DTD Application Protocol

Introduction

The DTD (Data Telemetry Device) from Instrumental Solutions uses Iridium's SBD (Short Burst Data) service to transmit data. Iridium delivers the data to the recipient by either email or their Direct IP service. The choice of delivery method is made when the modem is provisioned. Email delivery is subject to the same delays and losses as any other email message of course, but Iridium's email delivery system has proven to be quite reliable. The desired email delivery address is set when the modem is provisioned, and can be changed later. To receive messages using the Direct IP method, a server with a publicly-accessible IP address is required on the recipient side. This is referred to as Mobile Originated Direct IP. The Iridium system will connect directly to the recipient's server when it has a message to deliver. The IP address and TCP port number are set when the modem is provisioned. While faster and more reliable, there is an initial setup fee for this service, whereas email delivery is free. More information is available at NAL Research (http://nalresearch.com).

To conserve bandwidth and reduce message cost, the DTD transmits data in a binary format. This format consists of a variable-length header followed by the data. There are several different types of messages, with the type being identified by the first byte of the header. For Data-type messages, the data values are fixed-length 4-byte quantities in IEEE-754 format. While a description of the IEEE-754 format is outside the scope of this document, there are numerous references available on the Internet. In addition to Data-type messages, there are Alarm messages and Test messages. These are detailed in the Protocol section below.

When receiving messages from Iridium using the Direct IP method, the DTD data (the payload) is encapsulated in a message structure defined by Iridium. The full specification for Direct IP message delivery can be found at:
http://isidev.net/yahoo_site_admin/assets/docs/NAL_DirectIP_SBD.23590710.pdf.

Once the DTD payload is extracted from the message structure, the information in the Protocol section below describes how to decode it.

When receiving messages via email, the DTD data (the payload) is an attachment in the email. Because the data is binary, it is base64 encoded. Extracting the data can be as simple as saving the attachment when viewing the email with an email client such as Microsoft Outlook, Outlook Express, or Mozilla Thunderbird. However, it may be desirable to create an automated system for handling the received emails. All Internet email conforms to the RFC-822 standard for internal message formatting, so understanding this format is critical for anyone wanted to

construct an automated processing system. While information on RFC-822 is widely available on the Internet, the following section is provided as a quick-start guide to help in extracting the attachment containing the DTD data.

Email Delivery

A RFC-822 email consists of multiples lines of text with each line terminated with a carriage-return/linefeed pair, with lines in the header portion containing heading/value pairs. The message body portion contains one or more sections, with text marker "boundaries" used to delineate sections. The header and message body portions are separated by a blank line.

Below is a typical SBD message sent from Iridium. The lines in red are key in picking out the basic message parts. The first 3 red lines are in the header portion, and contain the From address, Subject line, and Content-Type respectively. Obviously, the From address can be used to determine whether or not the message is a SBD message from Iridium. The Subject line contains the originating modem's IMEI number, which uniquely identifies the modem that sent the message. The Content-Type line contains the boundary text marker that is used to locate the start of each section of the message body. Notice that each section in the message body has its own header with several lines of content information. Again, a blank line separates this header from the actual message.

There are 2 sections in the message body, with the first one containing the text of the email message, and the second containing the attachment. The start of each section is marked by a line containing the boundary text defined in the Content-Type line described above. These are the fourth and fifth red lines in the example. Note that the boundary marker has 2 dash characters prepended to it when it appears as an actual marker. Also, the same boundary marker is used to indicate the start of each section; the appearance of the second one serves as an indicator for the start of the next section, which obviously means the end of the previous section.

The email text contains useful information about the modem session that resulted in the data being transmitted, as well as the length of the binary data in the attachment. The DTD data is in the second section, which is base64 encoded. Base64 encoding is used to convert binary data into printable text data. It is a standard format that is supported in many programming languages, and there are plenty of references available on the Internet if implementing your own decoding function. The sixth line in red is the base64-encoded data. Note that there can be more than one line of data, depending on the data length. A blank line may follow the data line(s), which can be treated as an end-of-data indication. A closing boundary marker will always appear. Note that the closing marker is the same as the opening markers except that it has a dash character appended to it. This is the seventh line in the example.

```
X-Apparently-To: pf69bird@yahoo.com via 76.13.11.60; Wed, 06 Apr 2011 05:01:41 -0700
Received-SPF: none (mta1062.mail.bf1.yahoo.com: domain of sbdservice@sbd.iridium.com
does not designate permitted sender hosts)
X-YMailISG: HeqBJKscZAoqlzvZahM8hay8H2WX8eWzO5jWa6OgkY54nZZ0
 48t4hh814foQaR2QjTUcLherV4cPCr3Vw0R_2kJutwZGZT9m_mdC4ULGJFqh
 mhcMBstSVDmHzx9BBYBerTg9yyt0kX.MekycZcXysb3n3idlKXHH_F55wp0i
 i1Gk2cNuZeu5ZN3H_rPJtYBSN.6x4ocSsPmjCrG_i6kbWLWbKqDd5Xx_vuOa
 0fGk4DrecdR7a2Fjnbvb9KYVF4keMo7imK1RzZK5LmzW7DtpG902yxRZac3Y
 X.eqmNvHxBR6U90nrzTT.5lCaMzXLfxN3_3QuRSBJCZ0yai5Y63n0izkM_VF
 rZZkyrAxHcdbOfeyUmdVRgCv4Mjnb7PbIfm3jnrtB7hsJ7Rw2pWr1jaZvyWp
 g12sm5caiXsKpXfbeVY4mRJbPPBiIG8177BES2m6c5O2yHj__qGLBoQ6DVww
 OYjUbrOCoMJVl1HCoL5ECma6MbVE8pUhR_V.8scegaCHE6HZQizzgzHiN0gN
 DMdBLU7BQ8DVKIXG12GTS9oGV_iPDT7DpYL2IPYZxhmi7mnoIrEh735iGGMx
 MneDUziQC_wyqjsMpko7NfonGr2kRPjIrQFi3Lz4VMQAGEuNBPidb_gZZQVV
 S1q.nCIpmwBPX8ss.Q1PA1rQTgIXcvOb_mAVRHyPcdCL1ujaMfzGDApO4Pjm
 vt49vnqWes2XT51DqeamVPCfTxxY4_1oIWYb4nru3nsNq5baq5Nnsp3phNRQ
 AGOao0iVbdZvx7sB3rD5SCQkzjwZ_WYMHPARs74_Ka37YenkHwaqe4sphNwr
 6HhInuEB1RAwtPRs99Qd6lsMczTC4GNmZ18hOTWZ3gAP.qudwpF_o1G0t6nn
 7IJ2d1fxqKd_IDclBgJAGVGF_yAV1d7W7xKIv.gGdhvgFHhKHGHh2.wn.SfK
 _lB_uoSoE1Tj9j6H1QrOpqExY47JE9cVO4euKMIHULGZ6oy5Ax5H48yiTQbu
 xvCUQdgkzYlGKrS3r73d9SeL54jvHJz68YQjJloglnowO5.Sr8brTUAjg_xX
 IMZ0kxs.chV1.rP3s5T8WzcMbmjPpvY.3AFe2VM7lD3p1BRhay20XUhbGxZe
 Iyi2bdgsO2UEIrSF4_nh8y23tsyvK2Tip7olVynOJnHpPiu0ZN.sv94GhKg3
 F8WzHwi_LWld9eZRCcaeRDcDrakhyPjA9.Ew7uVF9dEh.AztNkvA0ckAxv9t
 ZoS6zqcXEdlg67KNLq3zSiWhK26l6qxt_g7wiVLeCmk9m2IB4s1PvZrrSH0F
 KoOnOhF7nBF37.zmCUAZQRksqcEd62C0DNpagpX4JreIOHZL4yds3pDLZf4h
 TvD9HHrtwDqTsHhzuKlNjRNBB7.Ohje_cTUwvyJ1kfZeySa5iSOYLCkU0nx0
 AYlC_91Mgh.U0436VQJn360lHpczz6.Go_qHLxUiWPuiNF6dAbXuO4Fdwe9N
 58Jdl8XcdvwTfR.4u7GHzy4bznfFM06LuF79XAGTjcvw7S_DrAyihRx76AzV
 CfliZs9hR4d7xHwPX79dmJDnRaKue57soaHJiM3qGde9oDXdCKKx7kIICrqr
 9KwtAZF6r7vX_a1kZnx_IXvKaP_RbAjqJwm8Np8FjmsYM4B2jooQlsJEUKTs
 xvSz9.ZqUbN3mn9XfOs7ZiDcs3TbTsgUQ49poEHphinxgT3of8fM8KC3hul2
 PjeR7ML.YSZtgnhvE24_7YrPf6GcoF9IxOHWWKKKaH5ICH8-
X-Originating-IP: [216.55.166.79]
Authentication-Results: mta1062.mail.bf1.yahoo.com  from=sbd.iridium.com;
domainkeys=neutral (no sig);  from=sbd.iridium.com; dkim=neutral (no sig)
Received: from 127.0.0.1  (EHLO 206-225-81-4.dedicated.abac.net) (216.55.166.79)
  by mta1062.mail.bf1.yahoo.com with SMTP; Wed, 06 Apr 2011 05:01:41 -0700
Received: by 206-225-81-4.dedicated.abac.net (Postfix)
       id 49831670255; Wed,  6 Apr 2011 07:01:39 -0500 (CDT)
Delivered-To: dtddemo@localhost.localdomain
Received: from 206-225-81-4.dedicated.abac.net (unknown [127.0.0.1])
       by 206-225-81-4.dedicated.abac.net (Postfix) with ESMTP id 3C7BD670247
       for <dtddemo@localhost.localdomain>; Wed,  6 Apr 2011 12:01:39 +0000 (UTC)
Received: by 206-225-81-4.dedicated.abac.net (Postfix, from userid 110)
       id 30D93670274; Wed,  6 Apr 2011 12:01:39 +0000 (UTC)
X-Original-To: sbd1@citssi.com
Delivered-To: sbd1@citssi.com
Received: from 206-225-81-4.dedicated.abac.net (unknown [127.0.0.1])
       by 206-225-81-4.dedicated.abac.net (Postfix) with ESMTP id 14B78670247
       for <sbd1@citssi.com>; Wed,  6 Apr 2011 12:01:39 +0000 (UTC)
Received: from istbsp01.sbd.iridium.com (unknown [12.47.179.11])
       by 206-225-81-4.dedicated.abac.net (Postfix) with ESMTP
       for <sbd1@citssi.com>; Wed,  6 Apr 2011 12:01:39 +0000 (UTC)
Reply-To: <sbdservice@sbd.iridium.com>
From: sbdservice@sbd.iridium.com
To: sbd1@citssi.com
Subject: SBD Msg From Unit: 300034012704760
MIME-Version: 1.0
Content-Type: multipart/mixed;boundary="SBD.Boundary.605592468"
Message-Id: <19740412024625.E9DC4FEBFF2EEEE1@istbsp01.sbd.iridium.com>
Date: Wed,  6 Apr 2011 12:01:36 +0000 (GMT)
```

```
SBM Message
--SBD.Boundary.605592468
Content-Type: text/plain;charset=US-ASCII
Content-Disposition: inline
Content-Transfer-Encoding: 7bit

MOMSN: 1125
MTMSN: 0
Time of Session (UTC): Wed Apr  6 12:01:35 2011
Session Status: 00 - Transfer OK
Message Size (bytes): 54

Unit Location: Lat = 32.294306 Long = -90.849393
CEPradius = 2



--SBD.Boundary.605592468
Content-Type: application/x-zip-compressed; name="SBMmessage.sbd"
Content-Disposition: attachment; filename="300034012704760_001125.sbd"
Content-Transfer-Encoding: base64

RFcAAgABfbQwQMAAAAB9tDCAwAAAAH20MMDAAAAAfbQxAMAAAAB9tDFAwAAAAH20MYDAAAAA

--SBD.Boundary.605592468—
```

Protocol

The first byte of the DTD data is a message-type indicator which is used to determine how to decode the rest of the message. There are currently 3 message types defined:

- Data message, where the first byte has a hex value of 0x44 (‘D’).
- Alarm message, where the first byte has a hex value of 0x41 (‘A’).
- Test message, where the first byte has a hex value of 0x54 (‘T’).

Data Messages

The following table describes the format of Data messages:

Offset	Description	Size	Range
0	Message type byte	1	0x44 (‘D’)
1	Sequence number	1	0x00 – 0xff
2	Instrument ID (MSB,LSB)	2	0x0000 – 0xffff
4	Field mask (MSB,LSB)	2	0x0000 – 0xffff
6	First data record	8+	0x00 – 0xff

Table 1 - Data message header format

The byte at offset 0 is always ‘D’ indicating that it is a Data message.

The byte at offset 1 is a sequence number generated by the DTD. It starts at 0x00 whenever the DTD is powered up, and is incremented by one for each message generated by the DTD. It is used to detect missing messages.

The bytes at offset 2 and 3 form a 16-bit Instrument ID number that is used to determine the instrument type and the fieldname list for the instrument. The byte at offset 2 is the most-significant byte (MSB), and the byte at offset 3 is the least-significant byte (LSB). See the Instrument ID section below for a list of Instrument IDs.

The bytes at offset 4 and 5 form a 16-bit Field mask value that is used to determine which fields in the fieldname list are enabled on the DTD and present in the DTD data. The byte at offset 4 is the most-significant byte (MSB), and the byte at offset 5 is the least-significant byte (LSB). Bit 0 of the mask corresponds to the first field in the fieldname list, bit 1 for the second field, and so on. If a bit is 0, the corresponding field is not enabled on the DTD, and there is not a data value for it in the received message. If a bit is 1, the field is enabled and there is a data value for it in the message. Each data value is a 4-byte floating-point number in IEEE-754 format. One or more data values are grouped along with a timestamp into a data record. A data record represents a single sampling point, whose length is determined by the number of fields that are enabled. For each field that is enabled, there will be a 4-byte data value in the record. The data values are arranged in the same order as the fields in the fieldname list, so processing them is a simple matter of testing each bit in the mask value starting with bit 0, and decoding the next 4-byte value for each bit that is set. If a bit isn't set, simply skip the corresponding field in the fieldname list. The minimum number of fields is 1, the maximum is 16. After testing all 16 bits in the Field mask and decoding data for each one that is set, the next 4 bytes of data will be the timestamp for the next data record unless you have reached the end of the data. A DTD Data message contains one or more data records. The following table describes the data record:

Offset	Description	Size
0	Timestamp	4
4	First field, IEEE-754 value	4
8	[Second field, IEEE-754 value]	4

Table 2 – Data record format

The timestamp is a 32-bit value that contains year, month, day, hour, and minute values. These values are encoded as bit fields and have the following format:

Bit positions	Value	Range
0 – 5	Minute	0 – 59
6 – 10	Hour	0 – 23
11 – 15	Day	1 – 31
16 – 19	Month	1 – 12
20 – 31	Year	0 – 4095

Table 3 – Timestamp field format

Alarm Messages

The following table describes the format of Alarm messages:

Offset	Description	Size	Range
0	Message type byte	1	0x41 ('A')
1	Instrument ID (MSB,LSB)	2	0x0000 – 0xffff
3	First alarm record	variable	0x00 – 0xff

Table 4 – Alarm message header format

The byte at offset 0 is always 'A' indicating that it is an Alarm message.

The bytes at offset 1 and 2 form a 16-bit Instrument ID number that is used to determine the instrument type and the fieldname list for the instrument. The byte at offset 1 is the most-significant byte (MSB), and the byte at offset 2 is the least-significant byte (LSB). See the Instrument ID section below for a list of Instrument IDs.

Alarm records are variable length, using a comma character as the record delimiter. An alarm message may contain 1 or more records. The following table describes the record format:

Offset	Description	Size
0	Field number in ASCII	2
2	Alarm type: 'H' or 'L'	1
3	Timestamp in ASCII	16
19	Data value in ASCII	Variable

Table 5 – Alarm record format

Test Messages

Test messages have an upper-case 'T' (0x54) as the Message type byte, and consist of plain text. The first word of a test message is "Test", with the 'T' serving as both the Message type byte and the first character of the text. Therefore, when processing a test message, simply display the text beginning at offset 0.

Instrument ID's

The following table lists some common Instrument ID's and the associated instrument name and field list.

ID	Name	Fields (in order)
1	InSitu LevelTroll 700	Pressure, Temperature, Level
2	Ott RLS	Distance
8	Hydrolab DS5X	TemperatureC, pH, Redox, SpecCondmS, Resistivity, SalinityUSGS, TDS, TDG, Depth, NH3+NH4, NO3-,CL-, turbidity, LDO%, LDOmg/L, chlorophyll
9	Isco 2105	Level, Velocity, Flow, Volume, Voltage, TemperatureC, AnalogCh1, Battery, Conductivity, SpecCond, DissSolids, Salinity, DO, pH, ORP, Turbidity