Spot Commercial Account Data Push

User Guide

Hari Gangadharan

Version 1.0.1

Published on: Monday, January 06, 2008

Contributors

Jonathan Beech
Santhosh John

2008

<u>THIS PAGE INTENTIONALLY LEFT BLANK</u>

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Introduction

About this document

This document explains the process required to setup the SPOT message data transfer between the SPOT servers and the SPOT customer's servers. This data transfer service is owned and operated by SPOT and will be referred to as SPOT Data Push Service or simply as the Data Push Service in the remainder of this document. This document is intended for the users of the SPOT Data Push Service. Some sections are indented for designers and programmers who write applications that consume data from SPOT Data Push Service.

THE KEY WORDS "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", AND "OPTIONAL" IN THIS DOCUMENT ARE TO BE INTERPRETED AS DESCRIBED IN RFC 2119 [http://www.ietf.org/rfc/rfc2119.txt].

Description

SPOT Data Push Service is designed for SPOT customers that have more than one SPOT unit in their account and require the messages be transferred to their servers as and when they arrive. Data Push uses eXtended Markup Language (XML) to transfer the messaging data. This service only pushes the incremental data periodically and will retry for a set period of time (called as Data Push Retry Duration) if the customer's service is unavailable.

Customers of the SPOT data push service will have the ability to login to the SPOT account pages and update the Data Push Service properties like where the data is to be sent. They will also have the ability to send mock test data, do data recovery and to look at the data transfer errors without the help of SPOT technical support.

More about Data Push

Understanding Supported Protocols

SPOT Data Push service currently supports data transfer using the following protocols and the customer must choose one of the following protocols. Additional requirements for each protocol are provided in following sections.

1. **Hyper Text Transfer Protocol / Hyper Text Transfer Protocol Secure (HTTP / HTTPS)** – SPOT Data Push service will POST an XML file on a Callback listener Web application. Customer must maintain an Internet accessible Web application that receives XML file.

The customer must provide through the Data Push Setup page, the full URL of this service.
NOTE: HTTP protocol is not secure and hence it is not recommended. If the protocol is HTTPS then customer must make sure that the certificate used in the server is from a valid Root Certification Authority like Verisign or Thawte. SPOT will not support self-signed certificates.

2. **Secure File Transfer Protocol (SFTP)** – Customer must maintain an internet accessible server with SFTP service enabled. SPOT will put message data as XML files in the directory specified by the user. In this case Customer must provide in the Data Push Setup page, the hostname of the server, the user name and the password to access the server using SFTP and the directory to which the files are to be sent to.

3. **File Transfer Protocol (FTP)** – This protocol is not recommended due to the security risks. Hence customers should consider using SFTP or HTTPS instead of FTP. If FTP is chosen as the protocol, then customer must maintain a server accessible from SPOT network or Internet. This server must have the FTP service enabled. Above this customer must enter in the Data Push Setup page, the hostname of the server, the username and password to access server using FTP and the directory to which the files are to be sent to.

SPOT has plans to support more protocols in the future. Request to support other protocols may be made to the SPOT product team.

Apply for Data Push Service

After choosing the protocol, the following are to be completed by the customer before the SPOT Data Push Service can be enabled:

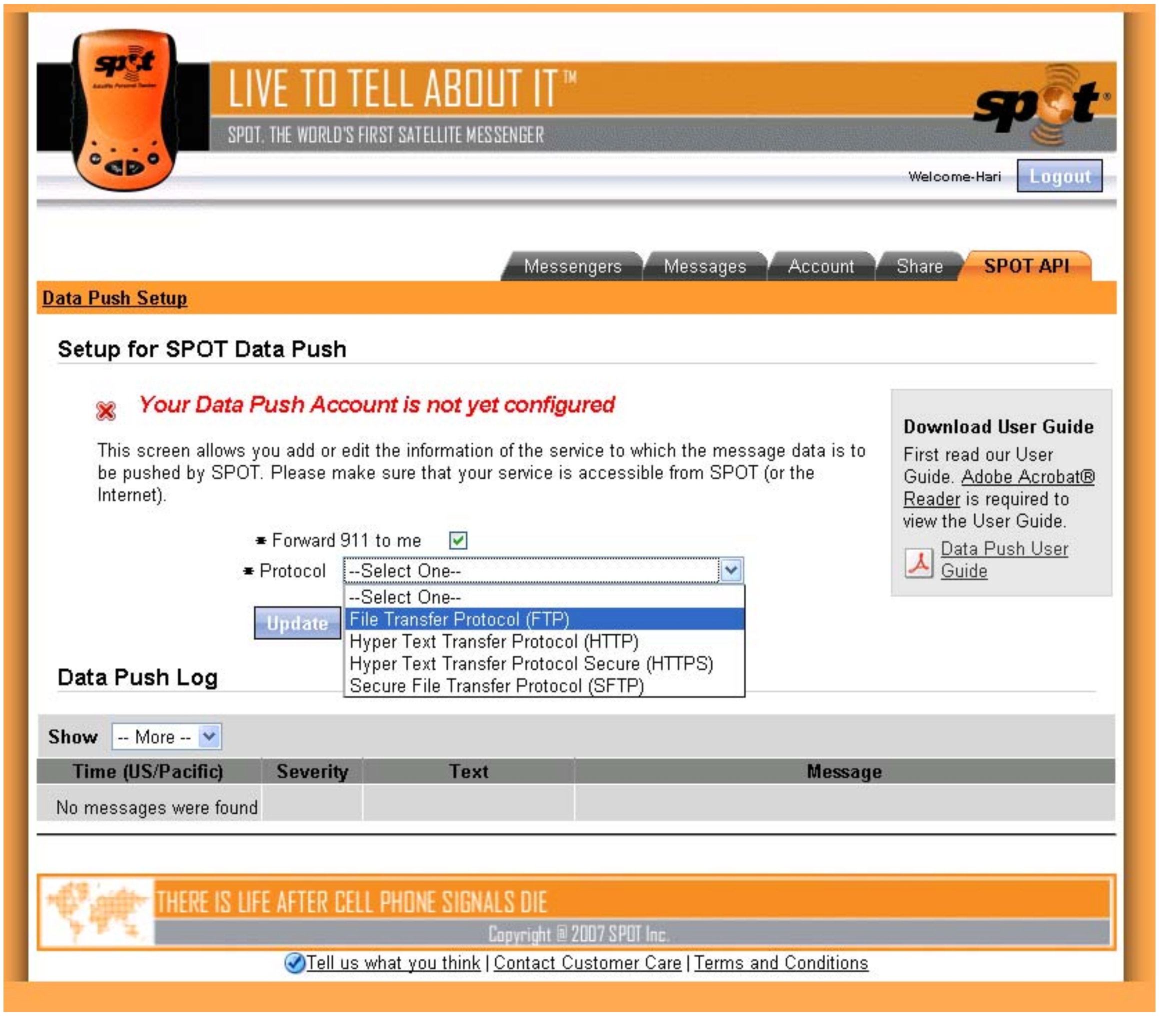
- Enable Data Push Feature in the account by signing the SPOT Commercial Data Access Agreement and paying the setup fee. Contact SPOT Product team on more information on this.

SPOT Product team will convert the customer as a Commercial Data Push Customer and will communicate that to the customer. At that time customer has to configure the Data Push by providing the customer's service information in SPOT Account login, https://login.findmespot.com. To learn more about how to configure the Data Push service, refer Sec. "Configuring Data Push".

Configuring Data Push

Once the SPOT Data Push is enabled in the customer's account, the customer gets access to the SPOT API tab and the web pages in them. When a customer clicks on SPOT API tab, the Data Push Setup page is shown as default. If the customer has not completed the setup of the Data Push, a message is shown indicating that the Data Push configuration is not complete.

FIGURE 1 – DATA PUSH SETUP PAGE (ACCOUNT NOT YET CONFIGURED)


LIVE TO TELL ABOUT IT™
SPOT. THE WORLD'S FIRST SATELLITE MESSENGER
Welcome-Hari
Logout
Messengers
Messages
Account
Share
SPOT API
Data Push Setup

Setup for SPOT Data Push

☒ *Your Data Push Account is not yet configured*

This screen allows you add or edit the information of the service to which the message data is to be pushed by SPOT. Please make sure that your service is accessible from SPOT (or the Internet).

* Forward 911 to me ☑

* Protocol --Select One--

--Select One--
File Transfer Protocol (FTP)
Hyper Text Transfer Protocol (HTTP)
Hyper Text Transfer Protocol Secure (HTTPS)
Secure File Transfer Protocol (SFTP)
Update

Download User Guide

First read our User Guide. Adobe Acrobat® Reader is required to view the User Guide.

Data Push User Guide

Data Push Log

Show -- More --

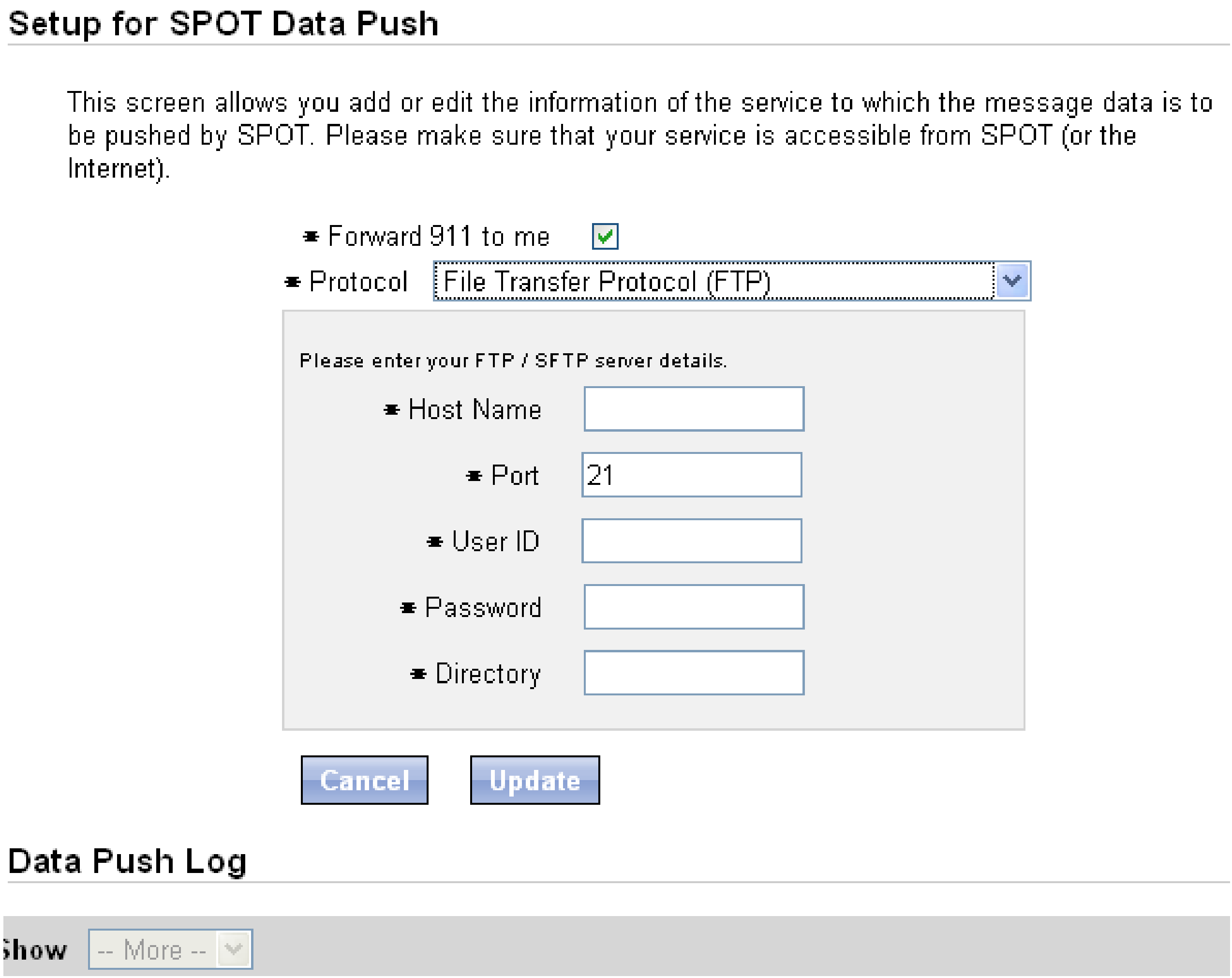
Time (US/Pacific)	Severity	Text	Message
No messages were found			

THERE IS LIFE AFTER CELL PHONE SIGNALS DIE
Copyright © 2007 SPOT Inc.
Tell us what you think | Contact Customer Care | Terms and Conditions

Initially the customer has to choose the protocol for the Data Push Service. To know more about the protocols supported by the Data Push Service, please refer Sec. "Understanding Supported Protocols".

Once a protocol is chosen, the customer will be prompted to enter additional configuration options pertaining to that protocol. For example, if FTP is chosen, the customer is asked to enter the FTP host name, port number, user name, password and directory to which the files are to be sent. In case of HTTP / HTTPS, the customer is prompted for the URL of the customer's callback service.

FIGURE 2 – DATA PUSH SETUP PAGE (ENTERING PROTOCOL SPECIFIC INFORMATION)



Setup for SPOT Data Push

This screen allows you add or edit the information of the service to which the message data is to be pushed by SPOT. Please make sure that your service is accessible from SPOT (or the Internet).

* Forward 911 to me ☑
* Protocol: File Transfer Protocol (FTP)

Please enter your FTP / SFTP server details.

* Host Name:
* Port: 21
* User ID:
* Password:
* Directory:

Data Push Log

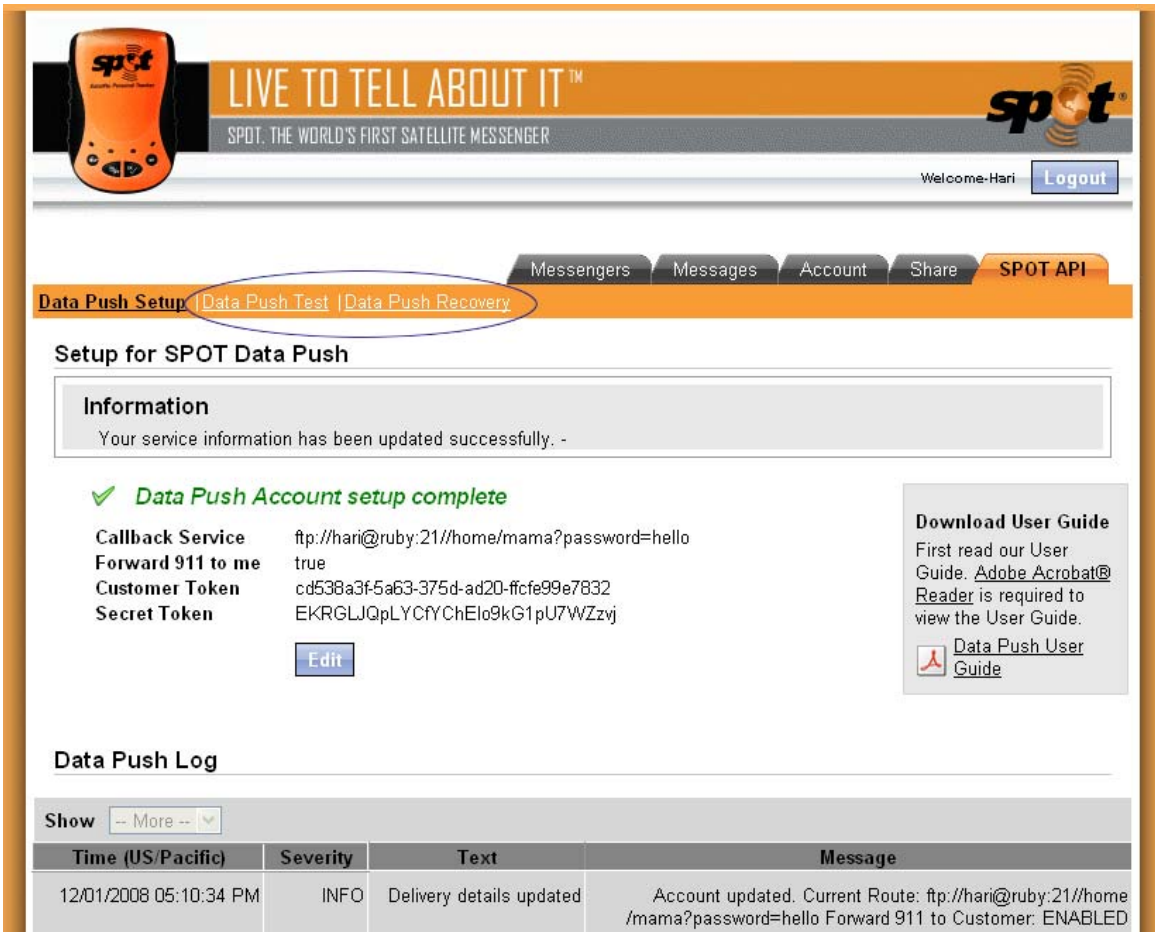
Show: -- More --

Once the needed information is entered and the "Update" button is pressed to confirm the entry, a message indicating the completion of configuration is shown. Also, at this point the customer will be shown the following:

- The customer token
- The customer secret token

These tokens are applicable mainly for the HTTP/HTTPS protocols and are not needed for FTP or SFTP. These are used to reduce replay attacks if HTTP/HTTPS are used as the data transfer protocol. The X-WSSE headers are created using these tokens. Sec. “Reducing the security vulnerabilities” provides more information on how to secure the Callback service.

FIGURE 3 – DATA PUSH SETUP COMPLETE



Setup for SPOT Data Push

Information

Your service information has been updated successfully. -

✓ *Data Push Account setup complete*

Callback Service	ftp://hari@ruby:21//home/mama?password=hello
Forward 911 to me	true
Customer Token	cd538a3f-5a63-375d-ad20-ffcfe99e7832
Secret Token	EKRGLJQpLYCfYChElo9kG1pU7WZzvj

Download User Guide

First read our User Guide. Adobe Acrobat® Reader is required to view the User Guide.

Data Push User Guide

Data Push Log

Time (US/Pacific)	Severity	Text	Message
12/01/2008 05:10:34 PM	INFO	Delivery details updated	Account updated. Current Route: ftp://hari@ruby:21//home /mama?password=hello Forward 911 to Customer: ENABLED

Once the setup is complete, the customer will have access to two more sub-tabs, the Data Push Test and Data Push Recovery. These sub-tabs provide access to the Data Push Test Tool and Data Push Recovery Tool respectively. For more information on these tools refer Sec. “Testing and Recovery”.

Next step will be to create a service to consume the data. The details are available in Sec. “Creating the Consumer”.

Note: It is also important that the customer tests the SPOT’s connection to the customer’s server (hereafter referred as **route**) by sending test data with the help of the Data Push Test Tool. This is to be repeated any time the customer changes the protocol or service information in the Data Push Setup screen. Otherwise it may take hours before the data push service starts using the new values. Refer Sec. “Test Messages” for more information on testing the routes.

Creating the Consumer

XML Format

The XML is structured such a way to send multiple messages in a single transfer. Refer to the XML sample and XML schema in the Appendix A - XML Format. The following are the description of each and every data element in the XML:

Element Name	Type	Description
messageList	XML Root	This is the root element of the XML
header	Complex Type	The header element contains the error information and other header fields.
totalCount	Numeric	Contains the number of messages in this request
mode	String	LIVE indicates LIVE data or data that is transmitted from the Messenger. “TEST” and “RECOVERY” for customer initiated test and data recovery.
desc	String	A Description of the transfer (currently not used)
errors	Complex Type	Contains error information if present. This is currently not used.
message	Complex Type	Represents a single message - this can repeat multiple times (as many as the count in totalCount element).
Id	Numeric	This will be a unique identifier for the message. For customer generated test data, this will be the Test ID.
esn	String	The ESN# of the messenger that sent this message.
esnName	String	The Alias (or Name) of the messenger that sent this message.

messageType	String	Can be TRACK, OK, HELP, HELP-CANCEL, 911, 911-CANCEL, 911 HELP, 911-CANCEL HELP, 911 HELP-CANCEL, 911-CANCEL HELP-CANCEL The last 4 messages occurs when HELP and 911 buttons are activated at the same time in a messenger.
messageDetail	String	Message associated with the message type.
timestamp	Date Time	Date and Time in GMT when this message is received.
timeInGMTSecond	Numeric	Seconds from Epoch in GMT
latitude	Float	The latitude sent with message
longitude	Float	The longitude sent with message
nearestTown	String	Nearest Town using Reverse Geocoding – will be missing or blank if the Reverse Geocoding was not done. Reverse Geocoding is not done if servers are busier than usual.
nearestTownDistance	String	The distance from the nearest town. Will be missing or blank (same case as the nearestTown).

This XML can be parsed using a Parser or can be unmarshalled to objects using a XML binding product like JAXB or Castor.

HTTP / HTTPS Callback listener

For handling the data transfer using HTTP and HTTPS protocol, the customer must create a Dynamic Callback Web program like a Servlet (see sample Servlet in Sec. Appendix B – Sample Java Code to verify X-WSSE headers). The SPOT Data Push Service will do a POST on this service with XML as the body of the POST. SPOT Data Push Service also provides WSSE headers which may be used to authenticate the calls.

SFTP / FTP

In this case XML files will be written to the directory specified during the Data Push Service Setup. As explained before, each XML file may contain more than one message and an XML file is sent if new messages are received. A backend job (like cron jobs) or a background process should be created by the customer to consume the files sent by SPOT Data Push Service.

Configuring Firewall

The Customer must make sure that the Firewall ports are to be opened in the Customer's side to allow SPOT to access the service. In some cases the customer may have to map the server port to a different firewall port. The hostname or URL provided to SPOT during the Data Push Setup should be the one with which the service can be accessed from the Internet or SPOT network. In many cases, the customer's internal host name and port may not be accessible from the Internet.

Reducing the security vulnerabilities

Data transferred using HTTP is not encrypted and hence the privacy of the data cannot be guaranteed. Hence SPOT recommends using HTTPS.

Even in HTTPS, attacks can occur on the user's callback service. The most common attacks are Denial of Service (DoS) and the replay attacks. SPOT uses WSSE UsernameToken security standard as implemented in Atom protocol. SPOT always provides the following headers to the data sent to HTTP/HTTPS Callback listeners.

```
Authorization: WSSE profile="UsernameToken"
X-WSSE: UsernameToken Username="sdfwew23q",
PasswordDigest="quR/EWLAV4xLf9Zqyw4pDmfV9OY=",
Nonce="d36e316282959a9ed4c89851497a717f", Created="2008-06-
13T16:41:32Z"
```

Here the following are fields:

Field	Calculation logic
Username	This is the customer token (provided by SPOT during API setup)
Nonce	A random number base64 encoded
Created	The creation time of message in GMT time zone
PasswordDigest	base64encode(sha1(Nonce + Created + customerSecret)) Here customerSecret is the Customer Secret Token provided by SPOT during API setup.

Callback (Customer's) service's responsibility

Callback service may ignore these headers but it is possible that the service may be attacked by external parties. In the customer's service the following should be implemented to reduce the security attacks:

1. Customer's service will retrieve the X-WSSE header and parse out the fields embedded.
2. The Base64 decoded nonce, created, the user name (customer token) and customer secret token will be combined and encoded using SHA-1 and Base64.
3. If that value is not same as PasswordDigest, the customer's service should respond with a 401 Unauthorized.
4. To reduce replay attacks the nonce is to be saved in the customer's side in a database for a specified interval known as timestamp freshness interval. If a nonce that is saved is encountered in a new request then the request is to be rejected with 401 Unauthorized. This is to eliminate the replay attacks.
5. If the created time is too old then also the message is to be rejected. The rule of thumb is that the messages beyond the save-period (see item 4 above) of nonce is rejected. SPOT recommends the timestamp freshness interval as 1 hour. The customer can choose a higher timestamp interval provided the nonce is saved for the same period of time. Freshness interval below the suggested minimum may result in increased rejection of valid messages.

If there was no rejection, the callback service can parse the body of the POST or push the body to a message queue for further processing. If there were no errors, the callback service must return a HTTP 200 OK status.

Other ways to reduce the security vulnerabilities

The following are some of the other ways the security vulnerability can be reduced:

1. **IP based reject:** Rejecting the requests if it did not originate from the Globalstar/SPOT network. The disadvantages are the user service is to be updated or reconfigured when Globalstar changes the ISP.
2. **Virtual Private Networking (VPN):** VPN is one of the most secure transmission methods and is especially recommended if the customer uses FTP for data transfer. The disadvantage however is the startup cost and complexity of configuration.

Automatically identifying Test Messages and Lost Transmissions

As in the case of any server to server data transmission, some parts of data can go into an undeliverable queue if the receiver is offline or if the receiver has technical difficulties. As explained before SPOT data push tries to redeliver multiple times until giving up.

SPOT provides the following fields for identifying the lost transmissions and test messages:

HTTP Header Field	Values	FTP/SFTP
routerMessageSeq	Zero padded numeric sequence number starting with 1. Example: 0000001, 0000002, etc	Character 2-8 of the file name
routerMessageMode	TEST -> A test message RECOVERY -> Message originated by a recovery request LIVE -> Normal Message	Character 1 of the file name T -> TEST R -> RECOVERY S -> LIVE

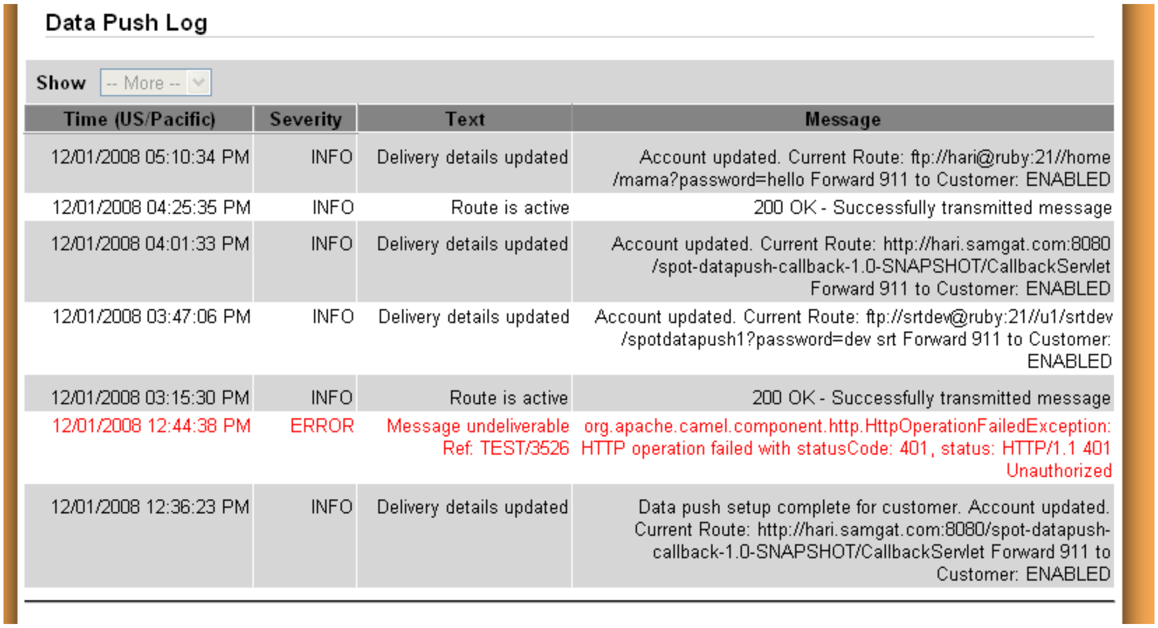
As shown in table above, SPOT uses a sequence number in transmitted data so that the consumer can identify parts of data that is lost. In case of HTTP/HTTPS transfers, the HTTP header routerMessageSeq can be used to identify the lost transmissions. For example if the transmissions with routerMessageSeq 0002002, 0002006 and 0002005 were received but the messages with routerMessageSeq 0002003 and 0002004 were missing even after the SPOT published retry duration then the consumer should assume that the messages with routerMessageSeq 0002003 and 0002004 were lost. Recovery request can be send with a start time and end time. The start time and end time for the recovery request can be figured out using the timeInGMT field of messages in the received parts that are before and after the lost parts.

Similarly the FTP/SFTP transfer also uses the routerMessageSeq. However this value is in the name of the file. In other words the file names for FTP/SFTP transfers are like S0002002.xml, S0002003.xml, and so on.

Data Push logs in the Data Push Setup screen also provides enough indication on the failures. If a message is undeliverable then the text of the message will be similar to “Message undeliverable Ref: LIVE/00002”. In case a message is undeliverable but SPOT has scheduled a retry then the message will be similar to “Message undeliverable (will retry) Ref: LIVE/00002”. From this a customer can manually verify whether SPOT has given up retries for a message or not. Here the 00002 is the message sequence.

Data Push Log

FIGURE 4 – DATA PUSH LOG



Data Push Log

Time (US/Pacific)	Severity	Text	Message
12/01/2008 05:10:34 PM	INFO	Delivery details updated	Account updated. Current Route: ftp://hari@ruby:21//home /mama?password=hello Forward 911 to Customer: ENABLED
12/01/2008 04:25:35 PM	INFO	Route is active	200 OK - Successfully transmitted message
12/01/2008 04:01:33 PM	INFO	Delivery details updated	Account updated. Current Route: http://hari.samgat.com:8080 /spot-datapush-callback-1.0-SNAPSHOT/CallbackServlet Forward 911 to Customer: ENABLED
12/01/2008 03:47:06 PM	INFO	Delivery details updated	Account updated. Current Route: ftp://srtdev@ruby:21//u1/srtdev /spotdatapush1?password=dev srt Forward 911 to Customer: ENABLED
12/01/2008 03:15:30 PM	INFO	Route is active	200 OK - Successfully transmitted message
12/01/2008 12:44:38 PM	ERROR	Message undeliverable Ref: TEST/3526	org.apache.camel.component.http.HttpOperationFailedException: HTTP operation failed with statusCode: 401, status: HTTP/1.1 401 Unauthorized
12/01/2008 12:36:23 PM	INFO	Delivery details updated	Data push setup complete for customer. Account updated. Current Route: http://hari.samgat.com:8080/spot-datapush-callback-1.0-SNAPSHOT/CallbackServlet Forward 911 to Customer: ENABLED

The Data Push Log section provided in the Data Push Setup Page allows users to view errors encountered during the transmission of data to the customer's server. Any errors occurred during the processing of messages will be displayed in this section. SPOT will report all the errors but will not report all the successful transmissions. Similarly the changes to configuration will also be reported in this section.

As explained before, all error messages will have "Ref: XXXX/9999" in the text (where XXXX can be LIVE, RECOVERY or TEST and 9999 is the message sequence number). If a message is undeliverable then the text of the message will be similar to "Message undeliverable Ref: TEST/32002". In case a message is undeliverable but SPOT has scheduled a retry then the message will be similar to "Message undeliverable (will retry) Ref: LIVE/00002". From this a customer can figure out whether SPOT has given up retries for a message or not.

Please refer "Appendix C – Common error messages and reason" for common error messages and possible solutions.

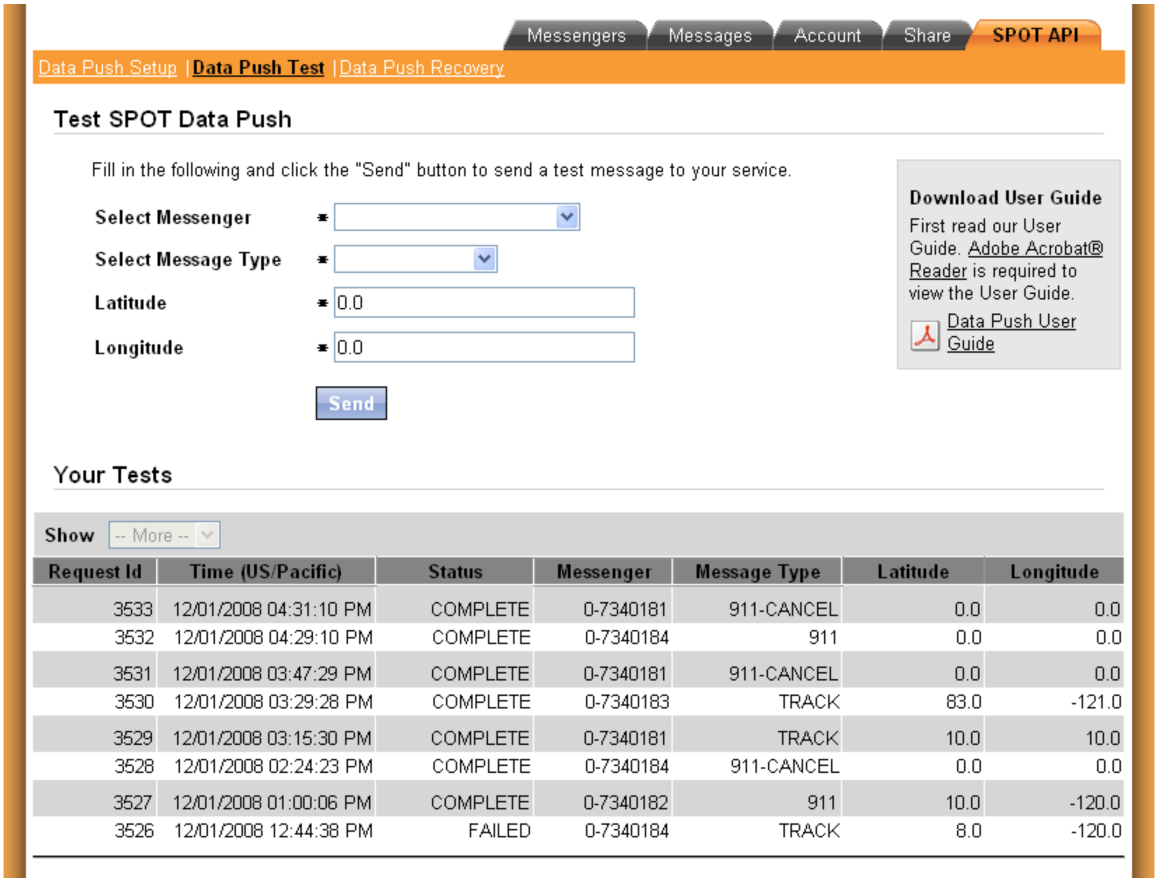
Testing and Recovery

SPOT currently provides UI based recovery and testing tool. Customers can request data recovery or do testing using these tools. These tools are available under the SPOT API Tab.
Note: Testing and Recovery services do not have the same redelivery policy as the live data. Hence SPOT will only retry less than a minute for Test / Recovery Requests. Hence the customer

should make sure that the customer's server/service is working properly before doing a Test or Recovery request.

Test Messages

FIGURE 5 – DATA PUSH TEST



Test SPOT Data Push

Fill in the following and click the "Send" button to send a test message to your service.

Select Messenger *	
Select Message Type *	
Latitude *	0.0
Longitude *	0.0

Download User Guide

First read our User Guide. Adobe Acrobat® Reader is required to view the User Guide.

Data Push User Guide

Your Tests

Request Id	Time (US/Pacific)	Status	Messenger	Message Type	Latitude	Longitude
3533	12/01/2008 04:31:10 PM	COMPLETE	0-7340181	911-CANCEL	0.0	0.0
3532	12/01/2008 04:29:10 PM	COMPLETE	0-7340184	911	0.0	0.0
3531	12/01/2008 03:47:29 PM	COMPLETE	0-7340181	911-CANCEL	0.0	0.0
3530	12/01/2008 03:29:28 PM	COMPLETE	0-7340183	TRACK	83.0	-121.0
3529	12/01/2008 03:15:30 PM	COMPLETE	0-7340181	TRACK	10.0	10.0
3528	12/01/2008 02:24:23 PM	COMPLETE	0-7340184	911-CANCEL	0.0	0.0
3527	12/01/2008 01:00:06 PM	COMPLETE	0-7340182	911	10.0	-120.0
3526	12/01/2008 12:44:38 PM	FAILED	0-7340184	TRACK	8.0	-120.0

Data Push Testing Tool (shown above) provides an easy mechanism for customers to test the customer's route.

Note: It is also important to send a test message if the Data Push configuration is changed. To improve the performance SPOT uses memory caches and those will be refreshed quickly if a data push test is done.

To access Data Push Testing Tool, navigate to **SPOT API -> Data Push Test**. Here the customer can enter message type, messenger, latitude and longitude. The entered values are pushed to the Callback service as a test message. The status of the test (viz. Pending, Processing, Completed or Failed) can be seen in "Your Tests" section of this page. Any errors occurred during the test will be displayed in the Data Push Log of the Data Push Setup Page.

The following are the status of tests and their meaning:

Status	Description
PENDING	SPOT has received the test data and has scheduled it for the data push.
PROCESSING	SPOT has formatted the message and has started delivering the message.
FAILED	We failed to deliver the message. Go to the Data Push Setup Page and look at the Data Push Log for error messages.
COMPLETED	The message has been sent to the customer's service

All test messages generated through the test tool will have Message Sequence Number same as the Test ID (can be seen in the Test Log) and XML file be named T0003202.xml (where 3202 is the Test ID). If the Test ID is larger than 7 characters then the Message Sequence Number will be the last 7 digits of the Test ID. Similarly for HTTP/HTTPS transfer the routerMessageMode header value will be set as TEST for test data.

Recovery Requests

Recovery requests can span multiple days and each recovery request can contain many messages. **SPOT limits recovery requests to 10,000 messages per recovery**. If a request contains more than 10,000 messages then the request will be rejected automatically. Make sure that recovery requests are properly sized so that there are no more than 10,000 messages in a single recovery request. It is recommended that multiple shorter range recovery requests are made instead of a single big one.

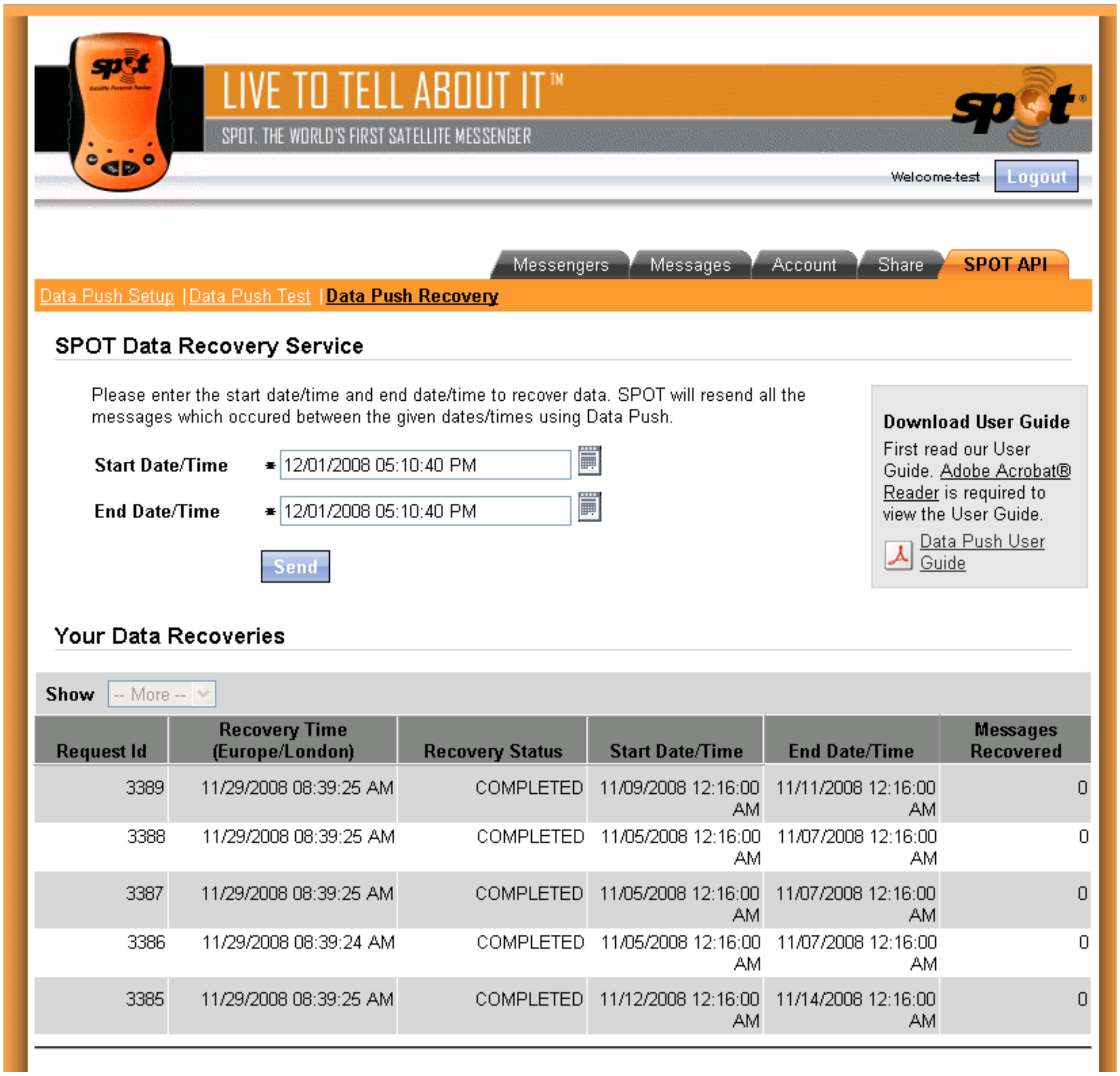
To access Data Push Recovery Tool, navigate to SPOT API -> Data Push Recovery. Here the customer can enter the start date and end date for the recovery. The messages between the start date and end date will be fetched and sent to the customer's service/server. The status of the recovery (viz. Pending, Processing, Sending, Completed, Partial or Failed) can be seen in "Your Data Recoveries" section of this page. As usual, any errors occurred during the recovery will be reported in the Data Push Log of the Data Push Setup Page.

The following are the status of recovery and their meaning:

Status	Description
PENDING	SPOT has received the test data and has scheduled it for the data push.
PROCESSING	SPOT has formatted the message and has started delivering the message.

SENDING	Some parts have been successfully transmitted – others are being transmitted currently.
PARTIAL	Some parts failed during transmission.
FAILED	We failed to deliver the message. Go to the Data Push Setup Page and look at the Data Push Log for error messages.
COMPLETED	The message has been sent to the customer's service

FIGURE 6 – DATA PUSH RECOVERY PAGE



SPOT Data Recovery Service

Please enter the start date/time and end date/time to recover data. SPOT will resend all the messages which occured between the given dates/times using Data Push.

Start Date/Time * 12/01/2008 05:10:40 PM

End Date/Time * 12/01/2008 05:10:40 PM

Download User Guide

First read our User Guide. Adobe Acrobat® Reader is required to view the User Guide.

Data Push User Guide

Your Data Recoveries

Request Id	Recovery Time (Europe/London)	Recovery Status	Start Date/Time	End Date/Time	Messages Recovered
3389	11/29/2008 08:39:25 AM	COMPLETED	11/09/2008 12:16:00 AM	11/11/2008 12:16:00 AM	0
3388	11/29/2008 08:39:25 AM	COMPLETED	11/05/2008 12:16:00 AM	11/07/2008 12:16:00 AM	0
3387	11/29/2008 08:39:25 AM	COMPLETED	11/05/2008 12:16:00 AM	11/07/2008 12:16:00 AM	0
3386	11/29/2008 08:39:24 AM	COMPLETED	11/05/2008 12:16:00 AM	11/07/2008 12:16:00 AM	0
3385	11/29/2008 08:39:25 AM	COMPLETED	11/12/2008 12:16:00 AM	11/14/2008 12:16:00 AM	0

Since each recovery request can contain as much as 10,000 messages, recovery requests are broken down and transmitted as multiple parts. Each part has the same XML structure but will contain only a maximum of 100 messages. Hence a recovery can be split into 100 transmissions and the parts are numbered 0 -99.

Recovery requests will have the sequence number as follows:

- Character 1 – 5: the last 5 digits of the Recovery Request ID
- Character 6 – 7: A sequence number from 00 – 99 indicating the part of the recovery transmission.

The name of recovery data file for SFTP/FTP transfer will be like R0304401.xml (Here 03044 is the last 5 digits of the Recovery ID and the 01 indicates that this is part 01 of the transmission). In case of HTTP/HTTPS transmission, the routerMessageMode header value will be set as RECOVERY for messages originated due to a recovery request.

Frequently Asked Questions

1. How much time will it take to complete the Data Push Setup?
   You can start testing using the Data Push Test T as soon the account is setup and the Customer side infrastructure (SFTP server or HTTP Listener) is ready. You have to login to your account and configure the Data Push Service by visiting the SPOT API tab.

2. What happens when the service at the SPOT customer's side (SFTP server or HTTP listener) is down?
   SPOT tries to resend data around 6 times within an hour if the customer's service is down. If the customer's service comes back up within that time then the data will be sent in the next attempt.

3. What is the SPOT recommended Data Push Service Protocols?
   SPOT recommends using HTTPS or SFTP.

4. We lost some data. How can this be resend?
   SPOT Data Push Recovery Tool can be used to recover the lost transmissions. Here the start date/time and end date/time is to be provided to recover data between the start and end date/time.

5. Can we get messages without delay?
   During nominal operations most data will be delivered within minutes and in most cases much faster. Periodically SPOT systems undergo planned maintenance and upgrades and this could impact message delivery. During planned and unplanned outages, 911 messages continue to be monitored by Globalstar 24x7 Control Center. SPOT B2B customers may also register with SPOT/Globalstar to receive downtime notifications

from the Control center. Contact information email or phone, should be provided when registering for SPOT B2B services. The SPOT Product team can provide additional information regarding notification procedures.

6. IS THERE ANY WAY TO FIGURE OUT WHETHER A FILE OR TRANSMISSION IS LOST?
   SPOT filenames will be of the format S9999999.xml where 9999999 is a zero padded sequence number. The sequence restarts at 1 when it reaches 9999999. Hence if you have received files S0001011.xml and S0001014.xml and nothing in between then it is an indication that the messages S0001012.xml and S0001013.xml were lost. Also remember that since SPOT retries for almost an hour, it is possible that an older message is delivered after a newer one. It is also important to verify whether there are any error messages in the Data Push Log. Refer Sec. “Data Push Log” for more information on Data Push Log.
   For HTTP/HTTPS transmissions, the header value routerMessageSeq can be used to find the transmission number.

7. WE FOUND THAT SPOT SENDS SOME HEADERS THAT ARE NOT DOCUMENTED. CAN WE USE THEM?
   The undocumented headers may not be supported in the future and can be removed. Hence it is recommended not to use undocumented headers. If a new element or header is needed then submit a request to SPOT product team.

8. HOW LONG WILL THE SPOT RETRY A FAILED TRANSFER?
   SPOT currently will retry for almost 24 hours before giving up.

Appendix

Appendix A - XML Format

The following is the sample XML format.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messageList xmlns="http://v2.shared.globalstar.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://share.findmespot.com/shared/schema/spo
tXml-v2.xsd">
  <header>
    <totalCount>3</totalCount>
    <mode>LIVE</mode>
    <errors>
      <error>
        <code>E-0001</code>
        <shortMessage>Short description of error</shortMessage>
        <longMessage>Longer message and full error description
        </longMessage>
      </error>
      <error>
        <!-- Multiple error messages possible.
        Snipped for Brevity -->
      </error>
    </errors>
  </header>
  <message>
    <id>43244556</id>
    <esn>0-7341007</esn>
    <esnName>Jono SPOT</esnName>
    <messageType>TEST</messageType>
    <messageDetail>This is a Test message. The device is powered
on and is working.</messageDetail>
    <timestamp>2008-05-20T10:52:55.000-07:00</timestamp>
    <timeInGMTSecond>1211305975</timeInGMTSecond>
    <latitude>37.425</latitude>
    <longitude>-121.8958</longitude>
    <nearestTown>Milpitas, CA, US</nearestTown>
    <nearestTownDistance>0 km(s)</nearestTownDistance>
  </message>
  <message>
    <!-- Snipped for Brevity -->
  </message>
  <message>
```

```
    <!-- Snipped for Brevity -->
  </message>
</messageList>
```

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://v2.shared.globalstar.com"
targetNamespace="http://v2.shared.globalstar.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="messageList">
  <xs:annotation>
   <xs:documentation>A list of messages get from the message
servlet</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="header" type="Header" minOccurs="0"/>
    <xs:element name="message" type="Message"
maxOccurs="unbounded" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

<xs:complexType name="Header">
   <xs:sequence>
    <xs:element name="totalCount" type="xs:integer">
     <xs:annotation>
      <xs:documentation>
       Total number of records in the query. Need not be the
       total number returned now if pagination is used.
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="mode" type="xs:string">
     <xs:annotation>
      <xs:documentation>
       A field that tells the mode of operation. TEST for data
       pushed as a part of a test operation or test request, LIVE
       for data from Production / Live system and RECOVERY for
       data pushed due to a data recovery request.
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="desc" type="xs:string">
     <xs:annotation>
      <xs:documentation>
```

```
       A description of transfer.
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="errors" type="Errors" minOccurs="0"/>
   </xs:sequence>
 </xs:complexType>

 <xs:complexType name="Errors">
   <xs:sequence>
     <xs:element name="error" type="Error" maxOccurs="unbounded"
minOccurs="0"/>
   </xs:sequence>
 </xs:complexType>

<xs:complexType name="Error">
   <xs:sequence>
    <xs:element name="code" type="xs:string">
     <xs:annotation>
      <xs:documentation>
       Error code if errors occurred during processing. Error
       codes start with E- on errors, W- on warnings and I- on
       informational messages.
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="shortMessage" type="xs:string">
     <xs:annotation>
      <xs:documentation>
       Brief Error message if errors occurred during processing.
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="longMessage" type="xs:string">
     <xs:annotation>
      <xs:documentation>
       Detailed Error message if errors occurred during
       processing.
      </xs:documentation>
     </xs:annotation>
    </xs:element>
   </xs:sequence>
 </xs:complexType>


 <xs:complexType name="Message">
   <xs:sequence>
    <xs:element name="id" type="xs:integer">
     <xs:annotation>
```

```
      <xs:documentation>
       This is the Id of the message
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="esn" type="xs:string">
     <xs:annotation>
      <xs:documentation>
       This is the DEVICE ESN field
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="esnName" type="xs:string" minOccurs="0"/>
    <xs:element name="messageType" type="MessageType">
     <xs:annotation>
      <xs:documentation>
       One of the defined types
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="messageDetail" type="xs:string"
minOccurs="0">
     <xs:annotation>
      <xs:documentation>
       Detailed Message in TEXT
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="timestamp" type="xs:dateTime">
     <xs:annotation>
      <xs:documentation>
       Time Message sent in GMT time
      </xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="timeInGMTSecond" type="xs:integer"
minOccurs="0"/>
    <xs:element name="latitude" type="xs:float" minOccurs="0"/>
    <xs:element name="longitude" type="xs:float" minOccurs="0"/>
    <xs:element name="nearestTown" type="xs:string"
minOccurs="0"/>
    <xs:element name="nearestTownDistance" type="xs:string"
minOccurs="0"/>
   </xs:sequence>
 </xs:complexType>
 <xs:simpleType name="MessageType">
   <xs:restriction base="xs:string">
     <xs:enumeration value="TEST"/>
     <xs:enumeration value="OK"/>
     <xs:enumeration value="TRACK"/>
     <xs:enumeration value="911"/>
     <xs:enumeration value="911-CANCEL"/>
     <xs:enumeration value="HELP"/>
     <xs:enumeration value="HELP-CANCEL"/>
     <xs:enumeration value="911 HELP"/>
     <xs:enumeration value="911-CANCEL HELP"/>
     <xs:enumeration value="911 HELP-CANCEL"/>
     <xs:enumeration value="911-CANCEL HELP-CANCEL"/>
   </xs:restriction>
</xs:simpleType>
</xs:schema>
```

Appendix B – Sample Java Code to verify X-WSSE headers

The following is a sample Servlet that verifies the X-WSSE headers.

```
package com.globalstar.router.test.servlet;

import java.io.BufferedReader;
import java.io.IOException;
import java.security.MessageDigest;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

/**
 * Servlet implementation class CallbackServlet
 */
public class CallbackServlet extends HttpServlet {

      public static final String CUSTOMER_TOKEN_KEY="UsernameToken 
Username";
      public static final String NONCE_KEY="Nonce";
      public static final String CREATED_KEY="Created";
      public static final String PASSWORD_DIGEST_KEY="PasswordDigest";

      private static final String MY_CUSTOMER_TOKEN="ofi2jj4k";
      private static final String MY_SECRET_TOKEN="232sadsd441324dff";
      private static final long NONCE_TIME_TO_LIVE= 2 * 60 * 1000; // 2
hours in millisec

      private static final long serialVersionUID = 1L;
      private Log log = LogFactory.getLog(this.getClass());


      private BASE64Decoder decoder = new BASE64Decoder();
      private BASE64Encoder encoder = new BASE64Encoder();
      private DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'");

      protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            response.sendError(401, "Unauthorized");
      }
      /**
       * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
       */
      @SuppressWarnings("unchecked")
      protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            if (isHeadersValid(request)) {
                  log.info("Valid WSSE Headers - request accepted");
            } else {
                  log.warn("Invalid WSSE Headers - request ignored");
                  response.setHeader("WWW-Authenticate", "WSSE
realm=\"SpotService\", profile=\"UsernameToken\"");
                  response.sendError(401, "Unauthorized");
                  return;
            }
            Enumeration headerNames = request.getHeaderNames();
            log.info("--------------- Headers ------------------");
            while(headerNames.hasMoreElements()) {
                  String header = (String) headerNames.nextElement();
                  Object value = request.getHeader(header);
                  log.info(header + " : " + value);
            }
            BufferedReader reader = new
BufferedReader(request.getReader());
            log.info("---------------  Body  ------------------");
            // Now get the body of the request - that contains XML
            String line = null;
            StringBuffer buffer = new StringBuffer();
            while ((line = reader.readLine()) != null) {
                  buffer.append(line);
            }
            log.info(buffer.toString());
      }

      private boolean isHeadersValid(HttpServletRequest request) throws
IOException {
            String wsseHeaders = request.getHeader("X-WSSE");
            String[] tokens = wsseHeaders.split("\", ");
            Map<String, String> keyValueMap = new HashMap<String,
String>();
            for (int i=0; i< tokens.length; i++ ) {
                  String token = tokens[i];
                  String[] keyValue = token.split("=\"");
                  keyValueMap.put(keyValue[0],
keyValue[1].replaceAll("\"", ""));
            }
            String customerToken = keyValueMap.get(CUSTOMER_TOKEN_KEY);
            if (!MY_CUSTOMER_TOKEN.equals(customerToken)) {
                  // This is not my request! Reject it.
                  log.error("Customer token did not match - token: " +
MY_CUSTOMER_TOKEN + " header: " + customerToken);
                  return false;
            }

            String nonceEncoded = keyValueMap.get(NONCE_KEY);
            /*
             * Here you should add code to check if the
             * nonceEncoded is in your nonce cache / storage.
             * If it exists in the nonce storage reject
             * this request.
             * if (securityService.nonceExists(nonceEncoded)) {
             *          return false;
             * }
             *
             */
            byte[] nonce = decoder.decodeBuffer(nonceEncoded);
            String created = keyValueMap.get(CREATED_KEY);
            try {
                  Date date = dateFormat.parse(created);
                  Date now = new Date();
                  if (now.getTime() - date.getTime() >
NONCE_TIME_TO_LIVE) {
                        log.error("Created too old: " + created);
                        return false;
                  }
            } catch (ParseException e) {
                  // log and ignore
                  log.warn("Ignored the error in parsing created date:
" + created);
            }
            log.info("nonceEncoded: " + nonceEncoded + " created: " +
created + " password: " + MY_SECRET_TOKEN);

            String passwordDigest = generatePasswordDigest(nonce,
created.getBytes("UTF-8"), MY_SECRET_TOKEN.getBytes("UTF-8"));
```

```
            String passwordDigestInHeader =
keyValueMap.get(PASSWORD_DIGEST_KEY);
            if (passwordDigest.equals(passwordDigestInHeader)) {
                  // the password digest we calculated and
                  // the one in the header is same.
                  // so this request is good - use it.
                  /*
                   * No insert code to save nonceEncoded
                   * in your nonce database.
                   * securityService.saveNonce(nonceEncoded);
                   */
                  return true;
            }
            log.error("Password Digest did not match - calculated: " +
passwordDigest + " header: " + passwordDigestInHeader);
            return false;

      }

      private synchronized String generatePasswordDigest(byte[] nonce,
byte[] created,
                  byte[] password) {
            try {
                  MessageDigest messageDigester =
MessageDigest.getInstance("SHA-1");
                  // SHA-1 ( nonce + created + password )
                  messageDigester.reset();
                  messageDigester.update(nonce);
                  messageDigester.update(created);
                  messageDigester.update(password);
                  return encoder.encode(messageDigester.digest());
            } catch (java.security.NoSuchAlgorithmException e) {
                  throw new RuntimeException(e);
            }
      }


}
```

Appendix C – Common error messages and reason

The following are the common error messages and the possible reasons for those errors.

Error Message	Description
HTTP operation failed with statusCode: 400, status: HTTP/1.1 400 Bad request	This mostly happens in older Application servers like Sun System Application server 8.2. This is due to the fact that the Application server does not support non-standard headers like WSSE. Please upgrade to Sun Application Server 9.1 or Glassfish 2.

HTTP operation failed with statusCode: 401, status: HTTP/1.1 401 Unauthorized	Are you properly doing the WSSE authentication? Are you suing the correct Customer Token and Secret Token? For some reason your application is rejecting the Data Push attempt. Please verify your server logs.
HTTP operation failed with statusCode: 404, status: HTTP/1.1 404 Not Found	The service URL provided is not correct or your service has been uninstalled. Contact your application administrator and verify the URL of your service. Also make sure that your service is running.
HTTP operation failed with statusCode: 500, status: HTTP/1.1 500 Internal Server Error	Your service is encountering an error and is unable to process our Data Push Transmission. Please verify your server logs and make sure that your service is running properly.
sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target	Your HTTPS service does not have proper certificates. Make sure that your server certificates are issued by a valid certification authority. SPOT Data Push does not support self-signed certificates.
Remote host closed connection during handshake	Same as above.
java.net.UnknownHostException: host.xyz	You have not provided the proper host name of your server.
Ftp operation failed: 553 /xxx/T003789.xml: Permission denied.	Make sure that the FTP directory has write and browse (execute) permission for the userID you provided in the Data Push configuration (Data Push Setup Page).
Connection refused	Is your service running in the correct port? Make sure that the firewall in your side is configured properly.