Iridium Burst^{sm} Service Developers Guide

Release 1.0
February 4, 2014

Revision History

Version	Date	Reason
1.0	4 February 2014	Initial Commercial Release

LEGAL INFORMATION, DISCLAIMER AND CONDITIONS OF USE

This Service Developers' Guide ("Guide") and all information for the Iridium Burst Service ("Product/Service") are provided "AS IS." The purpose of providing such information is to enable Value Added Resellers and Value Added Manufacturers (collectively, "Product Developer(s)") to understand the Product/Service and how to integrate it into a wireless solution. Reasonable effort has been made to make the information in this Guide reliable and consistent with specifications, test measurements and other information. However, Iridium Communications Inc. and its affiliated companies, directors, officers, employees, agents, trustees or consultants ("Iridium") assume no responsibility for any typographical, technical, content or other inaccuracies in this Guide. Iridium reserves the right in its sole discretion and without notice to you to change Product/Service specifications and materials and/or revise this Guide or withdraw it at any time. This Guide is a product provided in conjunction with the purchase of the Product/Service and is therefore subject to the Product Sales Terms and Conditions set forth at http://www.Iridium.com/support/library/Legal Notices.aspx. The Product Developer assumes any and all risks of using the Product/Service specifications and any other information provided in this Guide.

Your use of this Guide is restricted to the development activity authorized by your Partner Agreement with Iridium and is otherwise subject to all applicable terms and conditions of such Partner Agreement(s), including without limitation software license, warranty, conditions of use and confidentiality provisions. Please review your Partner Agreement and the Iridium Product Sales Terms and Conditions that govern your relationship with Iridium. This Guide is strictly Proprietary and Confidential to Iridium. Consistent with your Partner Agreement with Iridium, you may not disclose the Guide (or any portion thereof) to others without express prior written permission from Iridium. Any violation of your Partner Agreement's Proprietary and Confidentiality obligations shall result in remedies to the fullest extent available to Iridium at law or in equity.

IRIDIUM MAKES NO REPRESENTATIONS, GUARANTEES, CONDITIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, IMPLIED REPRESENTATIONS, GUARANTEES, CONDITIONS OR WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, SATISFACTORY QUALITY, NON-INTERFERENCE, ACCURACY OF INFORMATIONAL CONTENT, ARISING FROM OR RELATED TO A COURSE OF DEALING, LAW, USAGE, OR TRADE PRACTICE OR ARISING FROM OR RELATED TO THE PERFORMANCE OR NONPERFORMANCE OF ANY PRODUCTS AND/OR SERVICES, ACCESSORIES, FACILITIES OR SATELLITE SERVICES OR DOCUMENTATION EXCEPT AS EXPRESSLY STATED IN THE LIMITED WARRANTY. ANY OTHER STANDARDS OF PERFORMANCE, GUARANTEES, CONDITIONS AND WARRANTIES ARE HEREBY EXPRESSLY EXCLUDED AND DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. THIS DISCLAIMER AND EXCLUSION SHALL APPLY EVEN IF THE EXPRESS LIMITED WARRANTY AND DOCUMENTATION CONTAINED IN THIS GUIDE FAILS OF ITS ESSENTIAL PURPOSE.

IN NO EVENT SHALL IRIDIUM BE LIABLE, REGARDLESS OF LEGAL THEORY, INCLUDING WITHOUT LIMITATION CONTRACT, EXPRESS OR IMPLIED WARRANTY, STRICT LIABILITY, GROSS NEGLIGENCE OR NEGLIGENCE, FOR ANY DAMAGES IN EXCESS OF THE PRODUCT OR THE AMOUNT SET FORTH IN YOUR PARTNER AGREEMENT. NOR SHALL IRIDIUM BE LIABLE FOR INCLUDING ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES OF ANY KIND, LOSS OF REVENUE OR PROFITS, LOSS OF BUSINESS, LOSS OF PRIVACY, LOSS OF USE, LOSS OF TIME OR INCONVENIENCE, LOSS OF INFORMATION OR DATA, SOFTWARE OR APPLICATIONS OR OTHER FINANCIAL LOSS CAUSED BY THE PRODUCT/SERVICE (INCLUDING HARDWARE, SOFTWARE AND/OR FIRMWARE) AND/OR THE IRIDIUM SATELLITE SERVICES, OR ARISING OUT OF OR IN CONNECTION WITH THE ABILITY OR INABILITY TO USE THE PRODUCT/SERVICE (INCLUDING HARDWARE, SOFTWARE AND/OR FIRMWARE) AND/OR THE IRIDIUM SATELLITE SERVICES TO THE FULLEST EXTENT THESE DAMAGES MAY BE DISCLAIMED BY LAW AND WHETHER IRIDIUM WAS ADVISED OF THE POSSIBILITIES OF SUCH DAMAGES. IRIDIUM IS NOT LIABLE FOR ANY CLAIM MADE BY A THIRD PARTY OR MADE BY YOU FOR A THIRD PARTY.

Export Compliance Information

This Product/Service is controlled by the export laws and regulations of the United States of America. The U.S. Government may restrict the export or re-export of this Product/Service to certain individuals and/or destinations. Diversion contrary to U.S. law is prohibited. For further information, contact the U.S. Department of Commerce, Bureau of Industry and Security or visit www.bis.doc.gov.

Table of Contents

1.0 Introduction

1.1. Purpose

The purpose of this document is to provide technical and operational information sufficient for an Iridium Service Provider (SP) and its associated “content providers” to be able to develop an integrated data application that utilizes Iridium’s Burst Service. Additional information will be provided in other documentation related to the integration of the Iridium 9602GDB receiver and its associated AT Commands.

An overview of the Iridium satellite network is provided as well as descriptions of the terminal equipment and the end-to-end communications protocol for Burst. This document is intended for use by technical personnel and assumes a reasonable level of technical skill and familiarity with satellite and/or wireless data applications.

1.2. Scope

The scope of this document is the Burst system, how to send data, and major interfaces.

Additional documents are referenced which provide more specific detail on certain topics and these are listed in Section 1.3 of this document. This document does not specifically define the provisioning process, although it does reference it. This document assumes a working knowledge of the Iridium satellite system.

1.3. References

1.3.1. Specifically Referenced Documents

[1] MAN0009 ISU AT Command Reference
[2] 9602GDB Iridium Burst Receiver Developers Guide

These documents are accessible from the Iridium Web Support under Support: http://www.iridium.com.

1.4. Definitions, Acronyms, and Abbreviations

BCR	Broadcast Cancellation Request
BDR	Broadcast Delivery Request
BXR	Broadcast XML Request
DP	Data Provider
DSN	Delivery Status Notification
FA	Field Application
GDA	Global Delivery Areas
GGS	Gateway GDB Subsystem
MDN	Message Disposition Notification
MSR	Message Status Request
SBD	Short Burst Data
SMTP	Simple Mail Transfer Protocol
SP	Service Provider
TBD	To Be Determined
VA	Vendor Application
VAR	Value Added Reseller
VAM	Value Added Manufacturer
XML	Extensible Markup Language

2.0 Overview

2.1 Iridium Burst Network

Iridium's Burst network is a simple and efficient satellite network transport capability to broadcast data from a Data Provider (DP) to a centralized host computing system and on to Burst terminals.

The primary elements of the end to end Burst architecture are shown in Figure 2-1. Specifically, the elements consist of the Vendor Application (VA) and its peer Field Application (FA), the Gateway GDB Subsystem (GGS) located at the Iridium gateway, the Iridium satellite constellation, and the Burst terminals,

The Field Application represents the hardware and software that is configured by the Value Added Reseller for specific applications such as news and weather updates, sports scores, or traffic updates. The FA typically includes a Burst terminal and an attached device such as a laptop. The Iridium Gateway is responsible for receiving and formatting messages from the VA, also known as the DP, and sending them to the terminal (s) via the Iridium satellite constellation.

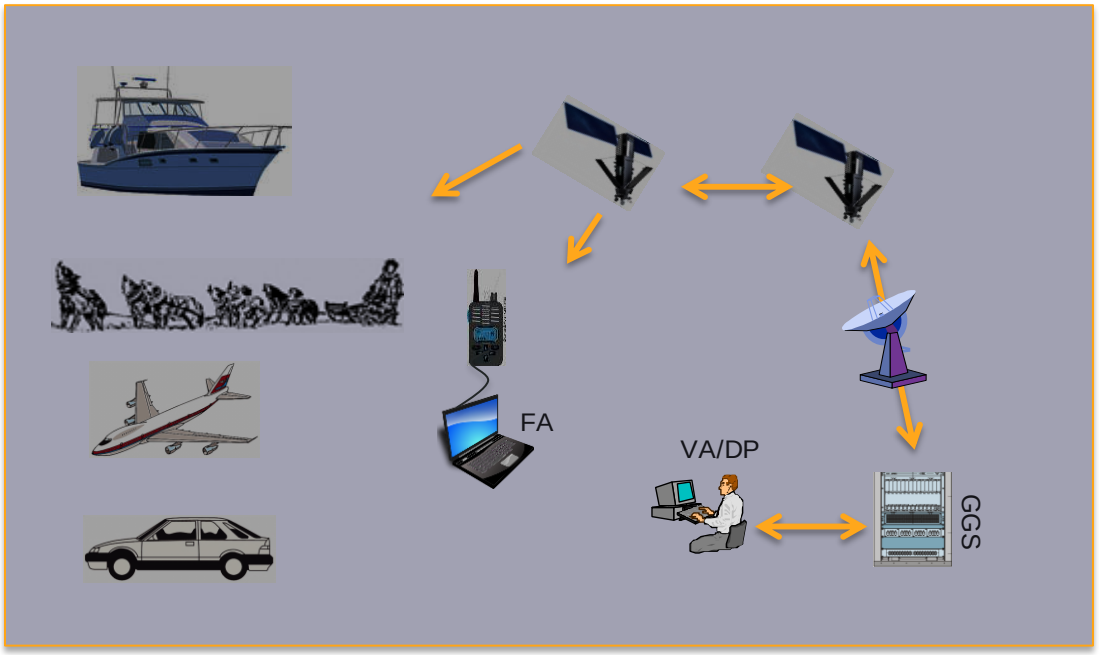

Figure 2-1 Iridium Burst Architecture

Data Providers may use two different interface types to communicate service delivery request messages to the Burst service at the Iridium Gateway. (1) Simple Mail Transfer Protocol (SMTP) and (2) web services.

Broadcasts destined for the FA are delivered to specific service addresses that are configured during provisioning using the Iridium SPNet provisioning system.

2.2 Terminal

The initial Burst Terminal will be a 9602GDB (shown in Figure 2-2 below) which is a 9602 Short Burst Data (SBD) Transceiver with a modified software load to support the receive-only requirements of the Burst system. Note that it will no longer function as an SDB device. Its dimensions are approximately 41mm X 45mm X 13mm. It will include the appropriate standard AT commands plus Burst unique AT commands.

For more detail on the terminal see the 9602GDB Developers Guide.

Figure 2-2 Iridium Burst Terminal

3.0 How Iridium Burst Works

3.1 Sending Data

The Iridium Burst service is a “data pipe.” No manipulation of content occurs. The service includes an option to add error detection/correction capability.

To send data to Burst terminals in the field, the data is submitted in an email to the GGS. For one form of the message, the contents of the email are delivery instructions and the actual data to be delivered. Alternatively, the email could have an XML attachment with delivery instructions and the actual data. Included in the message addresses of the email are the Service Address to which the data is to be sent and the location(s) to which the data is to be broadcast. Optionally, the date and time of transmission and whether the data should be sent more than once may be specified. The format of the delivery request messages is described in Section 4.

Using a secure connection, Burst Service receives the delivery request and validates its formatting. The Burst Service then schedules the data to be broadcast to the requested geographic area(s) and submits the data to the Iridium Network. The Iridium Network injects the data to the satellite network which routes the data to the appropriate satellite(s). The satellite(s) transmit the data on the appropriate beam(s) to cover the geographic area(s). Finally, the data is received by Burst terminals located within the geographic areas to which the data has been transmitted. Figure 3.1 below is a visualization of this process.

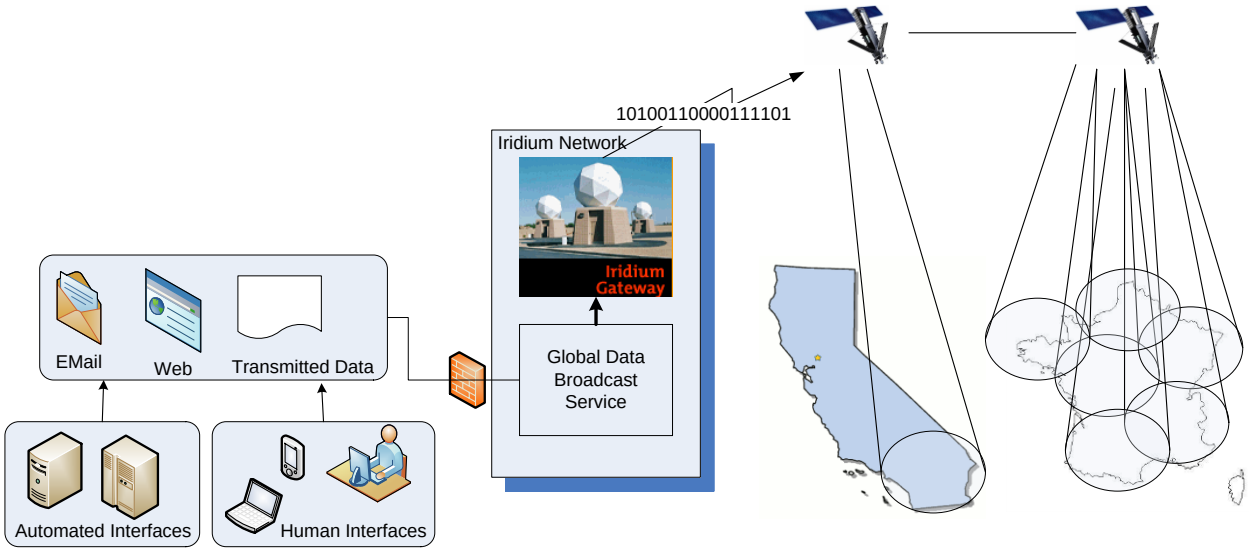

Figure 3.1 Broadcasting Data

3.2 Delivery Parameters-Geographic Areas

The delivery area for a Burst transmission may be identified by the sender in two ways.

3.2.1 Broadcast Coverage Areas

Iridium has pre-defined specific coverage areas, termed Broadcast Coverage Areas (BCAs) that encapsulate areas of coverage on the earth. These BCAs have several features:

- ***Large and Small:*** Areas on the globe are mapped for use in addressing messages to specific locations. Examples: France, Mexico, South America, North Pacific, Virginia
- ***Area Communication:*** The list of BCAs (available to authorized Burst VARs) includes a description of the number of Global Delivery Areas (GDAs) that are included in each BCA to allow the sender to understand the relative cost of sending messages to each area. The BCA Code and a small map representing the BCA are also available.
- ***Controlled by Iridium:*** Iridium will define the initial list and will administer any changes. Requests for new areas should be directed to Iridium.

Figure 3.2 below is a visualization of beams covering the France (BCA=”FR”) Broadcast Coverage Area.

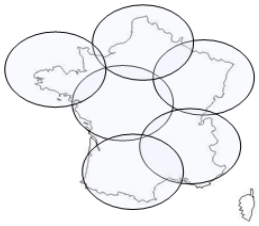

Figure 3.2 Geographic Name (BCA)

3.2.2 A Point on the Earth and Radius

Delivery Areas can be specified by the DP as a latitude, longitude, and radius (aka LLR). Note that the latitude and longitude are specified in decimal based degrees and the radius in km.

Figure 3.4 Point on the Earth

3.3 Receiving Data

Data broadcast to geographic areas is received by any device listening within that area. This could be one terminal or many terminals. See Figure 3.5 below. Each Burst terminal receives the messages broadcast to its geographic area and then decodes the message if the Service Addresses match the ones for which it is authorized. The terminal then determines if the Group Address to which the data was broadcast is one of the Groups for which it is provisioned. If so, it processes the data. If not, the data is discarded. If the Data Provider opts to scramble the data transmitted, the terminal will descramble the data using the security key provided during provisioning. The data is delivered in the format agreed to by the DP/VA and its partner solution application (FA) in the Attached Device. If there are multiple parts of the data, the Burst terminal waits for all of the pieces, reconstructs the segments and stores the completed file in memory. The partner solution application then pulls the available messages from memory into the application. Optionally, segments can be pushed to the Attached Device as they are received. New AT commands have been developed for managing interaction with the Burst terminal. It should be noted that the Burst service does not guarantee receipt by Group members. There are many reasons for non-receipt including system errors and blockage at the receiver. The expectation is that data will very seldom be missed due to system errors. The higher transmit power should overcome virtually all blockage situations. Note also that Services may utilize the sequence number assigned to the individual parts that make up the full content to determine if all of the parts have been received. Alternatively, the receiving application may embed sequence numbers within the data transmitted, to be aware of missed messages

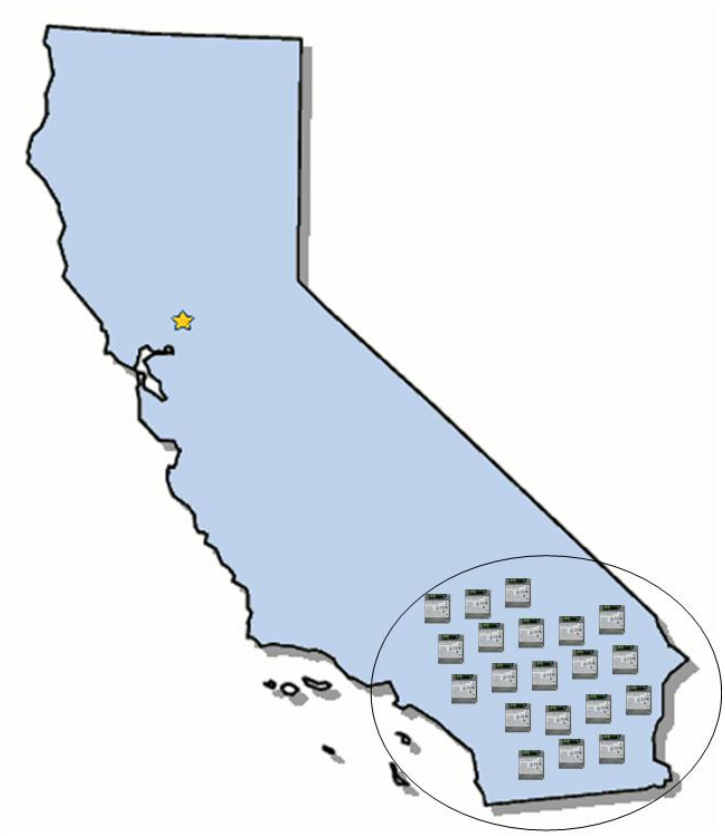

Figure 3.5 Receiving Data

4.0 Broadcast Requests

This section describes the message types sent from the DP to the Burst Service. See Appendix A for information on establishing the DP to Iridium Burst connection.

4.1 Broadcast Delivery Request

The fundamental purpose of the DP to Burst Service interface is to allow the DP to request broadcast transmission of a data item or stream. A Broadcast Delivery Request (BDR) message is defined for this purpose. Via the BDR message, a DP specifies which Service should receive the data, the coverage area in which the data should be broadcast, and the requested time at which the data should be broadcast. The BDR message is a basic format message. It uses standard email fields to carry Burst-specific information, as detailed in Appendix B.

4.2 Broadcast XML Request

The XML format allows a DP to embed Burst-specific delivery instructions in a structured message body that also includes the data in a separate file. A Broadcast XML Request (BXR) message is defined for this purpose. The BXR shares the same e-mail structure as BDR; however the details sent to the Burst Service are given through an XML attachment. Via the XML attachment, a DP specifies to the Burst Service a list of parameters which are: the Group Address (service name) to deliver the message, the coverage area in which the message should be broadcast, and the time at which the data should be broadcast. The BXR message is an XML Format message that is similar to the BDR message that uses an XML attachment in place of addressing for the Group Addresses, coverage areas, requested broadcast times, and options. Appendix C details the message format for the BXR message.

4.3 Web Services Broadcast Requests

The Web Services SOAP Server can process wBDRs, wBCRs, and wMSRs. These requests will be converted to an internal representation of BDRs, BCRs, and MSRs and will be treated the same by the GBTC. For a description of the elements in each message, see the field descriptions in Appendix G.

5.0 Other DP to Burst Service Messages

5.1 Broadcast Cancellation Request (BCR)

In support of the ability to specify a future transmission time in the BDR and BXR, the DP may request the Burst Service to cancel a request for which not all transmissions have occurred. Via the Broadcast Cancellation Request (BCR) message, a DP specifies to the Burst Service for which BDR or BXR the remaining transmissions should be canceled. The BCR message uses standard email fields to carry Burst-specific information, as detailed in Appendix D.

5.2 Message Status Request (MSR)

In support of the ability to specify a future transmission time in the BDR and BXR, the DP may request, from the Burst Service, the status for all transmissions of a message. Via the Message Status Request (MSR) message, a DP requests from the Burst Service the status of the transmissions for either a BDR or BXR. The MSR message uses standard email fields to carry Burst-specific information, as detailed in Appendix E.

6.0 Burst Service to DP Messages

In response to, or correlated with, a Broadcast Delivery Request, Broadcast Cancellation Request or Message Status Request message, the Burst Service will send to the DP's mailbox three types of messages. See Appendix F for a complete description of these messages.

6.1 Reflected BDR, BCR, or BXR

The first type of Burst Service to DP message is a Reflected Broadcast Request (RBR) which reflects the BDR, BCR, or BXR message requested by the DP due to inclusion of the DP address in the To: field of the original message. This type of message provides an indication to the DP that the message made it through the initial stages of processing at the Burst Service. It is a slightly more positive indication than the successful acceptance of the message at the MSP layer, and may serve as a message log if the DP requires one.

Since this message is an exact copy of the BDR, BXR, BCR, or MSR message in the relevant fields, the examples shown in the corresponding Appendix are applicable.

6.2 Message Disposition Notification

The second type of Burst Service to DP message is a Message Disposition Notification (MDN) as defined in RFC 3798. This type of message is sent if the DP requests it by including the "Disposition-Notification-To:" field in the header of a BDR, BCR, or BXR message. Note that common mail clients, like Thunderbird, for example, include this field when a "read-receipt" is requested. This message indicates that the Burst delivery has been completed, cancelled or expired.

6.3 Delivery Status Notification

The third type of Burst Service to DP Basic Format message is a Delivery Status Notification (DSN) as defined in RFC 3464. Two types of DSN messages can be generated for the DP: error and status message. Both are described in the sub-sections below.

6.3.1 Status Message

The DP may also request a DSN with the use of a Message Status Request (MSR) detailed in section 5.2. This message will contain a list of statuses for all transmissions specified for the BXR or BDR in the MSR. This message is only sent when the DP solicits it.

7.0 Field Applications

Data Services can provide virtually anything that can be digitized. The VA creates the binary version of the content and FA in the Attached Device presents it in the desired form. Data can be provided on a scheduled basis, an ad hoc basis, or streaming. The information can as small as a few bits or as large as hundreds or thousands of bits. Depending on the error checking/correcting options chosen, when data payloads are larger than 92 bytes (CRC only) or 73 bytes (CRC/FEC), they are broken down into multiple segments of the corresponding size.

All data is broadcast “Over The Air” and capable of being received by any device in the covered area. When protection of the data is desired, it can be scrambled using the Common Scrambling Algorithm. The de-scrambling key would be programmed during provisioning. When scrambled, the data is indecipherable without the key. See Section 8.2.

In challenging environments, the data can have error detection/correction features invoked and the data can be sent multiple times.

Below are some possible services:

- “Boston Sports Scores.” The service would provide up to the minute scores for the Bruins, the Red Sox, and the Patriots. Its primary delivery area would be the greater Boston area but a lonely Bostonian in Kuala Lumpur could receive the service as well if the DP includes the Kuala Lumpur in the targeted delivery area.

- National Hockey League Scores. This service would provide scores to all the cities with NHL Teams. Another lonely Bostonian in Hawaii could subscribe to the service as well.

- Current traffic information. The service would be for the greater metropolitan areas not only during rush hour but any time there was an event that could potentially impact drivers.

- Configuration commands. Configuration commands to remote operations equipment or personnel.

- Personalized messaging services. Services to individuals and groups in emergency and non-emergency situations

- Local Weather. Current and forecast weather, including severe weather warnings and watches to small and regional areas.

8.0 Security Features

Below are the basic security features associated with the Burst service.

8.1 Provisioning Security Key (PSK)

The PSK is loaded into the Burst device at the factory and is used for OTA provisioning of "Services". It is not visible to the subscriber nor can it be changed by the subscriber. Service additions and deletions are authenticated using this key.

8.2 Service Security Key (SSK)

The SSK is an optional key using the Common Scrambling Algorithm. When scrambled with an SSK the service data cannot be used unless it is unscrambled with the correct SSK. The SSK is provisioned along with the Service Name with either wired or OTA provisioning. It is not visible to the subscriber nor can it be changed by the subscriber.

9.0 Basic Trouble Shooting

9.1 Message Not Received

There are any number of reasons a broadcast message might not be received. The user of the Field Application can work with the SP/DP who will have records of when the message request was received, when the BDR/BXR was sent to the Gateway Service and when the Gateway Service reported the BDR/BXR was transmitted to the designated delivery area. The SP will also be able to interact with Iridium Customer Care for more in-depth trouble shooting if necessary.

Appendix A
Setting up the Connection/Interface from DP to Burst Service

Figure A.1 depicts the protocol stacks used on the DP to Burst Service interface. The individual elements of these stacks are detailed in the following sub-sections. The figure includes the network layers for easy reference to the following sub-sections.

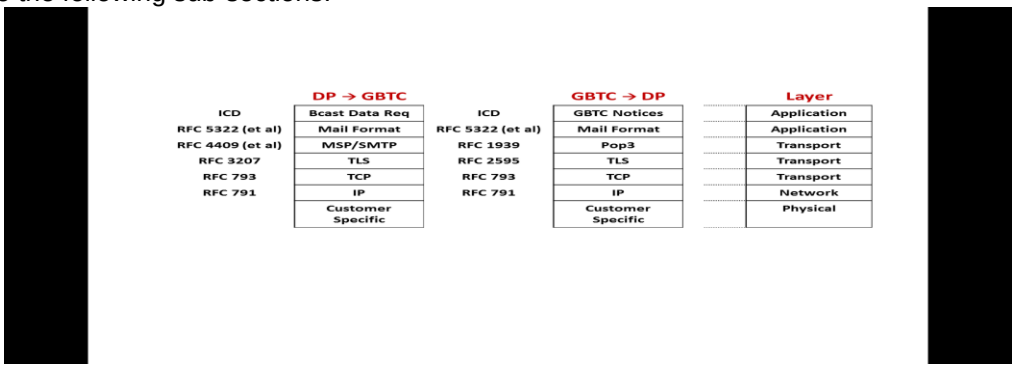

Figure A.1 DP - Burst Service Interface Protocol Stack with Layers

A.1 Physical Layer

The Data Provider and Burst Service interconnect using one or more physical network segments as appropriate for the specific configurations required by the customer and Iridium. Any physical network segment or combination of physical network segments that can carry the network layer specified in Section A.2 may be used.

One typical configuration would be for the DP and Burst Service both to use an ordinary connection to the Internet, with appropriate firewall and other common security mechanisms in place.

Another typical configuration would be for the customer and Iridium to establish a dedicated physical connection between a specific DP and the Burst Service. Common physical layer protocols for such private lines include Ethernet and T1. In any case appropriate firewall and other common security mechanisms would also be used.

The final network configuration is expected to include multiple DPs connected to the Burst Service at one or more Iridium Gateways. Therefore, these physical configurations and many others will likely be in use simultaneously in the deployed system.

A.2 Network Layer

The Data Provider and Burst Service interconnect using the Internet Protocol (IP). Depending on the specific version supported at the Iridium Gateway network access point the interface may use either IPv4 or IPv6.

Depending on customer and Iridium security requirements, this interface may optionally be configured to require Internet Protocol Security (IPSec).

Depending on customer and Iridium security requirements, this interface may optionally be configured to require a Virtual Private Network (VPN). In this configuration the VPN will be terminated on the Iridium side in an appropriate device in the Iridium Gateway network, not in the Burst Service.

Any standard combination of Network Layer feature set and Physical Layer configuration may be used in the deployed system, as required for each DP.

A.3 Transport Layer

Application messages are carried between the Data Provider and the Burst Service using IETF-standard messaging protocols. For the purpose of these protocols, the Data Provider acts as a mail client and the Burst Service acts as a mail server.

A.3.1 DP to Burst Service Direction

For messages from DP to the Burst Service, the Message Submission Protocol (MSP) defined in RFC 4409 is used. The DP acts as Mail User Agent (MUA) and the Burst Service acts as Message Submission Agent (MSA) as defined in that specification. MSP is a constrained configuration of the Simple Mail Transfer Protocol (SMTP) from RFC 5321, specifically designed for initial submission of a message by a client. This protocol requires authentication using SMTP-AUTH as defined in RFC 4954, and requires use of Transport Layer Security (TLS) using the STARTTLS option as defined in RFC 3207. For the DP to Burst Service interface, STARTTLS must always be used.

MSP provides for session establishment and authentication, and message transfer. In common use, a single message is submitted in a single session. However, MSP allows multiple messages to be submitted via multiple transfer segments within a single session thus the Burst Service will permit the DP to continue submitting messages during the same session.

A.3.2 Burst Service to DP Direction

For messages from the Burst Service to the DP, the Burst Service will provide a mailbox for the DP. The DP may access its mailbox using either the Post Office Protocol version 3 (POP3) defined in RFC 1939, or the Internet Message Access Protocol version 4 (IMAP4) defined in RFC 3501, both of which are designed for retrieval of messages. Authentication is required using the POP3 USER and PASS commands or the IMAP LOGIN command. Session encryption via TLS is required in order to protect the USER and PASS or LOGIN commands, using either the STLS command defined in RFC 2595 or the POP3S automatic TLS establishment mechanism commonly provided by mail servers for POP3, the STARTTLS command for IMAP, or the IMAPS automatic SSL establishment mechanism commonly provided by mail servers for IMAP.

A.3.3 Addressing

The port numbers used for the protocols described above, are listed here:

Protocol	Port
MSP	5587
IMAP	5993
HTTPS	8071

The Burst Service IP address will be available to the DP via a DNS lookup. The domain name to use is provided through the Data Provider provisioning process. For VPN-style network-layer deployments and dedicated physical-layer deployments, the Burst Service IP address may also be made available as an external datum that can be used in configuring the DP.

A.3.4 Credentials

The DP's username and password, for use in the authentication phase of the transport protocols, are provided through the Data Provider provisioning process.

Appendix B
Broadcast Delivery Request Detail

B.1 Message Header Format

The BDR message shall be formatted using standard message header fields as defined in RFC 5322 and RFC 2045, according to the specification in Table 1 below. All header fields not listed in the following table may be present in the message if they would be ordinarily be included by the mail client software used at the DP. They are effectively ignored by the Burst Service. That is, they will be preserved or processed according to ordinary/standard behaviors of the COTS mail server software incorporated in the Burst Service, but they will not affect any Burst-specific behavior. Other than presence of ignored header fields, messages that are not formatted as described will be rejected. If the error is detectable at the time of the MSP transaction, an error code will be returned; otherwise a separate Delivery Status Notification (DSN) message will be sent to the DP, if requested.

Field Name	Presence	Content Constraints
Date:	Ignored	This field is ignored by the GBTC
From:	Mandatory	Data Provider address as described below; only one address is permitted
Sender:	Prohibited	Message is rejected if this field is present, because it implies that there is more than one address in the From: field
To:	Mandatory	Recipient, area, time, and option addresses as below
Cc:	Optional	Treated as To:
Bcc:	Optional	Treated as To:
Message-ID:	Mandatory	Per standard
Subject:	Mandatory	Must contain "BDR" with no other characters
Content-Type:	Optional	As described below
Disposition-Notification-To:	Optional	Per RFC 3798; may contain only the DP address; controls whether the Burst Service will send a receipt message to the DP indicating completed/cancelled/etc. transmission of the BDR message

Table 1 Broadcast Delivery Request Message Header

B.2 Message Body Format

The BDR message shall be formatted using a message body as defined in RFC 5322, carrying the data to be broadcast. After any decoding steps, as defined below, the Burst Service will in turn interwork this data for transmission.

The message body contains MIME parts as defined in RFC 2045 and related specifications. The first part matching the specification in Table 2 will be decoded as an attachment and the resulting file will then be taken as the data to broadcast without further decoding. Any message not containing a MIME part as specified in Table 1, will be rejected by the Burst service. This rejection only applies to messages with the subject line as “BDR”. If the error is detectable at the time of the MSP transaction, an error code will be returned; otherwise a separate Delivery Status Notification (DSN) message will be sent to the DP, if requested.

Field Name	**Content Constraints**
Content-Type:	application/octet-stream;

	name="Broadcast Delivery Request.bin"
Content-Transfer-Encoding:	base64
Content-Disposition:	attachment; filename="Broadcast Delivery Request.bin"

Table 2 Broadcast Delivery Request Message Body - Attachment Option

B.3 Addressing

The Data Provider shall specify broadcast service, broadcast coverage area(s), requested broadcast time(s), and transmission options using To: addresses. Cc: and Bcc: addresses may also be used; they will be handled as if they were To: addresses. Multiple addresses are separated by commas, as specified in RFC 5322.

Each address contains a "local-part" and a "domain" connected by the "@" character, as specified in RFC 5322. In general, the "domain" will match the domain name used to discover the IP address of the Burst Service the "local-part" for any address will be as defined in the following subsections.

The specific addresses and addressing patterns identified in the sections that follow represent a snapshot of features that may be provided by the Burst Service. Additional features and corresponding addresses, as well as new address patterns, may be added at any time by upgrading the Burst Service and providing the corresponding information to Data Providers.

B.3.1 Data Provider Address

The form of a Data Provider address is "DP.[provider-name-string]@[domain]" where [provider-name-string] is an arbitrary designation selected or assigned during provisioning.

The DP learns its address through the Data Provider provisioning process.

The Data Provider Address is always used in the From: header field. It may be used in the To: header field if the DP prefers to receive a copy of the BDR message in response. It cannot be the only To: address specified.

B.3.2 Service Address

The form of a Service address is "SVC.[service-name-string]@[domain]" where [service-name-string] is an arbitrary designation selected or assigned during provisioning.

A Service address is used to name a particular set of data sent by a Data Provider, to receivers that are subscribed to the named Service. The Burst Service will treat the Service address as an alias, expanding it into the internal representation of one Broadcast Service Address assigned to represent the Service on the air interface. The Burst Service validates that the requesting DP is allowed to send data to the requested Service address, and rejects messages containing combinations that are not permitted.

The Service Address also includes a default coverage area.

The DP learns its Service addresses through the Data Provider provisioning process.

B.3.3 Coverage Area Addresses

Any number of Broadcast Coverage Area addresses may be provided in the BDR message. If no Broadcast Coverage Area address is provided in the BDR message, then the default coverage area for that Service will be used. In general, if permitted, specifying additional coverage areas will result in additional charges. In any case, the DP should request the minimum coverage area needed to accomplish its goals.

Broadcast Coverage Area addresses take multiple forms that may be extended through the life of the system. Forms listed here are suggestions, but other forms may be provided based on customer input. Service Providers and Data Providers may obtain a current list of supported Coverage Area addresses from Iridium through an operations-specific mechanism that is outside the scope of this specification.

CVG.global@[domain]
CVG.[continent-name]@[domain]
CVG.[ocean-name]@[domain]
CVG.[country-name]@[domain]
CVG.[region-name]@[domain]
CVG.[city-name]@[domain]
CVG.[lat-int].[lat-dec].[lon-int].[lon-dec].R.[radius-km] @[domain]
A [radius-km] value of 0 limits the transmission to a single beam.
Ex: CVG.38.8338889.-104.8208333.R.150@[domain]

The address that specifies the coverage area with geographic coordinates uses Latitude and Longitude (indicating an Earth surface point) and a Radius (indicating the area surrounding the surface point). The range of the Latitude is from -90 to 90 and the range of the Longitude is from -180 to 180. A negative number is used to indicate South of the Equator for Latitude and West of the Prime Meridian for Longitude and the negative sign should be included into the address.

The address that specifies the coverage area with Global Delivery Area (GDA) numbers uses a list of GDA numbers separated by a period. If this address is used then there must be at least one GDA number and at most 8 GDA numbers. If more than 8 GDA numbers are desired, then multiple addresses using GDA numbers will be required.

B.3.4 Time Addresses

The BDR message may specify a requested time at which to broadcast the data. If no Time address is provided, the Burst Service is requested to broadcast the data as soon as possible. If only one Time address is provided, the Burst Service is requested to broadcast the data at the requested time. If more than one Time address is provided, the Burst Service is requested to broadcast the data multiple times.

Time addresses take multiple forms that may be extended through the life of the system. Forms listed here are suggestions, but other forms may be provided based on customer input. Service Providers and Data Providers may obtain a current list of supported Time addresses from Iridium through an operations-specific mechanism that is outside the scope of this specification. The following list documents the Time formats along with representative examples.

TIME.now@[domain]
TIME.nowplus.[x].hours|minutes|seconds@[domain]
Ex: TIME.nowplus.45.seconds@[domain]
Ex: TIME.nowplus.15.minutes@[domain]
Ex: TIME.nowplus.2.hours@[domain]
TIME.exact.[UTC-time]@[domain]
Ex: TIME.exact.9.May.2011.12.34.56@[domain]

Note that the absolute time format ("time.[UTC-time]") takes the format specified in section 3.3 of RFC 5322, with the exception that white space ("FWS" in the specification) and the colon is replaced by the period (".") character, the "day-of-week" field is not permitted, and the "zone" is assumed to be +0000 and therefore is omitted. The requested may be no more than 192 hours in the future.

A repeat interval may be appended to any of the Time Formats to request multiple transmissions. The repeat count [n] specifies the number of additional transmissions after the first one, so the total number of requested transmissions would be count + 1. The repeat interval [x] specifies the number of hours, minutes, or seconds to elapse between transmissions. The repeat interval must appear immediately after the one of the Time Formats described earlier. Following is the format of the repeat addition with examples of all three Time formats.

TIME.[time-format].Repeat.[n].count.[x].hours|minutes|seconds@domain
Ex: TIME.now.Repeat.3.count.10.hours@[domain]
Ex: TIME.nowplus.15.minutes.Repeat.5.count.45.minutes@[domain]

Ex: TIME.exact.9.May.2011.18.30.00.Repeat.20.count.30.seconds
@[domain]

Note that if multiple transmissions are requested by the inclusion of multiple TIME addresses, a minimum interval of 30 seconds between requested transmission times is recommended; satisfaction of shorter intervals is at the discretion of the Burst Service based on current traffic load and constellation conditions.

An expiration value may be appended to any of the Time Formats to establish a time beyond which the message(s) should not be transmitted if it is still waiting to be transmitted or retransmitted if its initial transmission attempt fails for any reason. Two expiration formats may be used: a period of time (duration) or a specific time (exact). The duration [x] specifies the number of hours, minutes, or seconds to allow for any message retransmission attempts that may occur. The specific time [UTC-time] specifies an exact time that the message should be canceled. The format for the exact time is identical to the format for the Exact Time Format.

Once the elapsed time has passed, any remaining requested broadcasts will be cancelled and a Delivery Status Notification (DSN) message is sent to the DP indicating a timeout has occurred, if requested. The expiration value must appear immediately after the one of the Time Formats described earlier or the Repeat Format if one exists.

The following list demonstrates the format of the expiration addition with examples of the Time formats.

TIME.[time-format].Expires.[x].hours|minutes|seconds@[domain]
Ex: TIME.now.Expires.1.hours@[domain]
Ex: TIME.nowplus.15.minutes.Expires.30.minutes@[domain]
Ex: TIME.exact.9.May.2011.18.30.00.Expires.45.seconds@[domain]
TIME.[time-format].Expires.[UTC-time]@[domain]
Ex: TIME.now.Expires.9.May.2011.23.00.00@[domain]
Ex: TIME.nowplus.15.minutes.Expires.9.May.2011.23.00.00@[domain]
Ex: TIME.exact.9.May.2011.18.30.00.Expires.9.May.2011.23.00.00
@[domain]

The following list demonstrates the format of the repeat and expiration addition with examples of the Time Formats. Note that the Repeat Format must precede the Expiration Format.

TIME.[time-format].[repeat-format].[expiration-format]@[domain]
Ex: TIME.now.Repeat.3.count.2.hours.Expires.5.hours@[domain]
Ex: TIME.nowplus.15.minutes.Repeat.5.count.45.minutes.
Expires.30.hours@[domain]
Ex: TIME.exact.9.May.2011.18.30.00.Repeat.20.count.30.seconds.
Expires.9.May.2011.23.00.00@[domain]

The Burst Service validates that the requested time is within all cited limits and rejects any BDR message that specifies a non-conforming time.

B.3.5 Option Addresses

The BDR message may specify delivery options and other Burst Service features, as available from the Burst Service. Many Options are designed to enable a feature that is not provided by default, or to disable a feature that is provided by default. Other Options set message-handling attributes such as Priority. The name of the Option should be sufficiently descriptive as to be self-explanatory.

Option addresses generally take the form of keywords as shown in the examples below. Options listed here are preliminary indications of envisioned channel and Burst Service capabilities. Additional Options may be developed and added to the protocol as the system is developed, as well as through the life of the system,

based on customer input. Service Providers and Data Providers may obtain a current list of supported Options from Iridium through an operations-specific mechanism that is outside the scope of this specification.

To resend any segment for which a Message Delivery Order Reject (MDOR) is returned:
OPT.Retry_On_MDOR.[boolean]@[domain]
Ex: OPT.Retry_On_MDOR.True@[domain]
To send a message at a priority other than the Service's default priority:
OPT.Priority.[x]@[domain]
Ex: OPT.Priority.4@[domain]

B.4 Message Format Examples

Two examples of the basic BDR message format are provided here. The first example shows a message to a Service Name that implies a predefined set of receivers and coverage areas that is requested to be broadcasted at 18:30 on May 9th 2011. The second example shows a message to a number of specific services located within a specific region that is requested to be broadcasted three times in the next ten minutes, and for which disposition notifications are requested.

B.4.1 Simple Example

```
Message-ID: <4DC83201.2000308>
Date: Mon, 09 May 2011 12:17:13 -0600
From: DP.trial-data@burst.iridium.com
MIME-Version: 1.0
To: SVC.trial-service@burst.iridium.com,
 TIME.exact.09.May.2011.18.30.00@burst.iridium.com
Subject: BDR
Content-Type: multipart/mixed;
 boundary="------------000002080503040909050206"

This is a multi-part message in MIME format.
--------------000002080503040909050206
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

--------------000002080503040909050206
Content-Type: application/octet-stream;
 name="Broadcast Delivery Request.bin"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="Broadcast Delivery Request.bin"

SGVsbG8gV29ybGQhDQo=
--------------000002080503040909050206--
```

B.4.2 Complex Example

```
Message-ID: <4DC83472.3040004>
Disposition-Notification-To: DP.trial-data@burst.iridium.com
Date: Mon, 09 May 2011 12:37:38 -0600
From: DP.trial-data@burst.iridium.com
MIME-Version: 1.0
To: SVC.poobahs@burst.iridium.com,
 CVG.USCA@burst.iridium.com,
```

```
 CVG.USOR@burst.iridium.com,
 CVG.N.38.322.E.142.369.R.1500@burst.iridium.com,
 TIME.now@burst.iridium.com,
 TIME.nowplus.5.minutes@burst.iridium.com,
 TIME.nowplus.10.minutes@burst.iridium.com
Subject: BDR
Content-Type: multipart/mixed;
 boundary="------------000002080503040909050206"

This is a multi-part message in MIME format.
--------------000002080503040909050206
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

--------------000002080503040909050206
Content-Type: application/octet-stream;
 name="Broadcast Delivery Request.bin"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="Broadcast Delivery Request.bin"

SGVsbG8gV29ybGQhDQo=
--------------000002080503040909050206--
```

Appendix C
Broadcast XML Detail

C.1 Message Header Format

The BXR message shall be formatted using standard message header fields as defined in RFC 5322 and RFC 2045, according to the specification in Table C.1 below. All header fields not listed in the following table may be present in the message if and as ordinarily included by COTS mail client software used at the DP. They are effectively ignored by the Burst Service. That is, they will be preserved or processed according to ordinary/standard behaviors of the COTS mail server software incorporated in the Burst Service, but they will neither affect nor effect any Burst-specific behavior. Other than presence of ignored header fields, messages that are not formatted as described will be rejected. If the error is detectable at the time of the MSP transaction, an error code will be returned; otherwise a separate Delivery Status Notification (DSN) message will be sent to the DP, if requested.

Field Name	Presence	Content Constraints
Date:	Mandatory	Per standard
From:	Mandatory	Data Provider address as described below; only one address is permitted
Sender:	Prohibited	Message is rejected if this field is present, because it implies that there is more than one address in the From: field
To:	Mandatory	GBTC.XML@[domain] as described below.
Cc:	Optional	Treated as To:
Bcc:	Optional	Treated as To:
Message-ID:	Mandatory	Per standard
Subject:	Mandatory	Must contain "BXR" with no other characters
Content-Type:	Optional	As described below.
Disposition-Notification-To:	Optional	Per RFC 3798; may contain only the DP address; controls whether the Burst Service will send a receipt message to the DP indicating completion/cancellation/etc. of the BXR message.

Table C.1 Broadcast XML Request Message Header

C.2 Message Body Format

The BXR message shall be formatted using a message body as defined in RFC 5322, carrying the delivery instructions and the data to be broadcast. After any decoding steps, as defined below, the Burst Service will in turn interwork this data for transmission.

The message body contains MIME parts as defined in RFC 2045 and 3023. The first part matching the specification in Table C.2 will be taken as the XML data to interpret. The other part matching the specification in Table C.3 will be decoded as an attachment and the resulting file will then be taken as the data to broadcast without further decoding. Any message not containing a MIME part as specified in Table C.2 and Table C.3, will be rejected by the Burst Service. If the error is detectable at the time of the MSP transaction, an error code will be returned; otherwise a separate Delivery Status Notification (DSN) message will be sent to the DP, if requested.

Field Name	Content Constraints
Content-Type:	application/xml; name="Broadcast XML Request.xml"

Content-Transfer-Encoding:	base64
Content-Disposition:	attachment; filename="Broadcast XML Request.xml"

Table C.2 Broadcast XML Request Message Body – XML Attachment

Field Name	Content Constraints
Content-Type:	application/octet-stream; name="Broadcast Delivery Request.bin"
Content-Transfer-Encoding:	base64
Content-Disposition:	attachment; filename="Broadcast Delivery Request.bin"

Table C.3 Broadcast XML Request Message Body – BIN Attachment

C.3 Message Addressing

There exist two types of message addressing used for the BXR message: Data Provider and Service Addressing. Each address contains a "local-part" and a "domain" connected by the "@" character, as specified in RFC 5322. In general, the "domain" will match the domain name used to discover the IP address of the Burst Service as described in Section A.3.3 The "local-part" is for the Data Provider or Service Addressing.

The form of a Data Provider address is "DP.[provider-name-string]@[domain]" where [provider-name-string] is an arbitrary designation selected or assigned during provisioning. The DP learns its address through the Data Provider provisioning process. The Data Provider Address is always used in the From: header field. It may be used in the To: header field if the DP prefers to receive a copy of the BDR message in response. It cannot be the only To: address specified.

For the Service Address the Data Provider shall specify the Burst Service XML Handler address, and optionally the Data Provider's own address for message reflection, using the To: address. Cc: and Bcc: addresses may also be used; they will be handled as if they were To: addresses. Multiple addresses are separated by commas, as specified in RFC 5322. No other address form is permitted. The XML Handler address takes the form "GBTC.XML@[domain]".

C.4 Message Format Example

An example of the basic BXR message format is provided here.

```
Message-ID: <4DC83472.3040004>
Disposition-Notification-To: DP.trial-data@burst.iridium.com
Date: Mon, 09 May 2011 12:37:38 -0600
From: DP.trial-data@burst.iridium.com
MIME-Version: 1.0
To: GBTC.XML@burst.iridium.com
Subject: BXR
Content-Type: multipart/mixed;
 boundary="------------000002080503040909050206"

This is a multi-part message in MIME format.
--------------000002080503040909050206
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

--------------000002080503040909050206
```

```
Content-Type: application/xml;
 name="Broadcast XML Request.xml"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="Broadcast XML Request.xml"

--------------000002080503040909050206
Content-Type: application/octet-stream;
 name="Broadcast Delivery Request.bin"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="Broadcast Delivery Request.bin"

SGVsbG8gV29ybGQhDQo=
--------------000002080503040909050206—
```

C.5 XML Attachment Format

To supply the Burst Service parameter details for transmitting the binary a XML attachment is included in the BXR message. The XML attachment shall be formatted using the standard XML format as defined in RFC 3470. The XML attachment specifies to the Burst Service the data provider, service names, coverage areas, requested broadcast times, and broadcast options; each of which are separate elements that are encapsulated in the main BroadcastXMLRequest element. The following is an XML schema (XSD file) describing the XML attachment format:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!--Complex Type optiontype-->
  <xs:complexType name="optiontype">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="Option">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Key" maxOccurs="1" minOccurs="1">
                      <xs:simpleType>
                            <xs:restriction base="xs:string">
                                  <xs:enumeration value="priority"></xs:enumeration>
                                  <xs:enumeration value="retry_on_mdor"></xs:enumeration>
                            </xs:restriction>
                      </xs:simpleType>
                </xs:element>
                <xs:element name="Value" type="xs:string" maxOccurs="1" minOccurs="1"/>
              </xs:sequence>
            </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <!--Complex Type durationtype-->
  <xs:complexType name="durationtype">
    <xs:all minOccurs="1" maxOccurs="1">
      <xs:element name="Type">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="Hours" />
                <xs:enumeration value="Minutes" />
                <xs:enumeration value="Seconds" />
```

```
            </xs:restriction>
          </xs:simpleType>
      </xs:element>
      <xs:element name="Duration" type="xs:positiveInteger" />
    </xs:all>
  </xs:complexType>
  <!--Complex Type repeattype-->
  <xs:complexType name="repeattype">
    <xs:all minOccurs="1" maxOccurs="1">
      <xs:element name="IntervalDuration" type="durationtype" />
      <xs:element name="BroadcastCount" type="xs:positiveInteger" />
    </xs:all>
  </xs:complexType>
  <!--Complex Type expirationtype-->
  <xs:complexType name="expirationtype">
    <xs:choice>
      <xs:element name="Exact" type="xs:dateTime" />
      <xs:element name="IntervalDuration" type="durationtype" />
    </xs:choice>
  </xs:complexType>
  <!--Complex Type exacttype-->
  <xs:complexType name="exacttype">
    <xs:sequence>
      <xs:element name="DateTime" type="xs:dateTime" minOccurs="1" maxOccurs="1" />
      <xs:element name="Repeat" type="repeattype" minOccurs="0" maxOccurs="1" />
      <xs:element name="Expiration" type="expirationtype" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <!--Complex Type delaytype-->
  <xs:complexType name="delaytype">
    <xs:sequence>
      <xs:element name="Delay" type="durationtype" minOccurs="1" maxOccurs="1" />
      <xs:element name="Repeat" type="repeattype" minOccurs="0" maxOccurs="1" />
      <xs:element name="Expiration" type="expirationtype" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <!--Complex Type nowtype-->
  <xs:complexType name="nowtype">
    <xs:sequence>
      <xs:element name="Repeat" type="repeattype" minOccurs="0" maxOccurs="1" />
      <xs:element name="Expiration" type="expirationtype" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <!--Complex Type timetype-->
  <xs:complexType name="timetype">
    <xs:sequence>
      <xs:element name="TimeNow" type="nowtype" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="TimeDelayed" type="delaytype" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="TimeExact" type="exacttype" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--Complex Type geotype-->
  <xs:complexType name="geotype">
    <xs:all minOccurs="1" maxOccurs="1">
      <xs:element name="Latitude">
        <xs:simpleType>
```

```
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="-90" />
          <xs:maxInclusive value="90" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Longitude">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="-180" />
          <xs:maxInclusive value="180" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Radius" type="xs:nonNegativeInteger" /><!--must be in kilometers -->
  </xs:all>
 </xs:complexType>
 <!--Complex Type ldatype-->
 <xs:complexType name="ldatype">
  <xs:sequence minOccurs="1" maxOccurs="8">
    <xs:element name="LDANumber">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger">
                <xs:minInclusive value="1" />
                <xs:maxInclusive value="26631" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
   </xs:sequence>
 </xs:complexType>
 <!--Complex Type coveragetype-->
 <xs:complexType name="coveragetype">
  <xs:sequence>
    <xs:element name="CoverageAreaRegion" type="xs:string" minOccurs="0"
                                                          maxOccurs="unbounded"/>
    <xs:element name="CoverageAreaGeo" type="geotype" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="CoverageAreaLDAs" type="ldatype" minOccurs="0"
                                                          maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <!--Complex Type servicenametype-->
 <xs:complexType name="servicenametype">
  <xs:sequence minOccurs="1" maxOccurs="1">
    <xs:element name="ServiceName">
      <xs:simpleType>
        <xs:restriction base="xs:string">
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
   </xs:sequence>
 </xs:complexType>
 <!--Complex Type providertype-->
 <xs:complexType name="providertype">
  <xs:all minOccurs="1" maxOccurs="1">
    <xs:element name="ProviderName" type="xs:string" />
    <xs:element name="Domain" type="xs:string" />
```

```
    </xs:all>
  </xs:complexType>
  <!--Complex Type bmrtype-->
  <xs:complexType name="bmrtype">
    <xs:all minOccurs="1" maxOccurs="1">
          <xs:element name="DataProvider" type="providertype" />
          <xs:element name="ServiceAddress" type="servicenametype" />
          <xs:element name="CoverageAreas" type="coveragetype" />
          <xs:element name="Times" type="timetype" />
          <xs:element name="Options" type="optiontype" />
          <xs:element name="LatencyTest" maxOccurs="1" minOccurs="0"></xs:element>
    </xs:all>
  </xs:complexType>
  <!--Root Element-->
  <xs:element name="BroadcastXMLRequest" type="bmrtype" />
</xs:schema>
```

C.5.1 Data Provider Element

The Data Provider Element, named DataProvider, is a reflection of the Data Provider Address supplied in the BXR message. This element encapsulates two sub-elements named: ProviderName and Domain. The ProviderName is an arbitrary designation selected or assigned during provisioning and is the same as the [provider-name-string] in the Data Provider address. The Domain will match the domain name used to discover the IP address of the Burst Service and is the same as the [domain] in the Data Provider address.

C.5.2 Service Address Element

The Service Address Element, named ServiceAddress, encapsulates one sub-element named ServiceName that specifies the destination of the broadcast. The ServiceName is an arbitrary designation selected or assigned during provisioning.

C.5.3 Coverage Areas Element

The Coverage Areas Element, named CoverageAreas, is a list of coverage areas that specify areas to broadcast. The Coverage Areas Element encapsulates any number Coverage Area Elements. If no Coverage Area Elements exist, and the Service address does not imply a coverage area by provisioning, the Burst Service will reject the message. In any case, the DP should request the minimum coverage area needed to accomplish its goals.

A Coverage Area Element takes multiple forms that are specified by the name of the element. Four Coverage Area Element types exist: CoverageAreaGlobal, CoverageAreaRegion, and CoverageAreaGeo. Depending on the element, the sub-elements that are encapsulated within the Coverage Area Element vary. Table C.4 indicates the encapsulated elements that are required with every Coverage Area Element. Service Providers and Data Providers may obtain a current list of supported Coverage Area addresses from Iridium through an operations-specific mechanism that is outside the scope of this document.

Element Name	Sub-elements	Format
CoverageAreaGlobal	None	None
CoverageAreaRegion	None	String (encapsulated by Element)
CoverageAreaGeo	Latitude	Decimal Number (Range -90.00 to 90.00)
	Longitude	Decimal Number (Range -180.00 to 180.00)
	Radius	Non-negative Integer (0 limits the transmission to a single beam)

Table C.4 Coverage Area Element Types

The element that specifies the coverage area with geographic coordinates (CoverageAreaGeo) uses Latitude and Longitude (indicating an Earth surface point) and a Radius (indicating the area surrounding the surface point). The range of the Latitude is from -90 to 90 and the range of the Longitude is from -180 to 180. A negative number is used to indicate South of the Equator for Latitude and West of the Prime Meridian for Longitude and the negative sign should be included into the decimal number.

C.5.4 Times Element

The Times Element, named Times, is a list of times at which to request broadcasting of the message. The Times Element encapsulates any number of Time Elements. If no Time Elements exist, then the Burst Service is requested to broadcast the data as soon as possible. Otherwise, the Burst Service is requested to broadcast the data either as soon as possible after the requested time.

A Time Element takes multiple forms that are specified by the name of the element. Three Time Element types exist: TimeNow, TimeDelayed, and TimeExact. Depending on the element, the sub-elements that are encapsulated within the Time Element vary. Table C.5 indicates the encapsulated elements that are required with every Time Element. The TimeExact Element is constrained to be in the future by no more than 192 hours in the future in order to bound the Burst Service memory requirements. The Burst Service validates that the requested time is within all cited limits, and rejects any BXR message that specifies a non-conforming time.

Element Name	Sub-elements	Format
TimeNow	Repeat	Explained in Table C.6. (Optional)
	Expiration	Explained in Table C.7. (Optional)
TimeDelayed	Delay	2 sub-elements: String (Hours, Minutes, or Seconds) Positive Integer (Length of Delay)
	Repeat	Explained in Table C.6. (Optional)
	Expiration	Explained in Table C.7. (Optional)
TimeExact	DateTime	**YYYY**-**MM**-**DD**T**HH**:**MM**:**SS** (24 Hour) Ex: 2002-05-30T09:30:10
	Repeat	Explained in Table C.6. (Optional)
	Expiration	Explained in Table C.7. (Optional)

Table C.5 Requested Broadcast Time Element Types

Note that if multiple transmissions are requested by the inclusion of multiple Time Elements, a minimum interval of 30 seconds between requested transmission times is recommended; satisfaction of shorter intervals is at the discretion of the Burst Service based on current traffic load and constellation conditions.

The Time Elements have two sub-elements can be supplied to specify a repeat option named Repeat and an expiration option named Expiration. The Repeat sub-element is used to request multiple transmissions. The Repeat Element encapsulates two sub-elements named: BroadcastCount and IntervalDuration. The BroadcastCount Element specifies the number of additional transmissions after the first one, so the total number of requested transmissions would be n+1. The Interval Element specifies the duration between the transmissions. The Interval Element contains two sub-elements specifying the duration type (seconds, minutes or hours) and an integer for the duration (i.e. 30 seconds, 10 minutes, 3 hours, etc.) named Type and Duration respectively. Excluding the Repeat Element indicates that no repeat messages are desired.

Elements	Sub-elements	Format
IntervalDuration	Type	String (Hours, Minutes, or Seconds)
	Duration	Positive Integer
BroadcastCount	None	Positive Integer (encapsulated by Element)

Table indicates the encapsulated elements that are required for the Repeat Element.

Elements	Sub-elements	Format
IntervalDuration	Type	String (Hours, Minutes, or Seconds)
	Duration	Positive Integer
BroadcastCount	None	Positive Integer (encapsulated by Element)

Table C.6 Repeat Element Type

The other sub-element is the Expiration Element to request duration of time that message should be sent. The Expiration Element should contain only one of two sub-elements specifying a duration of time or an exact time of expiration. The Duration Element contains two sub-elements: a duration type (seconds, minutes or hours) and an integer for the duration (i.e. 30 seconds, 10 minutes, 3 hours, etc.) named Type and Duration respectively. The Exact Element contains the date and the time to expire the message. Excluding the Expiration Element indicates that no expiration is desired.

Once the elapsed time has passed, all remaining requested broadcasts will be canceled and a Delivery Status Notification (DSN) message is sent to the DP indicating a timeout has occurred. Table C.7 indicates the encapsulated elements that are required for the Expiration Element.

Elements	Sub-elements (one or the other)	Format
Expiration	Duration	2 sub-elements: String (Hours, Minutes, or Seconds) Positive Integer (Length of Duration)
	Exact	**YYYY**-**MM**-**DD**T**HH**:**MM**:**SS** (24 Hour) Ex: 2002-05-30T09:30:10

Table C.7 Expiration Element Type

C.5.5 Options Element

The Options Element, named Options, specifies delivery options and other Burst Service features, as available from the Burst Service. Many options are designed to enable features that are not provided by default, or to disable features that are provided by default. Other Options set message-handling attributes such as priority. The Options Element encapsulates any number Option Elements named Option. If no Option Element exists, then the Burst Service broadcasts with default options.

The Options Element encapsulates any number Option Elements named Option. The Option Element encapsulates two sub-element named: Key and Value. The Key Element specifies the option for the Burst Service to process and the Value Element specifies the value to pass. Both of the elements are string format and leave the Data Provider and Burst Service to process the Value Element appropriately. Table indicates the available Element Keys and the format of the Value Element represented as a string. Types listed here are suggestions, but other forms may be provided based on the customer input.

Elements	Format
Retry_On_MDOR	Boolean
Priority	Positive Integer

Table C.8 Available Options

C.5.6 XML Attachment Examples

Two examples of the XML attachment format are provided here. The first example shows a message to a Service Name that implies a predefined set of receivers and coverage areas that is requested to be broadcasted at 18:30 on May 9th 2011. The second example shows a message to a number of specific services located within a specific region, to be broadcast three times in the next ten minutes, and for which disposition notifications are requested. Note that these examples encapsulate the same information as the BDR messages.

C.5.6.1 Example 1:

```
<?xml version="1.0" encoding="utf-8"?>
<BroadcastXMLRequest>
  <DataProvider>
    <ProviderName>trial-data</ProviderName>
    <Domain>gbd.iridium.com</Domain>
  </DataProvider>
  <ServiceAddresses>
    <ServiceAddress>
      <ServiceName>trial-service</ServiceName>
    </ServiceAddress>
  </ServiceAddresses>
  <CoverageAreas />
  <Times>
    <TimeExact>
      <DateTime>2011-05-09T18:30:00</DateTime>
    </TimeExact>
  </Times>
  <Options />
</BroadcastXMLRequest>
```

C.5.6.2 Example 2:

```
<?xml version="1.0" encoding="utf-8"?>
<BroadcastXMLRequest>
  <DataProvider>
    <ProviderName>trial-data</ProviderName>
    <Domain>gbd.iridium.com</Domain>
  </DataProvider>
  <ServiceAddresses>
    <ServiceAddress>
      <ServiceName>poobahs</ServiceName>
    </ServiceAddress>

  </ServiceAddresses>
  <CoverageAreas>
    <CoverageAreaRegion>USCA</CoverageAreaRegion>
    <CoverageAreaRegion>USOR</CoverageAreaRegion>
    <CoverageAreaGeo>
      <Latitude>38.322</Latitude>
      <Longitude>142.369</Longitude>
      <Radius>1500</Radius>
    </CoverageAreaGeo>
  </CoverageAreas>
  <Times>
    <TimeNow />
```

```
    <TimeDelayed>
      <Delay>
        <Type>Minutes</Type>
        <Duration>5</Duration>
      </Delay>
    </TimeDelayed>
    <TimeDelayed>
      <Delay>
        <Type>Minutes</Type>
        <Duration>10</Duration>
      </Delay>
    </TimeDelayed>
  </Times>
  <Options />
</BroadcastXMLRequest>
```

Appendix D
Broadcast Cancellation Request Detail

D.1 Message Header Format

The BCR message shall be formatted using standard message header fields as defined in RFC 5322 and RFC 2045, according to the specification in the Table below. The “Subject:” header to be used and the presence of the “In-Reply-To:” header should be particularly noted here. Together these imply that the BCR may be easily constructed as a Reply email to the original BDR or BXR. All header fields not listed in the following table may be present in the message if and as ordinarily included by COTS mail client software used at the DP. They are effectively ignored by the Burst Service. That is, they will be preserved or processed according to ordinary/standard behaviors of the COTS mail server software incorporated in the Burst Service, but they will not affect any Burst-specific behavior. Other than presence of ignored header fields, messages that are not formatted as described will be rejected.

Field Name	Presence	Content Constraints
Date:	Mandatory	Per standard
From:	Mandatory	Data Provider address as described below; only one address is permitted
Sender:	Prohibited	Message is rejected if this field is present, because it implies that there is more than one address in the From: field
To:	Mandatory	GBTC.BCR@[domain] as described below
Cc:	Optional	Treated as To:
Bcc:	Optional	Treated as To:
Message-ID:	Mandatory	Per standard
In-Reply-To:	Mandatory	Contains the Message-ID value from the BDR or BXR to be cancelled
Subject:	Mandatory	Must contain “BCR” with no other characters

Table D.1 Broadcast Cancellation Request Message Header

D.2 Message Body Format

The BCR message shall be formatted using a message body as defined in RFC 5322. The actual content and structure of the message body are ignored by the Burst Service, and may be empty.

D.3 Addressing

The Data Provider shall specify a broadcast service group associated with the original BDR, or the BXR handler address associated with the original BXR, using the To: address. Cc: and Bcc: addresses may also be used; they will be handled as if they were To: addresses. Multiple addresses are separated by commas, as specified in RFC 5322. The BCR Handler address takes the form "GBTC.BCR@[domain]".

D.4 Message Format Example

An example of the BCR message format is provided here.

Message-ID: <5ED94312.3111419>
Date: Mon, 09 May 2011 17:12:43 -0600
From: DP.trial-data@burst.iridium.com
To: GBTC.BCR@burst.iridium.com
In-Reply-To: 4DC83201.2000308
Subject: BCR

Appendix E
Message Status Request Detail

E.1 Message Header Format

The MSR message shall be formatted using standard message header fields as defined in RFC 5322 and RFC 2045, according to the specification in the table below. The “Subject:” header to be used and the presence of the “In-Reply-To:” header should be particularly noted here. Together these imply that the MSR may be easily constructed as a Reply email to the original BDR or BXR. All header fields not listed in the following table may be present in the message if and as ordinarily included by COTS mail client software used at the DP. They are effectively ignored by the Burst Service. That is, they will be preserved or processed according to ordinary/standard behaviors of the COTS mail server software incorporated in the Burst Service, but they will neither affect nor effect any Burst-specific behavior. Other than presence of ignored header fields, messages that are not formatted as described will be rejected

Field Name	Presence	Content Constraints
Date:	Mandatory	Per standard
From:	Mandatory	Data Provider address as described below; only one address is permitted
Sender:	Prohibited	Message is rejected if this field is present, because it implies that there is more than one address in the From: field
To:	Mandatory	GBTC.MSR@[domain] as described below
Cc:	Optional	Treated as To:
Bcc:	Optional	Treated as To:
Message-ID:	Mandatory	Per standard
In-Reply-To:	Mandatory	Contains the Message-ID value from the BDR or BXR to request a status
Subject:	Mandatory	Must contain “MSR” with no other characters

Table E.1 Message Status Request Message Header

E.2 Message Body Format

The MSR message shall be formatted using a message body as defined in RFC 5322. The actual content and structure of the message body are ignored by the Burst Service, and may be empty.

E.3 Addressing

The Data Provider shall specify a broadcast service group associated with the original BDR, or the BXR handler address associated with the original BXR, using the To: address. Cc: and Bcc: addresses may also be used; they will be handled as if they were To: addresses. Multiple addresses are separated by commas, as specified in RFC 5322. The XML Handler address takes the form "GBTC.MSR@[domain]".

E.4 Message Format Example

An example of the MSR message format is provided here.

Message-ID: <5ED94312.3111419>
Date: Mon, 09 May 2011 17:12:43 -0600
From: DP.trial-data@burst.iridium.com
To: GBTC.MSR@burst.iridium.com
In-Reply-To: 4DC83201.2000308

Subject: MSR

Appendix F
Burst Service to DP Messages

F.1 Reflected BDR, BCR, or BXR

The first type of Burst Service to DP message is a Reflected Broadcast Request (RBR) which reflects the BDR, BCR, or BXR message requested by the DP due to inclusion of the DP address in the To: field of the original message. This type of message provides an indication to the DP that the message made it through the initial stages of processing at the Burst Service. It is a slightly more positive indication than the successful acceptance of the message at the MSP layer, and may serve as a message log if the DP requires one.

Since this message is an exact copy of the BDR, BXR, BCR, or MSR message in the relevant fields, the examples shown in previous sections are applicable.

F.2 Message Disposition Notification

The second type of Burst Service to DP message is a Message Disposition Notification (MDN) as defined in RFC 3798. This type of message is sent if the DP requests it by including the "Disposition-Notification-To:" field in the header of a BDR, BCR, or BXR message. This message indicates that the Burst Service request has been completed, expired, or cancelled..

An example of this message as it would be provided for the example BDR or the example follows.

```
Date: Mon, 09 May 2011 18:48:12 -0000
From: gbtc@burst.iridium.com
Message-ID: <4DC83A40.3090604>
Subject: MDN (broadcast)
To: DP.trial-data@burst.iridium.com
References: <4DC83472.3040004>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=disposition-notification;
        boundary="----------mdn020902010903000909040600"

----------mdn020902010903000909040600
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit

This message acknowledges that the data was sent for broadcast via the Iridium constellation and
not rejected by it. There is no guarantee that any receiver has actually received or processed the
data.

----------mdn020902010903000909040600
Content-Type: message/disposition-notification; name="MDNPart2.txt"
Content-Disposition: inline
Content-Transfer-Encoding: 7bit

Final-Recipient: rfc822;DP.trial-data@burst.iridium.com
Original-Message-ID: <4DC83472.3040004>
Disposition: automatic-action/MDN-sent-automatically; displayed

----------mdn020902010903000909040600--
```

An example of this message as it would be provided for the example BCR follows.

Date: Mon, 09 May 2011 17:12:58 -0000
From: gbtc@burst.iridium.com
Message-ID: <4DC83A40.3090604>
Subject: MDN (broadcast)
To: DP.trial-data@burst.iridium.com
References: <5ED94312.3111419>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=disposition-notification;
boundary="----------mdn020902010903000909040600"

----------mdn020902010903000909040600
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit

This message acknowledges a Cancellation Request identified in the References: header of this message, linked to an original Broadcast Delivery Request or Broadcast XML Request identified in the Original-Message-ID field of the attached Message Disposition Notification. All Burst transmissions associated with this original BDR or BXR, that have not already been sent to the Space Vehicles, are cancelled and will not be sent.

----------mdn020902010903000909040600
Content-Type: message/disposition-notification; name="MDNPart2.txt"
Content-Disposition: inline
Content-Transfer-Encoding: 7bit

Final-Recipient: rfc822;DP.trial-data@burst.iridium.com
Original-Message-ID: <4DC83201.2000308>
Disposition: automatic-action/MDN-sent-automatically; displayed

----------mdn020902010903000909040600--

F.3 Delivery Status Notification

The third type of Burst Service to DP Basic Format message is a Delivery Status Notification (DSN) as defined in RFC 3464. Three types of DSN messages can be generated for the DP: error, status, and cancellation messages. These are described in the sub-sections below.

F.3.1 Error Message

The first type of DSN message is used to report errors encountered for any of the messages from the DP. This type of message can be sent, without being solicited, if for any reason the data could not be broadcast after having been accepted by the GBTC. No optional per-message fields and no per-recipient fields will be included in the report.

An example of a DSN error message follows.

This message acknowledges the message request identified by the Email ID below has completed processing. There is no guarantee that any receiver has actually received or processed the data. Further information about this transmission:

Priority: 1

Coverage Areas: '[["Arizona"]]'

Disposition Requested: True

Email ID: '<20130221184709.26539.58480@localhost.localdomain>'

Expires at: '2013-02-21T19:47:09.462234'

Submitted Payload Size: 193 bytes

Total Transmission Size (including headers and any encoding): 283 bytes

Queue at: '2013-02-21T18:47:09.462234'

Service Name: 'WEATHER'

Status: Partially Sent (MDORs > Number of retries allowed)

Submitted at: '2013-02-21T18:47:09.508621'

Percent Sent: 90

F.3.2 Status Message

The DP may also request a DSN with the use of a Message Status Request (MSR) detailed in section **Error! Reference source not found.**. This message is only sent when the DP solicits it.

An example of a DSN status message follows.

This message contains status information for the message request identified by the Email ID below.

Priority: 1

Coverage Areas: '[[AZ]]'

Disposition Requested: False

Email ID: '<512e8e3e.UjOF8keVtife52U0%DP.MrWeather@burst.iridium.com>'

Submitted Payload Size: 193 bytes

Total Transmission Size (including headers and any encoding): 283 bytes

Queue at: '2013-02-27T22:52:46.850877'

Service Name: 'WEATHER'

Status: Scheduled

Submitted at: '2013-02-27T22:52:46.916042'

Percent Sent: 20

F.3.3 Cancellation Message

The DP will also receive a DSN on completion of a cancellation request via a BCR.

An example of a DSN cancellation message follows.

This message acknowledges a Cancellation Request was processed for the message request identified by the Email ID below. All GDB transmissions associated with the original request that have not already been sent to the Space Vehicles are cancelled and will not be sent.

Priority: 1

Coverage Areas: '[[[0.0,0.0,99999]]]'

Disposition Requested: False

Email ID: '<512e395d.xyBpyTt7SjjWes48%DP.MrWeather@ burst.iridium.com>'

Submitted Payload Size: 9092 bytes

Total Transmission Size (including headers and any encoding): 11458 bytes

Queue at: '2013-02-27T16:50:37.749767'

Service Name: 'WEATHER'

Status: Cancelled (partially transmitted)

Submitted at: '2013-02-27T16:50:38.085624'

Percent Sent: 40

Appendix G
Web Services

The Web Services SOAP Server can process wBDRs, wBCRs, and wMSRs. These requests will be converted to an internal representation of BDRs, BCRs, and MSRs and will be treated the same by the GBTC. The formats of the SOAP requests and responses are described in the following sections. For a description of the elements in each message, see the field descriptions in the corresponding Appendix above.

G.1 Connecting to the SOAP Server

To connect, via Hypertext Transfer Protocol Secure (HTTPS), to the Burst Web Services SOAP Server, a SOAP client will need to use the WSDL file shown in Appendix H. The python SOAP client library suds (https://fedorahosted.org/suds/) is used in this example.

G.2 Sending a wBDR to the SOAP Server

The information needed to send a wBDR to the SOAP Server is nearly the same as in standard BDRs. One difference is that there is no parameter for reflecting a request; the data in the request will be automatically reflected as a response to the SOAP request. Also, with wBDRs, the payload is included directly in the request, not in an attached file.

G.2.1 Request

The elements of the send_bdr request, used to send a wBDR, from the WSDL in Appendix H are as follows:

Element	Required	Description	Type
dp_name	Yes	The DP's name	String
dp_domain	Yes	The DP's domain	String
dp_password	Yes	The DP's password	String
email_id	Yes	The unique identifier of the broadcast request	String
service_name	Yes	The Service the broadcast request is for	String
coverage_area_region_list	No	The Broadcast Coverage Area(s) the broadcast request is for	String_Array
coverage_area_geo_list	No	The lat/lon/radius area(s) the broadcast request is for	CoverageAreaGeo_Array
coverage_area_lda_list	No	The LDA(s) the broadcast request is for	Int_Array
time_now	No	Used if the broadcast should be sent immediately	TimeNow
time_delayed_list	No	Used if the broadcast should be delayed for a certain period of time	TimeDelayed_Array
time_exact_list	No	Used if the broadcast should be set for an exact time in the future	TimeExact_Array
options	No	Used to set delivery options for the broadcast	Options
payload	Yes	The data to be sent in the request	base64Binary
Disposition_requested	No	Whether or not the DP wants an MDN sent when the request is processed	Boolean

Note that the email_id parameter is not a true email identifier since this request does not go through the Email Server, but rather a message identifier. The name of the parameter has been kept the same for consistency and will still be used to uniquely identify messages in the case of a wBCR or wMSR.

The email_id values used here and in emailed BDRs/BXRs ***must*** be unique for each message, and across both mail and web interfaces if both are used.

Also note that the payload may be left empty.

The elements of the complex data types used in the request are listed below. Complex data types ending in "_Array" are arrays of the types described.

Element	Required	Description	Type
lat	Yes	The latitude of the center of the requested delivery area	Float
lon	Yes	The longitude of the center of the requested delivery area	Float
radius	Yes	The radius of the requested delivery area	Int

Table G-1. CoverageAreaGeo Elements

Element	Required	Description	Type
use_time_now	Yes	Whether or not the request should be queued upon request	Boolean
expiration	No	The expiration parameters of the request (see Table G.6)	Expiration
repeat	No	The repeat parameters of the request (see Table G.5)	Repeat

Table G-2. TimeNow Elements

Element	Required	Description	Type
delay	Yes	Numeric length of requested delay	Int
delay_type	Yes	String description of requested delay (Hours, Minutes, Seconds)	String
expiration	No	The expiration parameters of the request (see Table G.6)	Expiration
repeat	No	The repeat parameters of the request (see Table G.5)	Repeat

Table G-3. TimeDelayed Elements

Element	Required	Description	Type
datetime_string	Yes	The requested time for the request to be sent. Format: **YYYY**-**MM**-**DD**T**HH**:**MM**:**SS** (24 Hour). Ex: 2002-05-30T09:30:10	String
expiration	No	The expiration parameters of the request (see Table G.6)	Expiration
repeat	No	The repeat parameters of the request (see Table G.5)	Repeat

Table G-4. TimeExact Elements

Element	Required	Description	Type
broadcast_count	Yes	The number of time the message should be repeated	Int
duration	Yes	Numeric value of the duration between transmissions	Int
duration_type	Yes	String description of the duration (Hours, Minutes, Seconds)	String

Table G-5. Repeat Elements

Element (Either exact OR duration/type)	Required	Description	Type
exact	No	The exact time the message should expire. Format: **YYYY**-**MM**-**DD**T**HH**:**MM**:**SS** (24 Hour). Ex: 2002-05-30T09:30:10	String
duration	No	Numeric value of the duration between transmissions	Int
duration_type	No	String description of the duration (Hours, Minutes, Seconds)	String

Table G-6. Expiration Elements

Element	Required	Description	Type
retry_on_mdor	No	Whether or not a segment of a message should be retried in the event of an MDOR being received	Boolean
priority	No	The requested priority of the message	Int

Table G-7. Options Elements

NOTE: A message requested with a better priority than the service allows will be adjusted to be sent at the service's best priority.

The service's provisioned priority is the best allowed priority for the service (the smaller the number for the priority, the better the priority - i.e. Priority of 10 is better than Priority of 20). Messages can be sent at a worse priority on a per message basis. If a message is sent at a worse priority, the service's priority boost and priority time parameters can be used to help prevent a message from being starved by better priority messages. The amount of gain a message can receive can never exceed the service's provisioned priority. Aging will raise the message priority from the requested priority to the provisioned priority adjusted by the max boost amount over an interval of max priority time. The message will not get better in priority than this, even if it has not reached the Service's provisioned priority.

G.2.2 Response

The response to a send_bdr call is the reflection of the parameters, contained in the following BDRReflected complex type. Also in the response is a boolean success value and a status field that is populated with an error description if the success value is False.

Element	Description	Type
success	Whether the SOAP request succeeded or not	Boolean
status	A description of any errors or warnings.	String

	NOTE: A warning, with success still being True, would be issued if a message was sent with an adjusted priority due to an invalid priority in the request.	
dp_name	The DP's name	String
dp_domain	The DP's domain	String
email_id	The unique identifier of the broadcast request	String
service_name	The Service the broadcast request is for	String
coverage_area_region_list	The Broadcast Coverage Areas the broadcast request is for	String_Array
coverage_area_geo_list	The lat/long/radius areas the broadcast request is for	CoverageAreaGeo_ Array
coverage_area_lda_list	The LDAs the broadcast request is for	Int_Array
time_now	Used if the broadcast should be sent immediately	TimeNow
time_delayed_list	Used if the broadcast should be delayed for a certain period of time	TimeDelayed_Array
time_exact_list	Used if the broadcast should be set for an exact time in the future	TimeExact_Array
options	Used to set delivery options for the broadcast	Options
payload	The data to be sent in the request	base64Binary
disposition_requested	Whether or not the DP wants an MDN be sent when the request is processed	Boolean

Table G-8. BDRReflected Elements

G.3 Sending a wBCR to the SOAP Server

G.3.1 Request

The elements of the send_bcr request, used to send a wBCR, from the WSDL in Appendix H are as follows:

Element	Required	Description	Type
dp_name	Yes	The DP's name	String
dp_domain	Yes	The DP's domain	String
dp_password	Yes	The DP's password	String
email_id	Yes	The unique identifier of the broadcast request	String

Table G-9. send_bcr Elements

G.3.2 Response

The response to a send_bcr call is the reflection of the parameters, contained in the following BCRReflected complex type. Also in the response is a boolean success value and a status field that is populated with an error description if the success value is False.

Element	Description	Type
success	Whether the SOAP request succeeded or not	Boolean
status	A description of any errors if success is False	String
dp_name	The DP's name	String
dp_domain	The DP's domain	String
email_id	The unique identifier of the broadcast request	String

Table G-10. BCRReflected Elements

G.4 Sending an MSR to the SOAP Server

G.4.1 Request

The elements of the send_msr request, used to send a wMSR, from the WSDL in Appendix H are as follows:

Element	Required	Description	Type
dp_name	Yes	The DP's name	String
dp_domain	Yes	The DP's domain	String
dp_password	Yes	The DP's password	String
email_id	Yes	The unique identifier of the broadcast request	String

Table G-11. send_msr Elements

G.4.2 Response

The response to a send_msr call is the following MessageStatusResponse complex type.

Element	Description	Type
success	Whether the SOAP request succeeded or not	Boolean
status	A description of any errors if success is False	String
dp_name	The DP's name	String
dp_domain	The DP's domain	String
email_id	The unique identifier of the broadcast request	String
message_status_list	The status of each message in the request	MessageStatus_Array

Table G-12. MSRReflected Elements

The elements of each MessageStatus type in the MessageStatus_Array are as follows.

Element	Description	Type
priority	The requested priority of the message	Int
coverage_area_list	The list of all CVG requests - including each coverage_area_region, coverage_area_geo, and coverage_area_lda - for the message	String
disposition_requested	Whether or not the DP requested an MDN	Boolean
expires_at	When the message expires, if it does	String
payload_size	The size of the provided payload (in bytes)	Int
total_transmission_size	The total size, including headers and encoding, of the message (in bytes)	Int
queue_at	When the message is queued for transmission	String
service_name	The name of the service the message is for	String
message_status	The status of the message, valid values are: • waiting • timedout	String

	• cancelled • expired • completed • scheduled expiring	
submitted_at	When the request entered the GBTC	String
percent_sent	How much, if any, of the message has been sent	Float

Table G-13. MessageStatus Elements

G.5 Sending an Events Request to the SOAP Server

The GBTC creates and logs events during the life cycle of a message. Events will be created when a message enters the GBTC, is done being processed, when a message expires or is cancelled, or when there is an error with the message. The events can be queried via a Web Services request.

G.5.1 Request

To request any events that may have been generated since the last request, the following elements are required:

Element	Required	Description	Type
dp_name	Yes	The DP's name	String
dp_domain	Yes	The DP's domain	String
dp_password	Yes	The DP's password	String

Table G-14. request_events Elements

G.5.2 Response

The response to a request_events call is the following Events complex type.

Element	Description	Type
success	Whether the SOAP request succeeded or not	Boolean
status	A description of any errors if success is False	String
event_list	A list of Events generated and logged by the GBTC.	Event_Array

Table G-15. Events Elements

The elements of each Event type in the Event_Array are as follows.

Element	Description	Type
event_type	0 = MessageExternalEntry - When the message request enters the GBTC 1 = MessageExternalLeave - When the message has been completely processed by the GBTC 2 = MessageInformation - Information related to the message 3 = MessageWarning - Warnings related to the message	Int

	4 = MessageError - Errors related to the message	
description	Information about the event. May be left empty, mostly used for MessageInformation, MessageWarning, and MessageError types.	String
timestamp	The time the event was logged. Format: **YYYY**-**MM**-**DD**T**HH**:**MM**:**SS**(24 Hour) Ex: 2012-05-30T09:30:10	String
email_id	The unique identifier of the broadcast request	String
num_transmissions	The number of transmission related to the email_id.	Int
transmission_index	Which transmission the event was tied to.	Int

Table G-16. Event Element

Appendix H
Web Services WSDL File

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:types="https://127.0.0.1:8071/ws/types" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="https://127.0.0.1:8071/ws" name="Burst_Web_Services"
targetNamespace="https://127.0.0.1:8071/ws">
   <wsdl:types>
      <xsd:schema elementFormDefault="qualified" targetNamespace="https://127.0.0.1:8071/ws/types">
         <xsd:complexType name="Base">
            <xsd:sequence />
         </xsd:complexType>
         <xsd:complexType name="File">
            <xsd:sequence>
               <xsd:element name="content" type="xsd:base64Binary" minOccurs="0" maxOccurs="1" />
               <xsd:element name="contenttype" type="xsd:string" minOccurs="0" maxOccurs="1" />
               <xsd:element name="filename" type="xsd:string" minOccurs="0" maxOccurs="1" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="CoverageAreaGeo">
            <xsd:sequence>
               <xsd:element name="lat" type="xsd:float" minOccurs="0" maxOccurs="1" />
               <xsd:element name="lon" type="xsd:float" minOccurs="0" maxOccurs="1" />
               <xsd:element name="radius" type="xsd:int" minOccurs="0" maxOccurs="1" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="Repeat">
            <xsd:sequence>
               <xsd:element name="broadcast_count" type="xsd:int" minOccurs="0" maxOccurs="1" />
               <xsd:element name="duration" type="xsd:int" minOccurs="0" maxOccurs="1" />
               <xsd:element name="duration_type" type="xsd:string" minOccurs="0" maxOccurs="1" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="Expiration">
            <xsd:sequence>
               <xsd:element name="duration" type="xsd:int" minOccurs="0" maxOccurs="1" />
               <xsd:element name="duration_type" type="xsd:string" minOccurs="0" maxOccurs="1" />
               <xsd:element name="exact" type="xsd:string" minOccurs="0" maxOccurs="1" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="TimeNow">
            <xsd:sequence>
               <xsd:element name="expiration" type="types:Expiration" minOccurs="0" maxOccurs="1" />
               <xsd:element name="repeat" type="types:Repeat" minOccurs="0" maxOccurs="1" />
               <xsd:element name="use_time_now" type="xsd:boolean" minOccurs="0" maxOccurs="1" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="TimeDelayed">
            <xsd:sequence>
               <xsd:element name="delay" type="xsd:int" minOccurs="0" maxOccurs="1" />
               <xsd:element name="delay_type" type="xsd:string" minOccurs="0" maxOccurs="1" />
```

```
            <xsd:element name="expiration" type="types:Expiration" minOccurs="0" maxOccurs="1" />
            <xsd:element name="repeat" type="types:Repeat" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="TimeExact">
         <xsd:sequence>
            <xsd:element name="datetime_string" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="expiration" type="types:Expiration" minOccurs="0" maxOccurs="1" />
            <xsd:element name="repeat" type="types:Repeat" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Options">
         <xsd:sequence>
            <xsd:element name="priority" type="xsd:int" minOccurs="0" maxOccurs="1" />
            <xsd:element name="retry_on_mdor" type="xsd:boolean" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="BDRReflected">
         <xsd:sequence>
            <xsd:element name="coverage_area_geo_list" type="types:CoverageAreaGeo_Array"
minOccurs="0" maxOccurs="1" />
            <xsd:element name="coverage_area_lda_list" type="types:Int_Array" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="coverage_area_region_list" type="types:String_Array" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="dp_domain" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="dp_name" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="email_id" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="options" type="types:Options" minOccurs="0" maxOccurs="1" />
            <xsd:element name="payload" type="xsd:base64Binary" minOccurs="0" maxOccurs="1" />
            <xsd:element name="service_name" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="status" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="success" type="xsd:boolean" minOccurs="0" maxOccurs="1" />
            <xsd:element name="time_delayed_list" type="types:TimeDelayed_Array" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="time_exact_list" type="types:TimeExact_Array" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="time_now" type="types:TimeNow" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="BCRReflected">
         <xsd:sequence>
            <xsd:element name="dp_domain" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="dp_name" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="email_id" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="status" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="success" type="xsd:boolean" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="MessageStatus">
         <xsd:sequence>
            <xsd:element name="coverage_area_list" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="disposition_requested" type="xsd:boolean" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="expires_at" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="message_status" type="xsd:string" minOccurs="0" maxOccurs="1" />
```

```
            <xsd:element name="payload_size" type="xsd:int" minOccurs="0" maxOccurs="1" />
            <xsd:element name="percent_sent" type="xsd:float" minOccurs="0" maxOccurs="1" />
            <xsd:element name="priority" type="xsd:int" minOccurs="0" maxOccurs="1" />
            <xsd:element name="queue_at" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="service_name" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="submitted_at" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="total_transmission_size" type="xsd:int" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="MessageStatusResponse">
         <xsd:sequence>
            <xsd:element name="dp_domain" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="dp_name" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="email_id" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="message_status_list" type="types:MessageStatus_Array" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="status" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="success" type="xsd:boolean" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Event">
         <xsd:sequence>
            <xsd:element name="description" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="email_id" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="event_type" type="xsd:int" minOccurs="0" maxOccurs="1" />
            <xsd:element name="num_transmissions" type="xsd:int" minOccurs="0" maxOccurs="1" />
            <xsd:element name="timestamp" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="transmission_index" type="xsd:int" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Events">
         <xsd:sequence>
            <xsd:element name="event_list" type="types:Event_Array" minOccurs="0" maxOccurs="1" />
            <xsd:element name="status" type="xsd:string" minOccurs="0" maxOccurs="1" />
            <xsd:element name="success" type="xsd:boolean" minOccurs="0" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="request_events">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="dp_name" type="xsd:string" />
               <xsd:element name="dp_domain" type="xsd:string" />
               <xsd:element name="dp_password" type="xsd:string" />
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="request_eventsResponse">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="result" type="types:Events" />
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="send_bcr">
         <xsd:complexType>
            <xsd:sequence>
```

```
                <xsd:element name="dp_name" type="xsd:string" />
                <xsd:element name="dp_domain" type="xsd:string" />
                <xsd:element name="dp_password" type="xsd:string" />
                <xsd:element name="email_id" type="xsd:string" />
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="send_bcrResponse">
          <xsd:complexType>
             <xsd:sequence>
                <xsd:element name="result" type="types:BCRReflected" />
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="send_bdr">
          <xsd:complexType>
             <xsd:sequence>
                <xsd:element name="dp_name" type="xsd:string" />
                <xsd:element name="dp_domain" type="xsd:string" />
                <xsd:element name="dp_password" type="xsd:string" />
                <xsd:element name="email_id" type="xsd:string" />
                <xsd:element name="service_name" type="xsd:string" />
                <xsd:element name="coverage_area_region_list" type="types:String_Array" minOccurs="0"
/>
                <xsd:element name="coverage_area_geo_list" type="types:CoverageAreaGeo_Array"
minOccurs="0" />
                <xsd:element name="coverage_area_lda_list" type="types:Int_Array" minOccurs="0" />
                <xsd:element name="time_now" type="types:TimeNow" minOccurs="0" />
                <xsd:element name="time_delayed_list" type="types:TimeDelayed_Array" minOccurs="0" />
                <xsd:element name="time_exact_list" type="types:TimeExact_Array" minOccurs="0" />
                <xsd:element name="options" type="types:Options" minOccurs="0" />
                <xsd:element name="payload" type="xsd:base64Binary" minOccurs="0" />
                <xsd:element name="disposition_requested" type="xsd:boolean" minOccurs="0" />
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="send_bdrResponse">
          <xsd:complexType>
             <xsd:sequence>
                <xsd:element name="result" type="types:BDRReflected" />
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="send_msr">
          <xsd:complexType>
             <xsd:sequence>
                <xsd:element name="dp_name" type="xsd:string" />
                <xsd:element name="dp_domain" type="xsd:string" />
                <xsd:element name="dp_password" type="xsd:string" />
                <xsd:element name="email_id" type="xsd:string" />
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="send_msrResponse">
          <xsd:complexType>
             <xsd:sequence>
```

```
                <xsd:element name="result" type="types:MessageStatusResponse" />
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:complexType name="TimeExact_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true" type="types:TimeExact" />
          </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="Int_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true" type="xsd:int" />
          </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="CoverageAreaGeo_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true"
type="types:CoverageAreaGeo" />
          </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="TimeDelayed_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true" type="types:TimeDelayed"
/>
          </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="Event_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true" type="types:Event" />
          </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="String_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true" type="xsd:string" />
          </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="MessageStatus_Array">
          <xsd:sequence>
             <xsd:element maxOccurs="unbounded" name="item" nillable="true"
type="types:MessageStatus" />
          </xsd:sequence>
       </xsd:complexType>
    </xsd:schema>
 </wsdl:types>
 <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="request_eventsRequest">
    <wsdl:part name="parameters" element="types:request_events" />
 </wsdl:message>
 <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="request_eventsResponse">
    <wsdl:part name="parameters" element="types:request_eventsResponse" />
 </wsdl:message>
 <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="send_bcrRequest">
    <wsdl:part name="parameters" element="types:send_bcr" />
 </wsdl:message>
 <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="send_bcrResponse">
    <wsdl:part name="parameters" element="types:send_bcrResponse" />
 </wsdl:message>
```

```
   <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="send_bdrRequest">
      <wsdl:part name="parameters" element="types:send_bdr" />
   </wsdl:message>
   <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="send_bdrResponse">
      <wsdl:part name="parameters" element="types:send_bdrResponse" />
   </wsdl:message>
   <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="send_msrRequest">
      <wsdl:part name="parameters" element="types:send_msr" />
   </wsdl:message>
   <wsdl:message xmlns="https://127.0.0.1:8071/ws/types" name="send_msrResponse">
      <wsdl:part name="parameters" element="types:send_msrResponse" />
   </wsdl:message>
   <wsdl:portType name="Burst_Web_Services_PortType">
      <wsdl:operation name="request_events">
         <wsdl:documentation>Requests all new events for the Service related to this DP
      @param dp_name: The DP's name
      @param dp_domain: The DP's domain
      @param dp_password: The DP's password
      @return: An Events object</wsdl:documentation>
         <wsdl:input message="tns:request_eventsRequest" />
         <wsdl:output message="tns:request_eventsResponse" />
      </wsdl:operation>
      <wsdl:operation name="send_bcr">
         <wsdl:documentation>Takes the parameters for a BCR and sends the cancellation request
      @param dp_name: The DP's name
      @param dp_domain: The DP's domain
      @param dp_password: The DP's password
      @param email_id: The unique identifier of the message
      @return: A BCRReflected response</wsdl:documentation>
         <wsdl:input message="tns:send_bcrRequest" />
         <wsdl:output message="tns:send_bcrResponse" />
      </wsdl:operation>
      <wsdl:operation name="send_bdr">
         <wsdl:documentation>Takes the parameters for a BDR and sends the broadcast request
      @param dp_name: The DP's name
      @param dp_domain: The DP's domain
      @param dp_password: The DP's password
      @param email_id: The unique identifier of the message
      @param service_name: The Service Name the broadcast is to be sent to
      @param coverage_area_region_list: The Broadcast Coverage Area(s) the broadcast request is for
      @param coverage_area_geo_list: The lat/lon/radius area(s) the broadcast request is for
      @param coverage_area_lda_list: The LDA(s) the broadcast request is for
      @param time_now: Used if the broadcast should be sent immediately
      @param time_delayed_list: The list of TimeDelayed objects for the broadcast
      @param time_exact_list: The list of TimeExact objects for the broadcast
      @param options: Any options that are set for the broadcast request
      @param payload: The data to be sent in the broadcast
      @param disposition_requested: Whether or not an MDN is requested by the data provider
      @return: A BDRReflected response
      Takes the parameters for a request and verifies them before inserting the message into the BSR. This
should
      closely mirror what happens in smtp_manager::process_message() and
smtp_manager::process_bxr().</wsdl:documentation>
         <wsdl:input message="tns:send_bdrRequest" />
         <wsdl:output message="tns:send_bdrResponse" />
      </wsdl:operation>
```

```
      <wsdl:operation name="send_msr">
        <wsdl:documentation>Takes the parameters for an MSR and sends the status request
      @param dp_name: The DP's name
      @param dp_domain: The DP's domain
      @param dp_password: The DP's password
      @param email_id: The unique identifier of the message
      @return: A MessageStatusResponse</wsdl:documentation>
        <wsdl:input message="tns:send_msrRequest" />
        <wsdl:output message="tns:send_msrResponse" />
      </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="Burst_Web_Services_Binding" type="tns:Burst_Web_Services_PortType">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="request_events">
        <soap:operation soapAction="request_events" />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="send_bcr">
        <soap:operation soapAction="send_bcr" />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="send_bdr">
        <soap:operation soapAction="send_bdr" />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="send_msr">
        <soap:operation soapAction="send_msr" />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="Burst_Web_Services">
      <wsdl:documentation>WSDL File for Burst_Web_Services</wsdl:documentation>
      <wsdl:port binding="tns:Burst_Web_Services_Binding" name="Burst_Web_Services_PortType">
        <soap:address location="https://127.0.0.1:8071" />
      </wsdl:port>
    </wsdl:service>
```

```
</wsdl:definitions>
```