Bundled Sell Utility for Application Developers

MODULE 202
REVISION 20050727

27 July 2005

Bundled Sell Documentation for Developers

What Is the Bundled Sell Utility?

The Bundled Sell utility is a Sabre host command that provides the capability to sell multiple segments of an air itinerary as a single transaction. This ability provides significant Scan cost and transaction volume reductions. Previous to the release of this functionality, developers were limited to utilizing the segment "long-sell" entries which required applications to construct an itinerary with individual segments, which in turn required multiple transactions to be sent to the host.

Why Use the Bundled Sell Utility?

Bundled Sell transactions are preferable to standard single-segment sell transactions for a variety of reasons.

- Bundled Sell transactions can be used to reduce the cost of transactions during the air shopping process when utilizing Sabre Tripsearch (JR) functionality. The standard method of selecting an itinerary option can be relatively costly and Bundled Sell represents an opportunity to avoid incurring the higher cost of that transaction
- In other applications, use of Bundled Sell transactions can result in a fewer number of "hits," or queries, to Sabre, which results in reduced resource utilization on the part of Sabre, which in turn results in a reduction in Scan charges related to the use of a given application. (For further information related to Scan, consult your Sabre e-Business Development Manager, Sabre Account Manager, or Sabre Account Executive.)
- Use of the Bundled Sell transaction type may result in reduced overall transaction response time, which may result in faster application response.
- Due to inventory management techniques (also known as "married connection logic") put into place in Sabre by several carriers, it is necessary to select, or sell, a combination of flights as a linked series (hence, the term "Bundled") rather than as individual components to form a complete itinerary.

A note to users of Sabre Web Services: The OTA_AirBookLLSRQ Service uses the Bundled Sell utility to accomplish the sale of air itineraries.

Reducing the Cost of Tripsearch Transactions

Sabre trip-led pricing, more commonly known as Tripsearch, is an air itinerary shopping process that performs a search for possible acceptable itineraries based upon qualifiers input by a user or application. The Tripsearch process involves two transactions. A shop/qualifier transaction initiates the shopping algorithm while simultaneously providing the parameters of the search; Sabre then returns a list of possible itineraries based upon the criteria provided. Following evaluation of those possible itineraries, a selection transaction is made which chooses an option from the list of results.

A typical use of the Tripsearch functionality is as follows:

1. Application performs Tripsearch shop/qualifier transaction
2. Sabre returns results in the form of possible itineraries
3. Possible itineraries evaluated
4. Selection made, Tripsearch selection transaction performed

Both transactions in this series (shop/qualifier and selection) are priced at the highest Scan pricing tier (or “bucket”) for transactions due to their impact on Sabre host resources. The first transaction is required to invoke Sabre’s Tripsearch functionality, and is unavoidable if trip-led pricing has been chosen as the booking path for an application. The second transaction, however, can be avoided and performed at the lowest Scan pricing tier with the assistance of the Bundled Sell utility. This would involve the caching of itinerary data within the application, where the actual selection is made. Rather than using the Tripsearch selection transaction, the application would then perform an independent Bundled Sell entry based upon the itinerary data.

A suggested use of the Bundled Sell utility to this end is as follows:

1. Application performs Tripsearch shop/qualifier transaction
2. Sabre returns results in the form of possible itineraries
3. Application caches possible itineraries
4. Possible itineraries evaluated
5. Selection made, application performs Bundled Sell transaction based upon cached information

Reducing Queries

Use of the Bundled Sell utility for the sale of air segments is not limited to Tripsearch. The Bundled Sell utility can be used at any time to sell multiple itinerary segments within a single transaction. Once the traveler’s desired itinerary is known, it is possible to sell an entire series of flights rather than build an itinerary piecemeal in a segment-by-segment fashion. Although single-segment sales are already priced at the lowest SabreScan transaction pricing tier, the reduction of multiple transactions to (ideally) a single transaction can produce significant cost savings and efficiencies over time.

Reduced Transaction Response Time

As single compound transactions sent to the Sabre host are generally more efficient than multiple transactions sent in series, it is possible to achieve a faster response from the host. A series of multiple transactions processed in serial fashion must be processed individually. Applications are forced to wait for a response from the host after each transaction before performing the next transaction. The Bundled Sell utility allows these transactions to be performed in a more parallel fashion. Reduced response times from the host generally result in reduced application response times for users.

Married Connection Logic and Bundled Sell

Married Connection Logic is programming in Sabre that allows air carriers to designate connecting seat inventory in specific markets as linked, or “married.” The linkage is reflected in the Sabre Passenger Name Record (PNR) itinerary at the time of sale, and remains a part of that itinerary as long as the itinerary segments in question are active in the record. This programming enforces the sale of connecting seat inventory to ensure that manipulation of that inventory, whether intentional or not, does not take place. It will inhibit the sale of invalid connections, and validate the availability of local market availability if itinerary segments are cancelled.

Married Connection Logic was implemented in Sabre as a way for air carriers to better manage inventory in a given market from origin to destination rather than on a point-to-point, leg-by-leg basis. Tighter control over the yield each seat sold in a market served by connecting flights can therefore be exercised rather than the piecemeal technique that had been used previously. For instance, inventory on an itinerary from San Francisco to Baltimore connecting in Denver could be managed from San Francisco to Baltimore rather than as two separate legs from San Francisco to Denver and from Denver to Baltimore. Use of the Bundled Sell utility allows applications access to inventory on the through-market basis, which may allow for access to inventory which may not be available on the local (point-to-point) market basis. This is due to the fact that connecting segments sold via the Bundled Sell utility are treated as sold on a through-market basis, with the married connection intact.

Use of the Bundled Sell utility will also aid in the avoidance of error responses from the host due to Married Connection programming. In the subscriber environment, error responses are returned when inventory in a market is unavailable or if an attempt is made to manipulate an itinerary in order to circumvent Married Connection Logic.

Note that Married Connection Logic only applies to itineraries involving connecting flights. Itineraries or portions of itineraries involving nonstop travel between origin and destination are unaffected by Married Connection Logic. Note also that not all carriers utilize married connection logic; a current list may be found in Sabre Format Finder under “MARRIED CONNECTIONS.”

A Special Note Concerning ARUNK (Arrival Unknown) Segments

At present, there is no special provision or functionality within the Bundled Sell utility to manage itinerary continuity issues arising from ARUNK segments.

An ARUNK (also noted as "ARNK") occurs within an itinerary when there is a break in the continuity of the order of origin and destination points within a given traveler's air journey. Sabre performs an edit at the time of the end transaction on all itineraries, and will provide an error response ("VERIFY ORDER OF ITINERARY SEGMENTS - MODIFY OR END TRANSACTION") if a break in that continuity is detected. Only the air itinerary and rail segments booked in the air segment format are edited in this manner. Other segments, such as car and hotel, are disregarded. Common situations where an ARUNK is required include open-jaw travel where the open jaw occurs at the destination, and travel which returns from an alternate airport within the same city as the destination. ARUNK segments are not required for itineraries where an open jaw occurs at the origin; there is no perceived break in the continuity of the air journey in that instance. ARUNKs added unnecessarily at the end of itineraries (known as "dangling" or "hanging ARUNKs") may actually cause intermittent problems with other Sabre host functionality.

A contrivance is available to acknowledge a break in continuity in the air journey within the Sabre host. This is known an "ARUNK" segment. In the Sabre itinerary, it is shown as "ARNK," and appears as its own independent numbered segment. No city code, dates, or times are associated with the ARUNK segment. The ARUNK segment can be inserted or reordered within the itinerary in a similar fashion to any other itinerary segment. ARUNKs are added to the itinerary using a variant of the Sabre sell command for air segments.

Though the error response resulting from omission of an ARUNK segment can be overridden with a second attempt to end the transaction, it is preferable to end the transaction successfully on the first attempt to preempt error response processing. It also avoids additional error responses encountered in future transactions performed within the same booking.

The ARUNK can be added to the itinerary in concert with the Bundled Sell utility in any one of several ways. None of these methods occurs within the Bundled Sell programming functionality; all require use of standard Sabre functionality. One method involves building continuity logic into the booking engine to insert the ARUNKs in sequence at the time of sale. Separate Bundled Sell transactions can be made for each continuous portion of the itinerary, separated by ARUNKs where required. Alternately, continuity logic can also be built to add the ARUNK segment in the appropriate location following completion of the sale of the entire itinerary via the Bundled Sell utility. The Sabre host command to be passed in this instance is "/xA", where x is the segment in the itinerary after which the ARUNK is to be inserted.

The third, and perhaps most expedient method involves selling the entire itinerary within one Bundled Sell transaction. Upon completion of the sale, and prior to end transaction, a host command can be executed which will automatically insert ARUNK segments at the appropriate point(s) within the itinerary. The Sabre host command to be passed is "0AA" (zero, A,A).

More information on ARUNK functionality can be obtained from the Sabre Format Finder help application.

Logical Records (LRECs)

A request from the client for fare-led availability functions consists of a data stream made up of Logical Records (LRECs) as defined in the World Fare Standard Interface Record (WFSIR). LRECs are essentially the next generation in the evolution of the Sabre MDR (Message Definition Record). The WFSIR contains LRECs defined for most existing World Fare functions as well as non-World Fare functions like PNR creation. This document only addresses the LRECs required to perform a Bundled Sell Utility transaction.

Helpful Hints for LRECs

The message block header always precedes the input LRECs, and “&&JA” typically precedes the message block header. The LRECs appear to Sabre as a long character string input message preceded by an “&&” indicating a GDS response is requested and a “JA” action code which is used to route the entry to the fares-led application. Responses from the host do not require a message block header. The ASCII symbol equivalent for the “&” character is [ALT+0134].

LRECs are positional and non-delimited. Where numeric values do not fill the allocated number of spaces, data should be right-justified within the field, utilizing leading zeros, except where otherwise noted. In contrast, alphanumeric fields are left-justified with trailing blanks (or placeholder characters).

When printing out or displaying LRECs with blank, inactive or omitted items, it is advantageous to use a non-alphanumeric placeholder such as the hyphen (-), caret (^), or tilde (~) rather than simply inserting a blank space or a period (.). Use of the hyphen, caret, or tilde is helpful when performing diagnostics as the input data is positional and non-delimited, and such characters are easier to count than a series of spaces or periods. Additionally, when printing out LRECs as hardcopy, it is recommended that a non-proportional font be used to facilitate the determination of character positions. Intermittent problems have been experienced by users inputting non-alphanumeric placeholder characters, however, and for coding purposes it is recommended to omit these characters in the data string transmitted to the host.

The Message Block Header and at least one Query Header are required with every WFSIR request.

Dates are input in the YYYYMMDD format, where 20040822 would represent August 22, 2004. Times are input in a 24-hour HHMM format, where 8:24AM would be represented as 0824 and 7:02PM would be 1902.

City codes are currently allocated five spaces throughout the LRECs, despite the fact that city codes are currently composed of only three alphabetic characters. The extra two characters are reserved for future expansion of city code information and must currently remain blank or be filled with a non-alphanumeric placeholder character. These two blank characters follow the three-letter city codes. Carrier codes, similarly, are currently allocated three spaces throughout the LRECS, even though carrier codes currently consist of only two alphanumeric characters. One character is held in reserve for future field expansion; the extra character trails and must remain blank or filled with a placeholder.

Note that certain fields within the LRECs may seem optional and if left blank do not appear to impact the functioning of the utility. In some instances, however, that otherwise seemingly extraneous data may actually be critical to the function of the LREC logic and should not be truncated or dropped. This is especially important with regard to the arrival date of each segment for a married connection (see note at end of the FL LREC section which follows).

XML Power Toolkit (PTK) users should take special action to ensure that the JA command is added to the Action Code Database within the Configuration Editor. This will ensure that the proper SDS AALCALENDAR MDR will be received. If the JA command is not added to the database, an unintelligible response from the SDSESAPIFRETXT MDR may be displayed.

INPUTS

Message Block Header (MBH)

The message block header follows the action code and serves as an envelope for all subsequent application LRECs. It is mandatory and consists of 6 characters:

Field	Length	Value/Meaning
Data Format Indicator	1	= 2 (message contains alphanumeric characters only)
Release Version	1	= 0 (initial release number)
Total Size of Data	6	Total size of message (excluding the prefix)

Following the message block header will be the LREC items, each of which is preceded with a nine-byte item header containing the size, ID, prime key, and unique key of the item. The order of the LRECs is not important. Some will be optional while others are required. *The exception to this is that if multiple Query Headers are input to Sabre, all of the LRECs that apply to a given Query Header **must** follow that header LREC*. The item sizes are subject to change in order to allow new data fields to be added to the end of the item. Therefore, any programs that read these logical record items must be written to ignore any new data at the end of an item that the program is not prepared to handle and must rely on the record length to skip to the next item. The specific uses of the prime key field in this interface will be explained on a per item basis below. The unique key serves as a sequentially assigned ordinal number for distinguishing between LRECs of the same ID. For example, if there were two date request LRECs, one outbound and one inbound, the first would have a unique key of 01 and the second a unique key of 02. The unique key also supports the establishment of relationships between the LRECs since a child LREC will point to its parent using the unique key of its parent.

Note that the total size of data indicated in the six-character-long field in the message block header does NOT include either the “&&” characters OR the “JA2000” which precedes the LREC input. LRECs will maintain a consistent number of characters related to the number of segments to be sold in one LREC transaction. The formula for the total size of the LREC is $80x + 56$, where x is the number of segments to be sold. The number of characters in the LREC is indicated below:

Number of segments	LREC Total Size of Data for MBH
1	136
2	216
3	296
4	376
5	456
6	536
7	616
8	696

Query Header LREC (QH)

The Query Header LREC is mandatory and tells Sabre what service is being requested and by whom. Some clients may not have access to all of the following data fields and will therefore leave those fields blank, allowing the host application to gather them from the settings in the host communication session if it exists. In addition to the Item Size and Item ID, the required fields (shown in **bold**) include the Application ID, the Transaction Code, the Originator Type, and the Client Type. All other fields are optional.

Field	Length	Value/Meaning
Item Size	4	= 0048
Item ID	2	= QH (signifies “Query Header”)
Prime Key	1	= 0
Unique Key	2	= 01 (only one supported at this time)
Application ID	1	= S (Segment Sell)
Transaction Code	2	= BS (for Bundled Sell)
User Location City	3	City Code
	2	*blank* (reserved for CRT loc. expansion)
User Location Country	2	International Country Code (not required; will default)
User Location Currency	3	International Currency Code (not required; will default)
Sabre Pseudo City Code	4	(may not be applicable for some users)
Corporate ID	5	(may not be applicable for some users)
CRS or Airline Code	3	Originating CRS or airline code (not required)
ARC/IATA Agent	9	ARC/IATA Agency ID Number (may not be applicable)
Agent Sine	3	Three alphanumeric characters (not required)
Originator Type	1	I = Internal CRS Location
Client Type	1	S = Sabre

*Note that that if multiple Query Headers are input to SABRE, all of the LRECs that apply to a given Query Header **must** follow that header LREC*. Note also that for developer customers, the Originator Type field and the Client Type field will always be I and S, respectively. Though other indicators exist, they are not for use by developers or subscribers.

Flight LREC (FL)

The Flight LREC is where the actual flight segments are sold. It is required for a Bundled Sell transaction when used in conjunction with a Tripsearch transaction. A single flight LREC will be present for each nonstop or direct flight segment in an itinerary; two LRECs will be present for each single connection, and three LRECs will be present for each double connection.

Field	Length	Value/Meaning
Item Size	4	= 0064
Item ID	2	= FL
Prime Key	1	= 0
Unique Key	2	Sequentially assigned ordinal
Date Request Link	2	Date request LREC number to which this item applies
	3	*blank* (not used by this application)
Board Point	3	Departure airport code
	2	*blank* (reserved for airport code expansion)
Off Point	3	Arrival airport code
	2	*blank* (reserved for airport code expansion)
Departure Date	8	YYYYMMDD
Departure Time	4	HHMM (24-hour clock)
Arrival Date	8	YYYYMMDD
Arrival Time	4	HHMM (24-hour clock)
Carrier Code	3	Two alphanumeric characters (one reserved for expansion)
Secondary Carrier Code	3	*blank* (not in use at present)
Flight Number	4	Up to four numeric characters
Class of Service	2	Up to two alphabetic characters
Equipment Code	3	*blank* (not in use at present)
Number of Stops	1	Number of stops or transit points

Flight LREC (FL) (continued)

The Prime Key, Unique Key, and Date Request Link fields are required in this LREC. The Prime Key field, though required, is not used in this LREC and must be held with a zero (null). The Unique Key field sequentially numbers the legs on a trip within the entire itinerary. The Date Request field sequentially numbers the legs within a given date, based upon date of departure. Note that a special provision exists for cases involving a connection with a change of date enroute; see the last part of this section for further details.

For example, for a simple itinerary from DFW to LAX, outbound on 1 March and returning on 8 March, the Prime Key, Unique Key, and Date Request fields would appear as follows:

Itinerary (segment/leg)	Prime Key	Unique Key	Date Request
DFW-LAX 1 March	0 (always 0)	01 (first segment for this series/itinerary)	01 (first request for this date)
LAX-DFW 8 March	0 (always 0)	02 (second segment for this series/itinerary)	01 (first request for this date)

For a single-carrier (marketing) connecting itinerary SEA to MIA connecting in both directions via ORD, outbound on 20 June and returning on 22 June, the Prime Key, Unique Key, and Date Request fields would appear as follows:

Itinerary (segment/leg)	Prime Key	Unique Key	Date Request
SEA-ORD 20 June	0 (always 0)	01 (first segment for this series/itinerary)	01 (first request for this date)
ORD-MIA 20 June	0 (always 0)	02 (second segment for this series/itinerary)	02 (second request for this date)
MIA-ORD 22 June	0 (always 0)	03 (third segment for this series/itinerary)	01 (first request for this date)
ORD-SEA 22 June	0 (always 0)	04 (fourth segment for this series/itinerary)	02 (second request for this date)

Flight LREC (FL) (continued)

For itineraries involving a single marketing carrier and a date change enroute (typically overnight flights, or “red eyes”), the date calculation for the Date Request field should be based upon date of origin for that portion of the itinerary. Do not treat connecting flights as operating on a different date for the purposes of the Date Request field in this instance. The Date Request field should be input as if the travel were occurring on the same date as the date of origin. For example, a connecting itinerary from PHX to BOS via DTW outbound late on the evening of 8 November, connecting on the morning of 9 November in DTW and returning on 12 November, the Prime Key, Unique Key, and Date Request fields would appear as follows:

Itinerary (segment/leg)	Prime Key	Unique Key	Date Request
PHX-DTW 8 November	0 (always 0)	01 (first segment for this series/itinerary)	01 (first request for this date)
DTW-BOS 9 November	0 (always 0)	02 (second segment for this series/itinerary)	02 (second request for this date*)
BOS-DTW 12 November	0 (always 0)	03 (third segment for this series/itinerary)	01 (first request for this date)
DTW-PHX 12 November	0 (always 0)	04 (fourth segment for this series/itinerary)	02 (second request for this date)

*despite the fact that the connecting flight actually operates on the next day following departure from the origin

Flights sold inconsistently with the method shown above may result in a successful sale but will be sold on a local basis (point-to-point) only and connections will not be married. If the desired inventory is not available on a local market basis, the result will be a failed sale indicated with an error response.

Flight LREC (FL) (continued)

For itineraries involving multiple marketing carriers where no married connection exists, the Date Request Link must be numbered in a slightly different manner. This numbering sells these legs as independent (unmarried) segments. Segments on differing carriers involving unmarried segments (whether connecting flights or not) inadvertently sold with an incorrect Date Request Link may fail and in turn cause other segments within the itinerary to fail. This failure is caused by incorrect instructions to the carrier to sell the legs as married when they are actually not. The sale fails because the inventory and validity of the itinerary cannot be verified from origin to destination for those segments. (Common examples of this kind of itinerary include flights on two different domestic carriers sold in the same itinerary, and international flights where a second carrier is used beyond the international port of entry. There are a variety of other instances where this might occur.) For example, an itinerary from PHL to MAD on 20 May on Carrier A, with a connecting flight on carrier B from MAD to BCN on 21 May, returning on 29 May on Carrier B from BCN to MAD, connecting to Carrier A on the same day from MAD to PHL, the Prime Key, Unique Key, and Date Request fields would appear as follows:

Itinerary (segment/leg)	Prime Key	Unique Key	Date Request
PHL-MAD 20 May [carrier A]	0 (always 0)	01 (first segment for this series/itinerary)	01 (first request for this date and carrier)
MAD-BCN 21 May [carrier B]	0 (always 0)	02 (second segment for this series/itinerary)	01 (first request for this date and carrier)
BCN-MAD 29 May [carrier B]	0 (always 0)	03 (third segment for this series/itinerary)	01 (first request for this date and carrier)
MAD-PHL 29 May [carrier A]	0 (always 0)	04 (fourth segment for this series/itinerary)	01 (first request for this date and carrier)

Note that in this example, all Date Request Links are shown as the first request for that date, despite the actual date of travel. This indicates that no segment marriage exists, and that each segment is to be sold independently and without reference to other segments.

SPECIAL NOTE REGARDING INCLUSION OF DATA FIELDS: In some instances, data fields may seem optional and if left blank do not appear to impact the functioning of the utility. In many instances, however, that seemingly extraneous data is critical to the function of the LREC logic and should not be truncated or dropped. Doing so may cause a failed sale, an unexpected result, or an error response. This is especially important with regard to the arrival date of each segment for a married connection. The Bundled Sell utility uses the arrival date of flight segments to properly sequence those segments for married connection logic validation in conjunction with the Date Request field in the FL LREC.

Related Product Information LREC (SP)

The Related Product Information LREC is essentially the sell status of the flight, and *is **required** whenever a flight is to be sold via the **Bundled Sell Utility***, one per Flight (FL) LREC. On input, the IATA code must be set to “NN” to sell seats on a flight. (“LL” may be used to waitlist a flight, but that option is not commonly used.)

Field	Length	Value/Meaning
Item Size	4	= 0016 (*see note below)
Item ID	2	= SP
Prime Key	1	= 0
Unique Key	2	Sequential number of this SP LREC
Reference	2	Reference number
Number in Party	3	Total number of passengers

IATA Status/Action Code	2	IATA action/advice/status code
		NN = need request
		LL = waitlist

Note that although this LREC is technically 19 characters long , the Bundled Sell input should consist of only the 16 characters shown above. An additional three characters (not shown in the table above) representing the segment number of the legs in the itinerary exist at the end of this LREC. For input purposes, the LREC is truncated following the IATA Status/Action Code as segments are not numbered upon input, unlike the output version of this LREC shown on page 15 of this document. This field is not populated until the sale is accomplished.

Note that the “Unique Key” and “Reference” fields in the above LREC do not correspond to fields in other LRECs (specifically the FL LREC). The Unique Key field in the SP LREC is the sequential ordinal number of the SP LRECs in this transaction. The Reference field links this SP LREC to its “parent” LREC within the transaction.

Testing LRECs Using the Sabre Host

As the LREC inputs are essentially Sabre entries, they can be tested using the standard Sabre for Windows emulator or the Sabre eVoya Webtop. Sample entries can be tested in the host simply by signing in to Sabre and making the subsequent input entry, starting with “&&JA.” A full example entry can be found on page 17 of this document. It should be noted that the flight numbers, dates, times, and other variable information used in a test entry must be actual valid flight information. The sale of the itinerary which results should be ignored in the host immediately following the entry to prevent the inadvertent misuse of airline inventory. Successful inputs will result in a “CALENDAR” response from the host; the subsequent itinerary can then be displayed with the *A entry. Unsuccessful entries will result in an obvious error response and no itinerary segments present in the Sabre work area.

RESPONSES (OUTPUTS)

A successful sale attempt will return a Sabre Data Stream (SDS) response in the form of the SDSAALCALENDAR Message Definition Record (MDR). Contained within this MDR will be the details of the sale, including an SP LREC for each segment sold, potentially followed by up to one SD LREC for each segment. For additional information regarding the SDSAALCALENDAR MDR and the elements where this information is returned, consult SDS documentation or the SDS Database available at http://sds.sabre.com.

Related Product Information LREC (SP)

The Related Product Information LREC will be output for each flight leg in response to a bundled sell request, and it will show the status of the sale. For example, a status/action code of "SS" will indicate a successful sell, but a "NO" indicates a problem that prevented the seats from being sold. This is the same LREC which is used as an input. See the **Inputs** section for a complete description of the **Related Product Information LREC.**

Field	Length	Value/Meaning
Item Size	4	= 0019
Item ID	2	= SP
Prime Key	1	= 0
Unique Key	2	Sequential number of this SP LREC
Reference	2	Reference number
Number in Party	3	Total number of passengers

IATA Status/Action Code	2	IATA action/advice/status code
		SS = segment sold
		NO = no action, sale not completed (UNSUCCESSFUL)
		UC = unable to confirm (UNSUCCESSFUL)
		US = unable to sell (UNSUCCESSFUL)
		LL = waitlist (received only in response to LL request)

Segment Number	3	Segment number of leg in itinerary (sequence of sale) (only shown upon output)

Sell Details LREC (SD)

The Sell Details (SD) LREC will also be output from the bundled sell utility, and one will exist for each Flight (FL) LREC. The reference number in the SD LREC will correspond to the unique key value in the parent FL. The SD LREC is only received as a response; there is no input for this LREC. Note that the SD LREC will only be returned when relevant information is contained within the LREC. If there is neither an Electronic Ticketing Eligibility Indicator (shown as "Electronic Ticket/Electronic Ticket Descriptor" below) nor married connection logic indicators, the LREC may not be returned.

Field	Length	Value/Meaning
Item Size	4	= 0026
Item ID	2	= SD
Prime Key	1	= 0
Unique Key	2	Sequential ordinal number
Reference	2	link to parent FL LREC
Flight Type	2	Flight type indicator
Electronic Ticket	2	Electronic ticket descriptor EN = not electronic ticket candidate ET = electronic ticket candidate

Polled	3	Polled/non-polled
		P = polled
		N = non-polled
		ACK = acknowledgement

Marriage Control	2	M = flight is old marriage grouping
		B = non-dominant flight
		C = potential marriage candidate

Marriage Grouping	1	Marriage flight grouping A = married connection
Group Number	3	Marriage group number
Sequence Number	2	Marriage sequence number

Message Information LREC (MI)

The Message Information LREC will be sent when an error condition has occurred. In the future, it may also be used to output non-error information such as endorsement or restriction messages. The MI LREC is only received as a response; there is no input for this LREC.

Field	Length	Value/Meaning
Item Size	4	= 0020 (with no free text; length may vary based upon additional free text at end of message)
Item ID	2	= MI
Prime Key	1	= 0
Unique Key	2	Ordinal number (i.e. nth MI LREC)
Reference	2	link to parent LREC causing error

Message Type	1	E = error message
		W = warning message
		D = display message
		N = endorsement message
		R = restriction

Message Number	4	*Bundled sell range (1 - 999)* 1 = Scheduled Times Differ from Booked Flight 100 = Invalid Board Point (departure point) 101 = Invalid Off Point (arrival point) 102 = Invalid Departure Date 104 = Invalid Class of Service 107 = Invalid Carrier Code 114 = Invalid Flight Number 117 = Schedule Change in Progress 118 = System Unable to Process 120 = Invalid Action Code 121 = Invalid Number in Party 130 = Invalid Flight for This City Pair 133 = FLIFO Exists for This Flight 293 = Unable 00 Available 309 = Received From Data Invalid 311 = Ticketing Arrangement Data Invalid 314 = Name Element Data Invalid 317 = Contact Element (Phone) Invalid 411 = Flight Not Operating for this Flight/Date (Noop) 418 = Flight Departed 421 = Unable 00 Available, Waitlist Closed 437 = Carrier Must be in Separate PNR 438 = Data Beyond 180 Days Not Available 439 = Waitlisting Not Allowed
Length of Free Text	4	Length of additional free text message
Free Text Message	Varies	Free text message

LOGICAL RECORD DEFINITION

The LRECs presented previously are displayed in a compressed, summarized format for clearer illustration.

Message Block Header (MBH)

Data format indicator	Release version	Data length
2	0	......

Query Header LREC (QH)

LREC Size	LREC ID	PK	Uniq Key	Appl ID	Trans Code	CRT City	CRT Country	Cur. Code	Pseudo City	Corp ID	CRS / Airline	ARC/ Agent	Agent Sine	Orig. Type	Client Type
0048	**QH**	0	01	.	JA	.....	..	...	....	.....	...	....	...	D	.

Flight LREC (FL)

LREC Size	LREC ID	PK	Uniq Key	DT Link	Blan k	Board City	Off City	Departure Date	Depart Time	Arrival date	Arrive Time	Carrier Code	Blank	Flight Number	Booking Class	Equipment Code	Number of Stops
0064	**FL**	0	..	..	...	.....	.....	....	....	....	....	...	...	....	..	...	.

Related Product Information LREC (SP)

LREC Size	LRECID	PK	Unique Key	FL link	Nbr. in Party	IATA code
0016	**SP**	0	..	..	...	..

Message Information LREC (MI) [RESPONSE ONLY]

LREC Size	LRECID	PK	Unique Key	Reference	Message Type	Message Number	Message Length	Message Text
0020+	**MI**	0	..	..	.	....	..	varies

Sell Details LREC (SD) [RESPONSE ONLY]

LREC Size	LREC ID	PK	Uniq Key	FL link	Flight Type	Electronic Ticket	Polled	Marriage Control	Marriage Grouping	Group Number	Sequence Number
0026	**SD**	0	..	..	..	..	...	..	.	...	..

SAMPLE TRANSACTION DIAGRAMS

The customer selects flights from the displayed options. The application sends a Flight (FL) LREC and a related product information (SP) LREC for each flight in the connection to be sold, or one of each LREC if it is a direct flight. The following reflects the sale of a round-trip journey with nonstop travel in each direction (i.e. two flights).

Query Example 1:

&&JA	MBH	QH	FL	SP	FL	SP

The host application returns the status of the sell in the SP and returns additional information about each flight in the sell details (SD) LREC, if information is available.

Response Example 1:

FL	SP	SD*	FL	SP	SD*

In the next example, flights selected are single-connecting, which means that there is a single connection in each direction, for a total of four segments. This time, four FL LRECs must be generated:

Query Example 2:

&&JA	MBH	QH	FL	SP	FL	SP	FL	SP
FL	SP							

In this instance, four conjoined FL, SP, and SD LRECs are returned by the host:

Response Example 2:

FL	SP	SD*	FL	SP	SD*	FL	SP	SD*
FL	SP	SD*						

*The SD LREC will be returned as part of the response only if information is contained within that LREC (e.g., Electronic Ticketing Eligibility or Married Connection Data). The SD LREC may not be transmitted as part of a response to every sale transaction.

SAMPLE INPUT DATA STRINGS

The following four-segment, round-trip itinerary is used in this example to produce a Sample Data Input String:

American Airlines flight 754, 1 May 2002
departs Los Angeles (LAX) 10:33AM, arrives Chicago O'Hare (ORD) 4:38PM

American Airlines flight 154, 1 May 2002
Departs Chicago O'Hare (ORD) 5:41PM, arrives Miami (MIA) 9:34PM

American Airlines flight 908, 8 May 2002
Departs Miami (MIA) 7:30AM, arrives Dallas/Fort Worth (DFW) 9:42AM

American Airlines flight 2421, 8 May 2002
Departs Dallas/Fort Worth (DFW) 11:04AM, arrives Los Angeles (LAX) 12:24PM

The desired itinerary is booked for one seat in "M" inventory.

The actual input data string for the Bundled Sell Utility sent to Sabre from a client would appear as follows (blanks are illustrated with the tilde (~) character):

```
&&JA200003760048QH001SBS~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~IS0064FL00101~~~LAX~~ORD~~200205011033200205011638AA
~~~~0754M~~~~~0016SP00101001NN0064FL00202~~~ORD~~MIA~~2002
05011741200205012134AA~~~~0154M~~~~~0016SP00202001NN0064FL0030
1~~~MIA~~DFW~~200205080730200205080942AA~~~~0908M~~~~~0016
SP00303001NN0064FL00402~~~DFW~~LAX~~200205081104200205081224AA~
~~~2421M~~~~~0016SP00404001NN
```

(continued next page)

Sample Input Data Strings (continued)

This data string can be dissected into its component parts:

&&JA
Action code

20000376
Message Block Header

0048QH~01SBS~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~IS
Query Header

0064FL~0101~~~LAX~~ORD~~200205011033200205011638AA~~~~0754M~~~~~
First Flight (FL) LREC (requests AA754 LAX-ORD)

0016SP00101001NN
First Related Product Information (SP) LREC for first FL LREC

0064FL~0202~~~ORD~~MIA~~200205011741200205012134AA~~~~0154M~~~~~
Second Flight (FL) LREC (requests AA154 ORD-MIA)

0016SP00202001NN
Second Related Product Information (SP) LREC for second FL LREC

0064FL~0301~~~MIA~~DFW~~200205080730200205080942AA~~~~0908M~~~~~
Third Flight (FL) LREC (requests AA908 MIA-DFW)

0016SP00303001NN
Third Related Product Information (SP) LREC for third FL LREC

0064FL~0402~~~DFW~~LAX~~200205081104200205081224AA~~~~2421M~~~~~
Fourth Flight (FL) LREC (requests AA2421 DFW-LAX)

0016SP00404001NN
Fourth Related Product Information (SP) LREC for fourth FL LREC

GLOSSARY

Carrier
Airline.

Codeshare
Marketing agreement between carriers which permits one carrier to offer flights on another distinct carrier. A codeshare agreement usually allows one carrier to place its two-character IATA code and a secondary flight number on a given flight in order to sell the inventory as if it were its own. Codeshare arrangements are usually identifiable by flight number range, and a variety of indicators in Sabre displays. Codeshare arrangements involve an operating carrier and one or more marketing carriers. Common examples of codeshare arrangements involve a carrier and its associated regional affiliates (e.g., Delta Air Lines and Comair), or carriers involved in marketing agreements or alliances with airlines operating networks in countries other than their own (e.g., American Airlines and Japan Airlines) Flights may have multiple secondary flight numbers, but the Bundled Sell Utility is primarily concerned with the marketing carrier in a given situation. Tripsearch, Bargain Finder Plus, and other associated Sabre Air Pricing tools display information and process queries primarily based upon marketing carrier.

Host
The Sabre mainframe computer, located in Tulsa, Oklahoma.

IATA
See International Air Transport Association.

International Air Transport Association
International trade association for airlines. Counts responsibility for setting standardized communications codes for carrier use among its various responsibilities. More information about IATA is available at www.iata.org.

Interline
Involving two or more distinct carriers. Typically applied in description of an itinerary, most often complex in nature.

Itinerary
The journey (whether speculative or confirmed) that a traveler may take. Itineraries are composed of segments. In Sabre, the itinerary is one of the basic parts of the reservation, or PNR.

Logical Record (LREC)
A component of the request sent to host Sabre to initiate the various functions described in this document. LRECs are also returned as parts of the Message Definition Record (MDR) of the SDS (Sabre Data Stream) response to such requests.

LREC
See Logical Record.

(continued next page)

GLOSSARY (continued)

Marketing Carrier
The carrier whose two-character IATA code is associated with a flight number for sale; i.e., the carrier who is selling that inventory. Used in context of a codeshare flight. Contrasts with Operating Carrier, which is the carrier whose aircraft and crew actually operate a given flight. See also codeshare.

Married Connection Logic
A methodology for air carriers to better manage inventory in a given market from origin to destination rather than on a point-to-point, leg-by-leg basis. Tighter control over the yield each seat sold in a market served by connecting flights is therefore exercised--in contrast to the previous piecemeal techniques used previously.

MDR
See Message Definition Record.

Message Definition Record (MDR)
The response from a Sabre Data Stream (SDS) request. An MDR may be composed of several components, or elements. In the context of the Bundled Sell Utility, the MDR encases several LRECs as parts of its response.

Native Sabre
Term used to describe format-driven human interaction with Sabre host via a terminal interface. Native Sabre is the most common form of interaction with Sabre in the travel agency environment.

Operating Carrier
The carrier who whose aircraft and crew physically operate a given flight. Contrasts with Marketing Carrier, which is the carrier who offers the flight for sale using its two-character IATA code but may not operate or crew the actual aircraft. See also codeshare.

Passenger Name Record (PNR)
Abbreviation for Passenger Name Record. PNR is the generic acronym for a reservation in a computer reservations system (CRS), whether it be Sabre or another reservations system.

PNR
See Passenger Name Record.

Sabre Data Stream (SDS)
Sabre's structured data product which communicates with/via the Sabre host. Formerly known as Generalized Data Stream, SDS provides an alternative to screen-scraping host responses from the standard native Sabre environment.

Scan
Formerly known as SabreScan, Scan is a system of tracking and pricing transactions (known colloquially in this context as "hits") sent to Sabre based upon resource utilization. Scan ranks and prices transactions in three tiers (or "buckets"): standard (lowest), pricing, and search (highest).

(continued next page)

GLOSSARY (continued)

SDS
See Sabre Data Stream.

Segment
Single component of an itinerary. In this instance, segments are individual flights within a complete itinerary. Also known as leg; air segments are commonly referred to as "flight segments." Segments are numbered in the body of the reservation record (PNR).

Transaction
In the context of this document, the term "transaction" refers to a single message sent to the Sabre host. (Note that the term "transaction" has a different meaning in different contexts.)

Trip-led Pricing
See Tripsearch

Tripsearch
A Sabre air itinerary shopping process that performs a search for possible acceptable itineraries based upon qualifiers input by a user or application. The Tripsearch process involves two transactions. A shop/qualifier transaction initiates the shopping algorithm while simultaneously providing the parameters of the search; Sabre then returns a list of possible itineraries based upon the criteria provided. Following evaluation of those possible itineraries, a selection transaction is made which chooses an option from the list of results. Formats related to Tripsearch functionality in native Sabre begin with the letters "JR".