDistribution.senderID	String	*Read-Only*. Populated by the Cascade Exchange Service.
dateTimeSent	EDXLDistribution.dateTimeSent	Date	*Read-Only*. Populated by the Cascade Exchange Service.
distributionStatus	EDXLDistribution.distributionStatus	MessageStatus	Required.
distributionType	EDXLDistribution.distributionType	DistributionType	Required.
language	CAP.Info.language	String	Defaults to 'English' if left unpopulated.
deliveryTime	CAP.Info.Parameter["deliveryTime"]	DeliveryTime	Required.
jurisdictionLevels	CAP.Info.parameter["Level"]	JurisdictionLevel[]	
recipientRoles	EDXLDistribution.recipientRole[]	String[]	Role names are not validated against the PCA-specified values.
distributionReference	EDXLDistribution.distributionReference	String	
explicitAddresses	EDXLDistribution.explicitAddress[]	String[]	
targetAreas	EDXLDistribution.targetArea[]	CascadeArea[]	Validated according to the rules in CascadeArea.
cascadeAlert	CAP	CascadeAlert	Required.
cascadeContent	EDXLDistribution.contentObject[]	CascadeContent[]	
destinationRoutes		DestinationRoute[]	At least one required, and all must be valid.
read		Boolean	*Read-Only*, true if the message has been marked read.

validation		CascadeValidation	*Read-Only*, contains validation information for the alert.

CASCADECONTENT

The CascadeContent object allows you to attach binary data to the CascadeDistribution. The data is stored as a byte array.

Name	Type	Notes
confidentiality	Confidentiality	Required.
mimeType	String	
size	Integer	*Read-Only*, populated by the Cascade Exchange.
digest	String	
uri	String	
contentData	byte[]	Required.

DESTINATION

The Destination (File Exchange Only) represents an alerting partner that you can send XML payloads to.

Name	Type	Notes
id	String	Partner Destination ID
name	String	Partner Name
oid	String	Partner OID
activities	Activity[]	List of Activity objects.

FIPS	String	

DESTINATIONROUTE

The DestinationRoute represents an alerting partner that you can send alerts and reports to. A list of destination routes can be retrieved using the getRoutes() method. When attaching them to a CascadeDistribution or CascadeReport object, only the cascadePartnerID field needs to be populated for the Cascade Exchange to be able to identify the route.

Name	Type	Notes
cascadePartnerID	Integer	*Read-Only*.
cascadePartnerOID	String	*Read-Only*.
cascadePartnerName	String	*Read-Only*.
cascadePartnerFIPS	String	*Read-Only*.

CASCADEQUEUE

The CascadeQueue is the metadata object about a sent payload containing the searchable information. It is the object that is aggregated by the query() method.

Name	Type	Notes
id	Integer	The Queue ID of the message.
direction	Direction	IN for received messages, OUT for sent messages.
payloadType	PayloadType	The type of message (alert, report).
queueDate	Date	Date the message was put in the system.
senderID	Integer	The ID of the sender.

senderName	String	The name of the sender.
read	Boolean	True if the message has been marked read, false otherwise.
sentTime	Date	Date the message was sent.
receiveTime	Date	Date the message was marked read`.

CASCADEQUEUELIST

The CascadeQueueList object is simply a container for a list of CascadeQueue objects returned from the query method.

Name	Type	Notes
items	CascadeQueue[]	*Read-Only*.
listDate	Date	*Read-Only*.

CASCADEREPORT

The CascadeReport object represents the PCA report message.

Name	Type	Notes
distributionID	String	Required.
senderID	String	*Read-Only*, populated by the Cascade Exchange.
dateTimeSent	Date	*Read-Only*, populated by the Cascade Exchange.
distributionStatus	MessageStatus	Required.
distributionType	ReportType	Required.
distributionReference	String	Required.

confidentiality	Confidentiality	Required.
route	DestinationRoute	Required.
sender	DestinationRoute	*Read-Only*, populated by the Cascade Exchange.
read	Boolean	
validation	CascadeValidation	*Read-Only*.

CASCADEVALIDATION

The CascadeValidation holds SUCCESS/FAILURE information about a Cascade Exchange action. It contains a list of any errors that might have occurred.

Name	Type	Notes
queueIDs	Integer[]	A list of the IDs that were affected by the action (multiple ones in the case of sending an alert to many routes).
result	Result	Success or Failure.
errors	CascadeAlertError[]	

PAYLOAD

The Payload object defines the meta-data (attributes) of the payload file it is associated with.

Name	Type	Notes
id	Long	*Unique payload id*
activity	Activity	*PHIN Exchange Program Activity*
applicationStatus	String	*Status of the payload transport process*

name	String	*Name of payload file*
payloadFile	PayloadFile	*The payload binary object*
payloadStatus	PayloadStatus	*Enumeration (see below)*
readDate	DateTime	*Date payload read by receiver*
receiver	Destination	*Destination object identifies receiver.*
receiverUser	User	*User object identifies receiving user.*
sender	Destination	*Destination object defines sender.*
senderUser	User	*User object identifies sending user.*
sentDate	DateTime	*Date the payload was sent.*

EXCHANGEQUERY

The ExchangeQuery object is utilized by the File Exchange service as criteria to query the PHIN Exchange repository.

Name	**Type**	**Notes**
activityIDs	Long[]	*Array of activity ids*
applicationStatuses	String[]	*Array of application status values*
direction	queryDirection	*Enumeration*
filename	String	*Payload filename*
orders	ExchangeQueryOrder[]	*Array of ExchangeQueryOrder objects*
pageNumber	Int	*Current results page number*
pageSize	Int	*Number of result pages*

payloadStatuses	PayloadStatus[]	*Array of payload status values*
readDateStart	DateTime	*Begin Date of read date range*
readDateEnd	DateTime	*End Date of read date range*
receivingDestinations	Long[]	*Array of Receiving Destination IDs.*
resultSize	Long	*Read Only: Size of payload files*
results	Payload[]	*Read Only: Array of payload objects*
sendingDestinations	Long	*Array of Sending Destination IDs.*
sendDateStart	DateTime	*Begin Date of send date range*
sendDateEnd	DateTime	*End Date of send date range*

EXCHANGEQUERYORDER

The ExchangeQueryOrder object defines the results order of a file exchange query.

Name	Type	Notes
order	Order	*Enumeration*
property	ExchangeQueryProperty	*Enumeration*

PAYLOADERROR

The PayloadError object lists error found in the PayloadValidation object when submitting a payload file.

Name	Type	Notes
message	String	*Error message*

object	String	*Name of object containing error.*
type	payloadErrorType	*Enumeration*

PAYLOADFILE

The PayloadFile object contains the binary representation of the payload file.

Name	Type	Notes
id	Long	*Unique payload file id.*
binary	Base64Binary	*Binary payload file.*

PAYLOADVALIDATION

The PayloadValidation object contains list of Payload errors. Note that a payload is not transmitted if any validation errors exist.

Name	Type	Notes
payloadID	Long	*Unique payload id.*
result	Result	*Enumeration if validation was a success or failure.*
payloadError	PayloadError[]	*List of payload validation errors (if Failure)*

ROUTELIST

The RouteList object is simply a container for the DestinationRoute objects returned from the getRoutes method.

Name	Type	Notes
routes	DestinationRoute[]	*Read-Only.*

listDate	Date	*Read-Only*.

USER

The User object is used to identify the sending and receiving users of File Exchange payload files.

Name	**Type**	**Notes**
id	Long	*Unique id of user*
name	Date	*User Name*
emailAddress	String	*User Email Address*
phoneNumber	String	*User Phone Number*
principal	String	*User Principal*
destination	Destination	*Destination the user is associated with.*

APPENDIX B. DATA ELEMENT ENUMERATION VALUES

Enumerations are required data element values as defined by the EDXL and CAP schemas and the Cascade Exchange. The following code illustrates how an enumeration is set in Java. The CascadeAlertSearchForm.Direction value is set to Direction.IN enumeration value and the CascadeAlertSearchForm.Type value is set to PayloadType.ALERT enumeration value.

```
private void getNewAlerts() {
    CascadeAlertSearchForm searchForm = new CascadeAlertSearchForm();
    searchForm.setDirection(Direction.IN);
    searchForm.setType(PayloadType.ALERT);
    searchForm.setSeen(Boolean.FALSE);
    searchForm.setDays(10);

    CascadeQueueList list = cascadeExchange.query(searchForm);

    for (int i = 0; i < list.length; i++) {
        handleIncomingAlert(list.getList()[i];
    }
}
```

Below is a listing of the Enumeration Types and Values utilized in the PHIN Exchange services.

CERTAINTY	CONFIDENTIALITY	DELIVERYTIME
• OBSERVED • LIKELY • POSSIBLE • UNLIKELY • UNKNOWN	• SENSITIVE • NOT_SENSITIVE	• WITHIN_15_MINUTES • WITHIN_60_MINUTES • WITHIN_24_HOURS • WITHIN_72_HOURS

DIRECTION • IN • OUT	**DISTRIBUTIONTYPE** • REPORT • UPDATE • CANCEL • REQUEST • RESPONSE • DISPATCH • ACK • ERROR	**EXCHANGEQUERYPROPERTY** • NAME • PAYLOAD_STATUS • APPLICATION_STATUS • SENT_DATE • READ_DATE • SENDER_NAME • RECEIVER_NAME • ACTIVITY_NAME
ERRORTYPE • REQUIRED_FIELD • INVALID_ROUTE • INVALID_DATA • DATA_LENGTH	**JURISDICTIONLEVEL** • FEDERAL • STATE • LOCAL	**MESSAGESTATUS** • ACTUAL • EXERCISE • TEST • SYSTEM
MESSAGETYPE • ALERT • UPDATE • ERROR • CANCEL	**ORDER** • ASC • DESC	**PAYLOADERRORTYPE** • RECEIVER_REQUIRED • ACTIVITY_REQUIRED • NAME_REQUIRED • ACTIVITY_NOT_SUPPORTED

PAYLOADSTATUS • SENT • READ • ERROR • ARCHIVED	**PAYLOADTYPE** • ALERT • REPORT • BOTH	**QUERYDIRECTION** • SENT • RECEIVED
REPORTTYPE • ACKNOWLEDGE • REPORT • ERROR	**RESPONSETYPE** • SHELTER • EVACUATE • PREPARE • EXECUTE • MONITOR	**RESULT** • SUCCESS • FAILURE
SCOPE • PUBLIC • RESTRICTED • PRIVATE	**SEVERITY** • EXTREME • SEVERE • MODERATE • MINOR • UNKNOWN	**URGENCY** • IMMEDIATE • EXPECTED • FUTURE • PAST • UNKNOWN