Sabre® Data Source

SDS Overview

February 2000

Contents

General Information

Contact

Introduction

For several years, the Personal Computer (PC) has been used as a user-friendly interface to the SABRE Host software. A PC-based program allows users to enter parameters for a host command. These commands are then formatted and sent to SABRE for a response. When the SABRE response is received, the PC program picks the responses off the host "screen" and displays them in an easy to read format. With this method, PC programs are restricted to the format of the host command and the host response display.

The "Challenge"

When the host screen formats change, PC programs (being restricted to the host screen formats) display blanks or incomplete data, or sometimes fail. Failure due to host changes results in an emergency change and implementation of a new version of the PC program. The new version must then be distributed to all users (sometimes thousands). A utility was needed that makes the PC program independent of the host screen format.

The "Solution"

A solution has been designed allowing the PC and the host system to exchange structured messages that are easy for the machine to decode. The format and protocol for these messages is known as Generalized Data Streams (GDS) (GDS is the utility program, SABRE Data Source - SDS - is the SABRE result).

When an application program uses SDS, it is engaging in a form of Electronic Data Interchange (EDI). The format of a Generalized Data Streams message is very similar to the format of an EDIFACT message.

This overview will explain the layout of a SDS message, the layout of a Message Definition Record (MDR) which describes the layout of a particular SDS Message and how to use the MDR to properly decode a SDS message.

Definitions

Entry	A message that has been sent from the client system to the host system.
SDS Message	A structured message that is composed of a SDS message header and one or more SDS message segments.
SDS Message Header	The SDS Message Header uniquely identifies the layout of a SDS Message. It is composed a SDS Message identifier and a version code.
SDS Message Identifier	A fourteen-byte character identifier that identifies the type of message. When taken with the SDS Message Version, it uniquely identifies the layout of a particular SDS message.
SDS Message Segment	A SDS Message Segment is composed of a SDS Segment Header followed by one or more data elements.
SDS Message Version	A two character numeric code, which identifies the layout version of a particular SDS message.
SDS Segment Header	A SDS Segment Header is used to identify the layout of a SDS message. It consists of a six-character segment identifier and a one-character numeric nesting level.
Message Definition Record	A record, which defines the layout of a SDS message. It is similar in format to a SDS message and can be downloaded from the host system.
Response	A message sent back from the host system to the client system in response to an Entry. See Entry.

SDS Message Format

A SDS Message is, at its most elemental level, simply a delimited stream of character data. At a slightly higher level, the SDS Message is a structured record whose fields are separated from one another through the use of special character delimiters. These characters were previously undefined characters in the SABRE character set. The message format has been designed such that it may be sent through the existing Sabre six-bit network as well as standard networks.

SDS Message Delimiters

The following four delimiters can occur in a SDS Message: Element Delimiter, Segment Delimiter, End of Repeat Data Delimiter, and End of Message Delimiter. The Element Delimiter is used to signal the end of one field and the start of another. The Segment Delimiter is used to signal the start of a message segment. The End of Repeat Data delimiter is used to signal the end of a repeating sequence of data element values.

The SDS Message Delimiters are as follows:

Delimiter Name	EBCDIC Hex	EBCDIC Character	ASCII Hex	ASCII Character	SABRE HEX
Element Delimiter	**0x50**	**&**	**0x86**	**†2F**	
Segment Delimiter	**0x5D50**	**)&**	**0x89**	**‰**	**2E2F**
End of Repeat Data	**0x5D61**	**)/**	**0x88**	**ˆ**	**2E11**
End of Message	**0x5D505D50**	**)&)&**	**0x8989**	**‰‰**	**2E2F2E2F**

SDS Message Layout

The format of a SDS message is as follows:
<Block Header><Message Identifier><Element Delimiter><Message Version>
<Segment Delimiter><Segment Identifier><Element Delimiter><Nesting Level> See Note 2
<Element Delimiter><Data Element Value> -- See note 1
<End of Message Delimiter>

Note 1: The sequence <Element Delimiter><Data Element Value> occurs 1 or more times within a message segment.

Note 2: A message segment occurs zero or more times within a message.

SDS Block Header

The block header is used to sequence the data blocks sent between the client and the host. When a SDS message must be broken into multiple blocks, the block header will be prefixed to each block so that the receiver can reassemble the blocks back into a complete SDS message. The format of the block header is as follows:
<Block Sequence Number><Total Message Size>

The block sequence number is 3-digit number that is right justified and filled with zeros on the left.

The total message size is a five-digit number that is also right justified and filled with zeros on the left. In the block header for block one, this value will be the total size of the SDS message across all the blocks excluding the size of the block headers. For the rest of the blocks, this field is simply a placeholder and will be filled with zeroes.

SDS Message Header

The SDS message header is used to identify the purpose and layout of a particular message. The SDS header consists of two fields, a message identifier field of 14 bytes and a version number of 2 digits. The format of the message header is as follows:

<Message Identifier><Element Delimiter><Message Version>

The message identifier is logically broken down into 3 separate fields. A product code of three characters, a requester / recipient code of three characters, and a function code of eight characters. This logical breakdown is for information only and will usually have no impact on the construction of an API.

The message version is 2-digit value, which starts at 01 for the first version of a message, goes through 99, and rolls over back to 00 for the hundredth version.

SDS Segment

The SDS message segment is logically grouped collection of data elements. The segment is identified by the use a segment header. The segment header consists of two fields, a segment identifier of six characters, and a nesting level of one digit. The layout of a SDS segment is as follows:

<Segment Delimiter><Segment Identifier><Element Delimiter><Nesting Level>
<Element Delimiter><Element Value>

The <Segment Delimiter> signals the start of a new segment.

The <Segment Identifier> (or segment id as its commonly referred to) is a six character identifier, which when taken with the nesting level, can uniquely identify a particular segment layout within this message. The segment will be described in the MDR corresponding to the message.

The <Nesting Level> field is a one-digit value ranging from 0 to 9. This is used to represent the amount to which a segment is nested within the previous segment. An unnested segment has a value of zero for this field.

The sequence <Element Delimiter><Element Value> must occur at least one time and may occur more than one time depending on the layout of the segment which is detailed by the MDR

SDS Data Element

The data element contains the data values being sent. In a SDS message, there are two types of data elements: repeating and non-repeating. A repeating element is one where multiple values are sent for a data element. These values will be separated by an element delimiter and the last element value for this data element will be appended with a end of repeat data delimiter to signal the end of element values for this data element. The element delimiter for the next data element can safely be omitted since the end of repeat data signals the end of the data element. The end of repeat data can safely be omitted if it occurs at the end of a segment since the next character will either be a segment delimiter (which signals the start of a new segment) or the end of message delimiter (which signals

the end of this SDS message). A non-repeating data element will be a single value. The format for a simple (non repeating) element

The format for a repeating element is:

<Element Delimiter><Element Value>
<Element Delimiter><Element Value>
<End of Repeat Data Delimiter>

Example
Given the following name display from SABRE

1.3 Ervin / Steve / Carol / Michael
2.1 Martin / Gene

The corresponding SDS message would be:
00100075AIRAALNAMEDISP†01‰NAMSEG†0†1†3†Ervin†Steve†Carol†Michael ^‰NAMSEG†0†2†1†Martin†Gene^‰‰

Dissecting the message, we have:

1	001 is the block number.
2	00075 is the total length of the SDS message (which follows this field).
3	AIRAALNAMEDISP is the Message Id.
4	01 is the message version. Notice that is separated from the Message Id by the SDS element delimiter.
5	‰, the SDS segment delimiter, signals the start of a new segment. Following this delimiter is segment id with a value of NAMSEG which identifies that this segment is a names segment.
6	0 is the nesting level. A segment nested at level zero is an un-nested segment. Notice that it is separated from the segment id by an element delimiter.
7	1 is the first element value in this segment. This element is the Party Number. It is separated from the nesting level by an element delimiter.
8	3 is the second element value. This element is the Number in Party. It is separated from the previous element by an element delimiter.
9	Ervin is the third element value. This element is the Surname. It is separated from the previous element by an element delimiter.
10	The next element is a repeating element. The element is First Name. The values are Steve, Carol, and Michael and are separated from one another (and the previous element) by element delimiter.
11	^, end of repeat data delimiter signals the end of the repeating element.
12	‰, the SDS segment delimiter, signals the start of a new segment. Following this delimiter is segment id with a value of NAMSEG which identifies that this segment is a names segment.
13	0 is the nesting level. A segment nested at level zero is an un-nested segment. Notice that it is separated from the segment id by an element delimiter.
14	2 is the first element value in this segment. This element is the Party Number. It is

	separated from the nesting level by an element delimiter.
15	1 is the second element value. This element is the Number in Party. It is separated from the previous element by an element delimiter.
16	Martin is the third element value. This element is the Surname. It is separated from the previous element by an element delimiter.
17	The next element is a repeating element. The element is First Name. Even though this is a repeating data element, there is only one element value which is Gene and is separated from the previous element by element delimiter. The end of repeat data delimiter follows this element.
18	‰‰ is end of message delimiter. This signals the end of message. Any data after this delimiter should be ignored.

MDR Databases

Central to the concept of SDS, is the fact that MDRs may be downloaded on demand from the SABRE host. This requires that the host keep a central database of MDRs so that clients may download a MDR at any time. The client should keep a local database of MDRs that mirrors the host database and download a MDR only when it does not exist in its local database. Furthermore, to protect against fallback of the Host application, the local database should keep the current and the previous versions of a MDR available. The databases used by SDS are described below:

SDS Production Database

A central database in SABRE that contains the next , current, and previous versions of all MDRs in use. Prior to new version of a host application going on-line, MDRs are loaded via tape in the next position (slot) of MDR database. Right before the new host software goes on-line, the previous version is deleted, the current is moved to the previous, and the next becomes the current version. Either the current or previous versions of a MDR are available for downloading by the client. The Developer must be prepared for MDR version control. **SDS will provide support for a previous MDR version for only 60 days**. At the end of 60 days, all applications must be prepared to utilize the current version or risk having their application become non-responsive.

Local MDR database

A database maintained by the client that mirrors the current and previous slots of the host MDR database. The Message Definition Record (MDR) describes the layout of a SDS message. Like the SDS message, it's fields are separated from one another by the use of SDS delimiters. While the SDS message is used to transmit actual data, the MDR describes the type of data being sent.

The MDR uses much the same delimiters as the SDS message. The only exception is that the End of Repeat Data delimiter is not used. The delimiters are:

Delimiter Name	EBCDIC Hex	EBCDIC Character	ASCII Hex	ASCII Character	SABRE HEX
Element Delimiter	0x50	&	0x86	†	2F
Segment Delimiter	0x5D50	)&	0x89	‰	2E2F
End of Message	0x5D505D50	)&)&	0x8989	‰‰	2E2F2E2F

MDR Layout

The MDR layout is as follows:
<Block Header><Message Header>
<Segment Header>
<Element Record>
<End of Message Delimiter>

If we expand the headers, the MDR will appear as follows:

<Block Number><Message Size><MDR Id><Element Delimiter><MDR Version>
<Segment Delimiter><Segment Id><Element Delimiter><Nesting Level>
<Element Delimiter><Mandatory/Optional Flag><Element Delimiter>
<Repeat Factor><Segment Delimiter>
<Element Id><Element Delimiter><Data Type><Element Delimiter>
<Minimum Size><Element Delimiter><Maximum Size><Element Delimiter>
<Mandatory/Optional Flag><Element Delimiter><Repeat Factor>
<Element Delimiter>
<End of Message Delimiter>

Note: At least one segment must be defined within an MDR. At least one element must be defined within a segment.
For the last element record within a segment, the final element delimiter must be omitted.
MDR Block Header
The block header for the MDR is identical to the block header for the

The block header is used to sequence the data blocks sent between the client and the host. When a SDS message must be broken into multiple blocks, the block header will be prefixed to each block so that the receiver can reassemble the blocks back into a complete SDS message. The format of the block header is as follows:

<Block Sequence Number><Total Message Size>

The block sequence number is 3-digit number that is right justified and filled with zeros on the left.

The total message size is a five digit number that is also right justified and filled with zeros on the left. In the block header for block one, this value will be the total size of the SDS message across all the blocks excluding the size of the block headers. For the rest of the blocks, this field is simply a placeholder and will be filled with zeroes.
MDR Header

The MDR Header is identical to the SDS Message header. This fact allows an API to use the SDS message id and version has a key to retrieve the MDR corresponding to that message. The MDR Header consists of two fields, a MDR Id of 14 bytes, and a MDR version number of two digits. The two fields are separated from each other by an element delimiter. The format of the MDR header is as follows:

<MDR Identifier><Element Delimiter><Message Version>

The MDR id (MDR identifier) is logically broken down into 3 separate fields. A product code of three characters, a requester / recipient code of three characters, and a function code of eight characters. This logical breakdown is for information only and will usually have no impact on the construction of an API.

The MDR version is 2-digit value, which starts at 01 for the first version of a MDR, goes through 99, and rolls over back to 00 for the hundredth version.

MDR Segment Record

The MDR segment record describes a SDS Segment that may (or must) appear in a SDS message. The first two fields of a MDR segment record are identical to the first two fields of a SDS message. This allows an API to use these two fields from the SDS message as a key to find the corresponding segment in the MDR. The next two fields, Mandatory / Optional flag and Repeat Factor, indicate properties of the segment. The layout of the segment record is as follows:

<Segment Delimiter><Segment Id><Element Delimiter><Nesting Level>
<Element Delimiter><Mandatory / Optional Flag><Element Delimiter>
<Repeat Factor><Segment Delimiter>

A segment delimiter signals the start of a segment record. Following this is the segment id, which is six characters long. The next field is the nesting level; this field is a single digit number in the range zero to nine. The nesting level is separated from the segment ID by an element delimiter.

Following the nesting level, and separated from it by an element delimiter, is the mandatory / optional flag. This field is a one character field and may have of value of either 'M' or 'O' (for Mandatory or Optional respectively). A mandatory segment must appear at least once. Please see appendix B concerning Nesting. Nested mandatory segments must appear at least once under their parent segment. An optional segment is one that may appear within a SDS message, but it is not an error if it is omitted.

The last field in the segment record is the repeat factor. This field is a one to 3 digit field and indicates how many times a segment may appear within a message. The minimum value for this field is one and the maximum is 999. The value 999 in this field should treated as an infinite repeat factor.

The segment record is terminated by a segment delimiter. This allows the new fields to be inserted in the segment record and still be processed by older software.

Following the end of the segment record is at least one element record. The element record is described in the next section.

MDR Element Record

The MDR Element Record describes the properties of an element within a segment. There are five fields that make up an MDR element record:

Element Id (4 digit base-36 number)
Data Type (1 character)
Minimum Size (1 to 4 digits)
Maximum Size (1 to 4 digits)
Mandatory / Optional Flag (1 character)
Repeat Factor (1 to 3 digits).

Element record

The layout of a MDR Element Record is as follows:
<Element Id><Element Delimiter><Data Type><Element Delimiter>
<Minimum Size><Element Delimiter><Maximum Size><Element Delimiter>
<Mandatory/Optional Flag><Element Delimiter><Repeat Factor><Element Delimiter>

The Element ID is a four digit base-36 number. A base-36 number is a number whose digits range from 0-9,A-Z. Each element identifier uniquely defines a certain element such as flight number, number in party, etc. The data elements and their definitions are available from the application representative. An element delimiter will be found following the element id field. Data Type
The Data Type is a one character field, which describes the type of data that will be transmitted: Current data types are:
C Character (Any character found in the SABRE character set.)

N Numeric (0-9)
D SABRE Date (ddmmmyyyy)
T Local Time
G GMT Time
L Local Date (USA Format)
U GMT Date (USA Format)
F Local Date (European Format)
E GMT Date (European Format)
K Local Date (Julian Format)
J GMT Date (Julian Format)
I Local Date (Commercial Format)
M GMT Date (Commercial Format)
An element delimiter is used to separate the Data Type field from the minimum size field, which follows it.

Minimum Size

The minimum size field defines the minimum number of characters (or digits) a data element must be in size if it is not omitted (an omitted element will appear as a zero length data value within a SDS message. The minimum size field must be in the range 1 to 4096 inclusive. An element delimiter follows the minimum size to separate from the next field, which is the maximum size field.

Maximum Size

The maximum size field defines the maximum width in characters (or digits) of a data element value. The maximum size field must be in the range 1 to 4096 inclusive and must be greater than or equal to the value of the minimum size field. An element delimiter follows the maximum size field.

The next field is the Mandatory / Optional Flag. This field is a one character value that has a value of either M for mandatory or O for optional. A mandatory data element is an element that must appear in the message segment. Conversely, an optional data element is an element that may be omitted. An optional element that has been omitted will appear as zero length element value within the SDS message. An element delimiter follows the Mandatory / Optional Flag.

Repeat Factor

The last field in an element record is the repeat factor. This field tells how many times this element occurs within the segment. This is a one to three digit field and can have a value from 1 to 999. When this field has a value of more than 1, the element is considered to be a repeating element. A repeating element is represented in the SDS message as sequence of zero (in the case an optional element has been omitted) or more data values which are separated from one another by an element delimiter and terminated by an end of repeat delimiter. A value of 999 for the repeat factor is used to signal an infinite

sequence of values for this element. Unless this element record is the last element record in the MDR segment, an element delimiter follows the repeat factor. When this is the last element record in a segment, either an end of message delimiter will occur (to signal the end of the MDR), or a segment delimiter (to signal the start of a new segment).

Example

The following is example print out of how the MDR might look when printed out by the SDS off-line database. The MDR given is the one corresponding to the example message given in the previous section.

AIR AAL NAMEDISP 01

Element Data Min Max M/O Rpt Description
Id Type Size Size Flag Fctr

Product AIR Req/Rcpt AAL Function NAMEDISP Version 01
Segment NAMSEG Nesting 0 M/O Flag M Repeat Factor 099
0029 N 1 2 M 1 Group Number
000I N 1 2 M 1 Number In Party
0021 C 1 30 M 1 Surname / Last Name
0022 C 1 30 O 99 Forename / First Name

This is the logical view of the MDR. When downloaded from the host system, the MDR will look like the following:

00100102AIRAALDISPNAME†01‰NAMSEG†0†M†99‰0029†N†1†2†M†1†
000I†N†1†2†M†1†0021†C†1†30†M†1†0022†C†1†30†O†99‰‰
In the SABRE processing environment, the PC (client) will send an entry to the SABRE host (server) and wait for the response to come back before sending the next entry. A SDS API needs to manage the sending and receiving of these entries and responses on behalf of the Application.

Nesting Rules

There are times when a segment must be nested within another segment. Traditionally, a nested segment might be represented as follows:
Message A
Segment 1 Nest level 0
Element 1
Element 2
Element 3
Segment 2 Nest Level 1
Element A
Element B

Element C
End of Segment
Element 4
Element 5
End of Message

This structure is represented in a SDS message as:

Message A Version 01
Segment 1 Nesting Level 0
Element 1
Element 2
Element 3
Element 4
Element 5.
Segment 2 Nesting Level 1
Element A
Element B
Element C
End of Message

Nesting Rules

1) No segment may have itself nested immediately under it.

2) The repeat factor for a segment at nest level 0 specifies the maximum number of times a segment may appear in the entire message. For segments with a nesting level of more than zero, the repeat factor specifies the maximum number of times the segment may appear under its parent segment.

3) If a nested segment is omitted, all segments nested under it must also be omitted.

4) When changing levels, the nesting level may only increase by one. It may however, decrease by any. amount.

5) When a nested segment is listed as mandatory, it must appear at least once under its parent segment, if and only if, the parent segment appears in the message.

Intentionally left blank

SARAALGDS0STAR Version 1

Segment ID STRLST
Nesting Level 0 - Mandatory/Optional Flag O - Repeat Factor 999

Element ID	Data Type	Min. Size	Max. Size	Mandatory /Optional	Repeat Factor	Description
01PA	N	1	4	O	001	Line Number
01IZ	C	1	55	O	001	File Name/File Identifier
01XL	C	1	60	O	001	Free Text
00LX	N	1	1	O	001	Processing Level
01YN	C	1	1	O	001	Level 2
01YO	C	1	1	O	001	Attached TPR
00YT	C	1	1	O	001	Indicator (Generic)
000T	C	1	70	O	001	Free Text

Segment ID STARID
Nesting Level 0 - Mandatory/Optional Flag O - Repeat Factor 999

Element ID	Data Type	Min. Size	Max. Size	Mandatory /Optional	Repeat Factor	Description
01IZ	C	1	55	O	001	File Name/File Identifier
00WV	C	1	3	O	001	Category
01YP	C	1	3	O	001	Viewership
00GJ	C	3	5	O	001	IATA City Code
00G6	C	1	1	O	001	Agent SABRE Signon Duty Code
01XR	C	3	5	O	001	SABRE Sign
001H	C	5	7	O	001	Date of Creation (DDMMYY)
01YQ	N	1	10	O	001	Access Number
01YS	C	1	10	O	001	Purge Data
000T	C	1	70	O	001	Free Text

Segment ID STR000
Nesting Level 1 - Mandatory/Optional Flag O - Repeat Factor 999

Element ID	Data Type	Min. Size	Max. Size	Mandatory /Optional	Repeat Factor	Description
01PA	N	1	4	O	001	Line Number
002R	C	1	10	O	001	Type
007K	C	1	2	O	001	Qualifier
000T	C	1	70	O	001	Free Text

Application	MDR Name	Input Entry Expected Response
Air	AIRAALCONNTIME	Connection Time Data
Air	AIRAALCONPOINT	Connection Point Data
Air	AIRAALFLIFO000	Flight Information Data
Air	AIRAALSADAVAIL	Availability or Schedule Data
Air	AIRAALSADSKED0	Verify Availability Data
Car	AIRAALCARPOLIC	Car Policy Data
Car	CAR000CFR00000	Car Shoppers List Data
Car	CAR000CQR00000	Car Rate Quote Data
Car	CAR000CRR00000	Car Availability Data
Car	CAR000RUR00000	Car Rules Response Data
Car	CAR000SEQ008000	Sell Query
Car	CAR000SER00800	Sell Response
Drs	DRSAALGDS0DRS0	DRS Response Data
Fare Quote	AIRAALFAREQUOT	Fare Quote Data
Hotel	HTLAALPRPDESCR	Hotel Description and Hotel Rate Display Data
Hotel	HTLAALPRPOLICY	Guarantee and Deposit Information Data
Hotel	HTLAALPRPRTCDE	Property Rate Code Data
Hotel	HTLAALSHOPLIST	Hotel Shoppers List Data
PNR	AIRAALSAD0PNR0	PNR Data
Pricing	AIRAALAIRPRICE	Pricing Data
Queues	AIRAALQUEUEACC	Access a Specific PNR Queue
Queues	AIRAALQUEUEANA	Queue Analysis of PNRs
Queues	AIRAALQUEUEMOV	Move PNRs from One Queue to Another Queue
Queues	AIRAALQUEUEQMR	Restricted Message Queue Viewership
Queues	AIRAALQUEUESET	Set UMSG Indicator to Advise When Messages are Received in a Specific Queue
Queues	AIRAALQUEUESQE	Create Special Queue Number for PNR Placement
Queue	AIRAALQUEUETRN	Close Queue
Queue	AIRAALQUEUECNT	Display Queue Count
Stars	SARAALGDS0STAR	Star Information