BagMessage

Interface Description

SITA Airport & Desktop Services

http://www.sita.aero

Table of Contents

CONFIDENTIALITY AND COPYRIGHT NOTICE

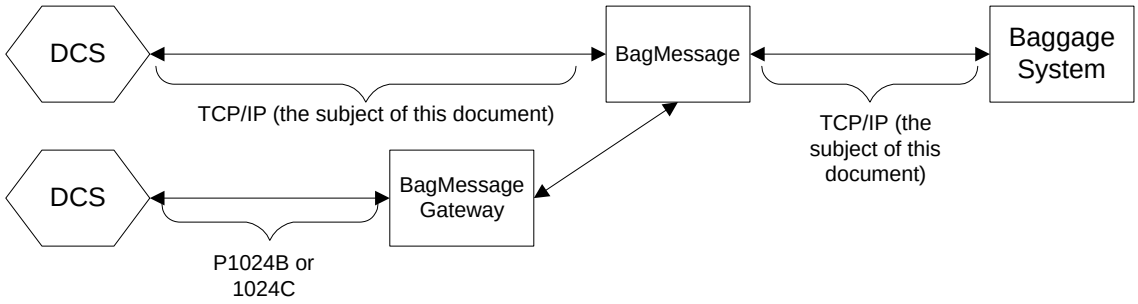
SITA Airport & Desktop Services
1 London Gate
252 – 254 Blyth Road
Hayes
Middlesex
UB3 1BW
United Kingdom

Revision History

Revision	Date	Change	Author
1.01	31 AUG 97	References to STX/ETX outside serial link section removed. Clarification of ADR sending at BSU logon	JK
1.02	1 NOV 1997	Info to/from vendor added	
1.03	10 NOV 1997	Clarified that address stamp usage only happens with bi-directional mode	JK
1.04	10 DEC1997	Minor formatting and scope changes Clarifications to message structure "Message ID's" replace "sequence numbers"	JK
1.04a	15 April 1998	Clarification on use of Message ID when NAK received	JK
1.04b	30 July 1998	Clarification of "normalisation" Clarifications in Address message usage Clarifications on BPMs added	JK
1.04c	01 SEP 1998	Clarifications on BPMs added	JK
1.04c1	16 SEP 1998	"??" removed from 6.1.3 and sentence added about ADR not being put in store file in full BagMessage implementation (1999)	JK
1.05	27 NOV 98	Add dual TCP/IP connections for BSMs and BPMs	JA
1.06	30 NOV 98	Add note that dual TCP/IP connection must be configurable and able to use a common single address	BG
1.07	28 JAN 2000	Reworded TCP/IP status section to clarify that BOTH BagMessage and the Client needs to send the status message every 30 seconds. Added the option of the baggage system to become a TCPIP Server and multiple connections for the same port.	ML
1.08	4 June 2000	Clarifications of TCP/IP protocol	SF
1.08	August 2001	Updated for new Brand naming to BagMessage	HM
1.09	April 2002	Removal of obsolete connectivity types.	HM
1.10	Dec. 2002	(a) Re-wording of appid information; (b) added Section 8 – Information from the Customer.	MC
1.10d	June 2004	Addendum to clarify use of separate connection for BSMs and BPMs (sections 5.1.3.4 and 6.10). and immediate requirement for ACK. Reference to recent edition IATA PSC Manual updated.	JK/RF
1.10e	June2005	Review, clerical edits	RF
1.10f	May2007	Review, clerical edits	RF
1.10g	Aug2007	Update to references, clerical edits	RF
1.10h	Oct 2009	Updating Section 4.1, 4.2, 5.1.3.4 and removed Error reporting.	Peter Drummond

1 Scope

The interface described in this document is the interface between the SITA BagMessage service and the Baggage System as well as the TCPIP interface between BagMessage and the airline DCS, as diagrammed below. Other interfaces, such as that between the BagMessage gateway and airline DCS using P1024B/C are discussed in other documents.



1.1 References

1. IATA/ATA BWG, Recommended Practice 1745 - Baggage Service Messages, Passenger Services Conference Resolutions Manual 27th Edition, 2007.

2. IATA/ATA BWG, Resolution 740 - Form of Interline Baggage Tag, Passenger Services Conference Resolutions Manual 27th Edition, 2007.

3. IATA/ATA BWG, Recommended Practice 1797b - Baggage System Interface (BSI), Passenger Services Conference Resolutions Manual 27th Edition, 2007.

1.2 Authority

This document is issued under the authority of Airport and Desktop Services Division of SITA.

1.3 Glossary of Terms and Abbreviations

ACK	Acknowledgement message
BSI	Baggage System Interface. See Ref #2.
BSM	Baggage Source Message. . For definition see Ref #1.
BUM	Baggage Unload Message. . For definition see Ref #1.
BPM	Baggage Processed Message. . For definition see Ref #1.
BSU	Baggage System Unit. The computer controlling a baggage handling system such as a sortation or a reconciliation system.
BagMessage	SITA's Global Message Distribution Service
SITA	Société International de Télécommunications Aéronautique
CUTE	Common Use Terminal Equipment
DCS	Departure control System. The airline system that generates and may receive baggage messages
IATA	International Air Transport Association
IP	Internet Protocol – Communications Protocol
P1024B	A legacy communications protocol used from airline DCS to BagMessage
P1024C	A legacy communications protocol used from airline DCS to BagMessage
TCP/IP	Communications Protocol
SITA Type 'B'	Communications Protocol
HSRP	Hot Standby Routing Protocol
VDU	Visual Display Unit

2 Introduction

The purpose of this document is to specify the interfaces between SITA's BagMessage and attached baggage systems, and the interface between SITA's BagMessage and airline DCS systems. The document is intended for distribution to baggage system vendors and airlines to enable them to implement the interface in their baggage system or airline DCS.

SITA's BagMessage Service is a system that enables one or more airline DCS to exchange baggage messages with one or more automated baggage systems.

To achieve this objective, BagMessage is able to:

- Send and receive all baggage messages (including, but not limited to BSM, BPM, BUM) from users, via one or more of the standard data transmission methods employed by the user community
- Process baggage messages in order to remove communications addressing and line protocol control characters.
- Send and receive baggage messages from baggage systems via fast and secure links.

BagMessage allows airline host computers to send Baggage Messages (BSM) to an automated baggage system which processes baggage tagged with an IATA ten digit sortation/identification bar coded tag numbers.

The interfaces described in this document:

- Receives BSMs from multiple carrier host systems
- Multiplexes all received BSMs onto a single connection to a baggage system
- Uses a TCP/IP message interface as a standard interface to baggage systems
- Supports the two-way exchange of messages necessary for advanced baggage handling and baggage reconciliation
- Describing the TCP/IP connectivity between BagMessage and airline DCS.

3 BagMessage System Overview

SITA's BagMessage provides a bridge between airline communication systems, and automated airport systems, especially baggage systems. In doing so, it complies fully with the IATA License Plate concept for baggage handling and IATA *RP 1797b, Baggage System Interface*.

Figure 1 illustrates the basic function of BagMessage.

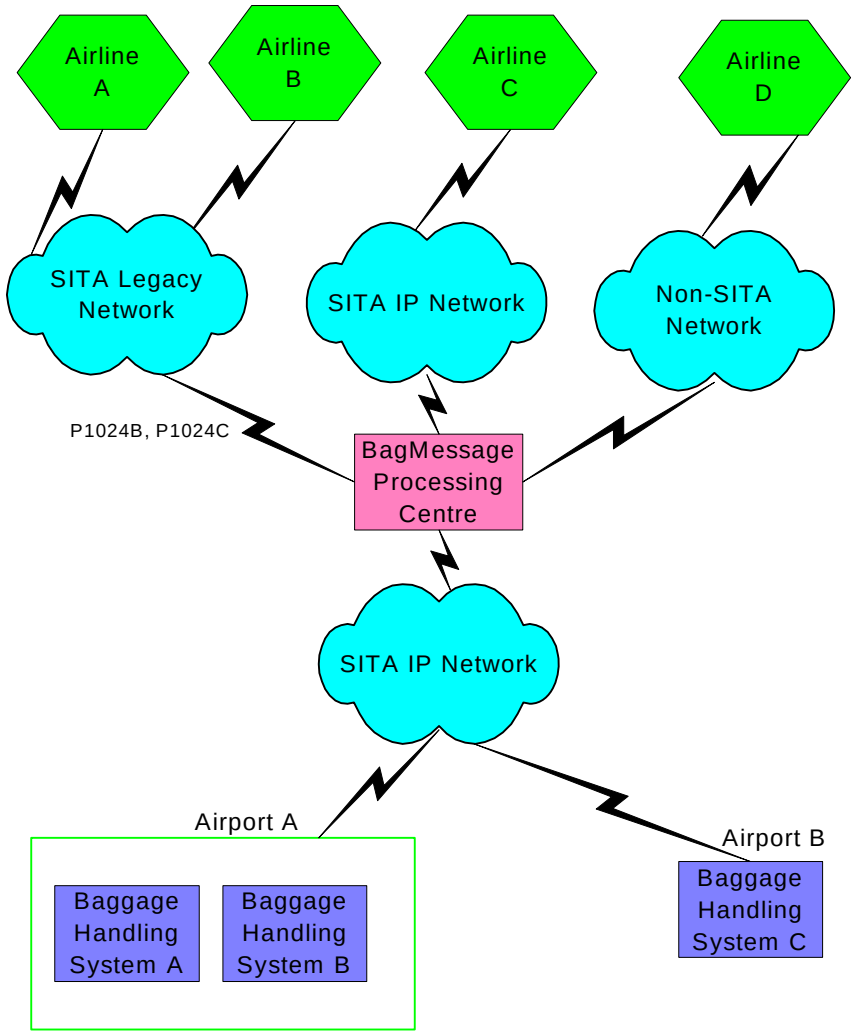


Figure 1

BagMessage provides an interface to the air transport industry's data transport network via the SITA Network, or via direct lines to a carrier's host.

BagMessage provides a TCP/IP interface to the airport's baggage system.

BagMessage software includes the following functions:

- Connectivity to carrier host systems via P1024B, P1024C, or TCP/IP connections
- Number of connections limited only by installed hardware
- TCP/IP connectivity between BagMessage and baggage system with a LAN-based, TCP/IP connection

- Storage of BSMs if the connection between BagMessage and baggage system fails
- Message acknowledgment between BagMessage and carrier host systems
- Support for both VDU and Printer host connections
- 'Normalization' of incoming host messages
- Full operator's console and configuration utilities
- Support for redundant configurations, given redundant hardware
- Reporting of BagMessage and host line status to baggage system using standardized messages (optionally configurable).
- Bi-directional communication enabling two-way communication between carrier hosts and baggage systems, for baggage reconciliation and baggage systems requiring two-way communications.

BagMessage functionality may be logically divided into three sections (see Figure 2):

- Airline Host to BagMessage Communications
- Internal BagMessage Processing
- BagMessage to Baggage System Communications

This document concentrates on the third area - the communications interface to the Baggage System. However, some aspects of the BagMessage processing of messages is useful knowledge for interface developers and is covered in section 4.

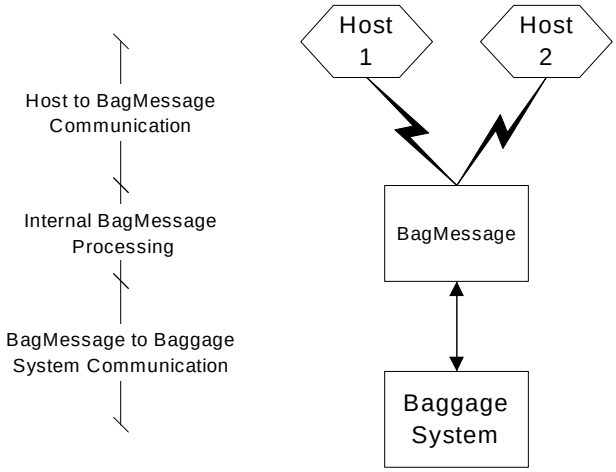


Figure 2

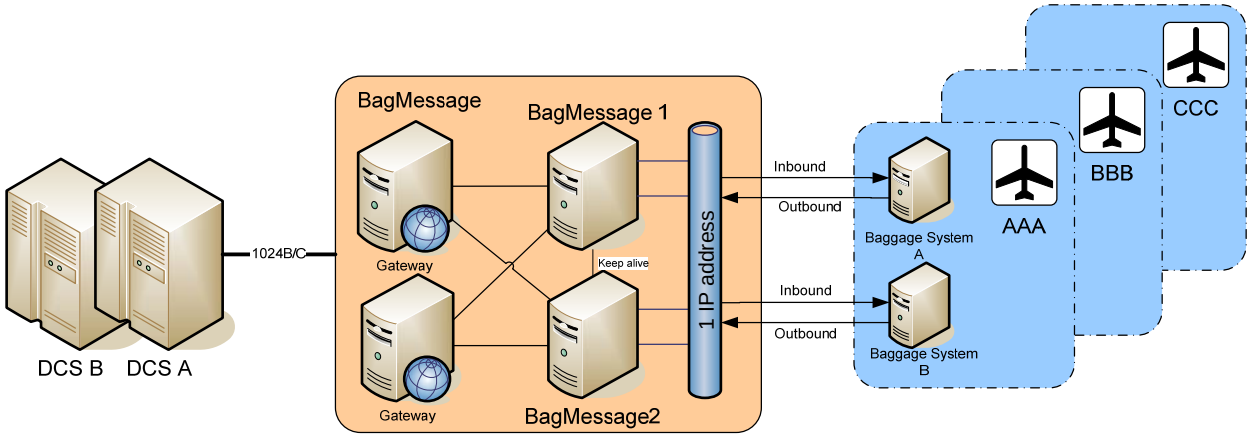
4 BagMessage Operation and Processing

4.1 Redundancy

BagMessage is operated in a redundant configuration to add security to the host-to-baggage system connection, in the event of a system failure the secondary server will take over processing of Messages which is transparent to the client connecting.

BagMessage Server failure is determined in one of three ways:

- When one of the BagMessage Servers in a redundant pair is initialised and does not see the second BagMessage Server for a defined time period.
- When both BagMessage Servers are operational and one Server fails to communicate with its peer for a defined time period.
- When both BagMessage Servers are operational and one BagMessage Server reports its delivery sub-system has failed.



Any particular DCS will be assigned a primary path to the Baggage System. In the event that the primary path fails, for any reason, the data from and to the DCS will be routed through the alternate path.

The 'Keep Alive' connection is a network TCP/IP connection between the redundant BagMessage Servers. This connection is used to control the routing of baggage messages over the primary or alternate path.

4.2 Redundant Operation

The following section describes how redundant operation works in practice for a real and complex baggage system connection.

The general philosophy of BagMessage is that control of duplicate and backup links is maintained by the BagMessage Servers and not require intelligence or decision making in the connected baggage systems.

BagMessage now employs High availability software that manages the IP addressing of the servers. In this case most baggage system do not need to maintain two connections (traditionally known as either load sharing and or Hot standby)

The Baggage System computer opens a session to the Bag Message System IP address and port supplied by SITA, an ADR message can be sent indicating all the airline data will be sent on this link. The appropriate BSMs will now be sent down this link.

It should be noted that unless the baggage system computer is using bi-directional communications, the ADR message will generally be disregarded.

The BS[P] computer can (note this may not be required due to the high availability software maintaining the IP address) opens a second session with BagMessage Disaster recovery server. Again, BagMessage system examines its configuration: if configured for hot standby, this connection is a backup link and BSMs will not be sent on this backup connection. An ADR message is sent once and status messages are sent regularly on the backup link.

In the event of a network or Server failure baggage data is still delivered. The BagMessage system will fail over to a secondary machine and restores connectivity then starts delivery of BSMs. If it is unable to deliver messages due to network failure or Baggage system failure, no more messages are delivered and BagMessage Server will store the BSMs to disk. It will await re-opening of the links by the baggage system computer.

4.3 Envelope Removal

BSMs and other messages received from airline DCS systems over a legacy protocol such as P1024B/C will be contained in the protocol's particular envelope. The BSM message is removed from the protocol envelope.

For example, in a P1024C connection, the BSM will be contained in a P1024 protocol envelope:

<RID> <SID> <DID> <STX> <BSM Message> <ETX>

In this case, the message contents, <BSM Message>, is extracted from the protocol envelope.

In general, BagMessage does not process BSMs. In particular, BSMs for multiple tag printing are not translated into multiple BSMs. All BSM processing should be done in the baggage system according to local requirements.

BagMessage can pass any message which conforms to the following format:

<Message Header> < Message Content> < Message Trailer>

Message Header: a three-character string denoting the start of a baggage message. Example: 'BSM'. BagMessage is configured with a 'valid message header table', which contains the set of acceptable three character strings which signal the beginning of a valid message.

Message Content: BagMessage, in general, ignore the contents of a message. For BSMs, however, BagMessage parses the message to count number of message received from each host, the number of characters received from each host, and the number of bags represented by the data contained in the BSM.

Message Trailer: a six-character string denoting the end of a baggage message. The format of the message is: 'END' < Message Header>

Example: 'ENDBSM'. For every Message Header there must be a corresponding Message Trailer.

4.4 Baggage Message Storage

If communication is broken for any reason between BagMessage and the baggage system, BagMessage stores the BSMs destined for each baggage system, awaiting re-establishment of the link.

If the communication breakdown between BagMessage and the baggage system persists for more than a configurable time period, BSMs are discarded on a first in/first out basis.

When communication is restored, BagMessage begins to transfer BSMs to the Baggage Sortation system on either a first in/first out or last in/first out basis. The order is configurable for each baggage system attached to BagMessage. Stored messages are handled on a priority basis - that is, the transfer of stored messages does not block the timely transmission of current BSM messages.

5 BagMessage to Baggage System Communications

The functions described in this section are independent of the protocol used between BagMessage and the Baggage System.

5.1 Bi-directional Communications

If bi-directional communication is used, typically for BSUs sending BPMs, additional messaging is required between BagMessage and baggage system in order to coordinate de-multiplexing of messages in redundant configurations. For this purpose, the IATA RP 1897b for BSI describes the Address Stamp Protocol. SITA's BagMessage supports this protocol, which is described below. The address messages described are only generally of significance when the baggage system wishes to use the bi-directional feature to send data back to an airline DCS (or other application, such as tracking) in a redundant BagMessage configuration.

5.1.1 Address Stamp

Only if bi-directional communications is configured, messages exchanged between BagMessage and baggage systems have an address stamp affixed to the beginning of the message indicating the host connection (carrier) from which the message was received, or the carrier to which the message is addressed.

Address stamps:

- Are two to four characters in length
- Contain the two or three character airline designator, as defined in IATA Resolution 762, plus any additional characters as needed up to the maximum four characters.

Construction	**Example**	**Format**
Address Stamp	SR1	MM(MM) <≡
Standard Message Id	BSM	AAA<≡
End Of Message	ENDBSM	AAAAAA<≡

The extra character(s) in the address stamp field can be used to distinguish between multiple connections to the same carrier. For example, SR might have two separate connections to its host over different paths. A numeric character can be used to label the two SR connections, SR1 and SR2.

The address stamp 'BSI' is reserved to designate messages originating in BagMessage itself.

Note that the address stamp is not part of any message sent or received from a carrier's host DCS system. Address stamps are only part of the protocol between BagMessage and a baggage system. Address stamps are added to baggage messages by BagMessage before being transmitted to the baggage system. When responding to a baggage message, a baggage system will add the same address stamp that was affixed to the message being responded to. BagMessage will remove address stamps from baggage messages received from baggage systems before transmitting the message to the host indicated by the address stamp.

The purpose of the address stamp is to allow BagMessage to route messages received from baggage systems to the proper destination. Note that without an address stamp, the destination of a message received from a baggage system cannot be determined.

The address stamps used for each host channel are configurable through the BagMessage configuration utility. It can be any 2-4 alpha-numeric string, but it is recommended that the alphanumeric airline designator be used as the basis of the address stamp string.

5.1.2 Address Stamps and Channel Assignments

In a redundant configuration BagMessage is connected to the baggage system via two independent channels (connections). To indicate to the baggage system which address stamps are valid on a particular connection, BagMessage sends an 'ADR' message to the baggage system. Note that this only happens if bi-directional communications are configured. The ADR message received on a specific channel contains all the address stamps which are valid for that channel. A new ADR message overrides the channel assignments of all previous ADR messages.

The ADR message is sent whenever BagMessage starts or whenever the redundant partner takes over. It is also sent whenever a baggage system logs on to BagMessage (An option exists to allow/disallow the sending of ADR messages from BagMessage). It is recommended that the baggage system keep a record of the last received ADR message in case it fails and needs to be restarted.

Baggage systems should modify channel assignment tables based on the contents of ADR messages. For example, if an ADR message received on channel 1 contains the address stamp, 'SR', then the baggage system should expect to receive messages with address stamp 'SR' on channel 1, and should transmit messages with address stamp 'SR' on channel 1, until the channel is changed by a subsequent ADR message. Note that address stamps can only be valid on one channel at a time. Note also that when operating in hot-standby mode, the BSU should expect one ADR message from the primary Server containing all the connected airline address stamps and a second, empty, ADR message from the secondary BagMessage Server.

The format of an ADR message is as follows:

'BSI' <≡ 'ADR' <≡ <AS 1> <≡ <AS 2> <≡ ... <AS n> <≡ 'ENDADR'

where:

- 'BSI' is a special address stamp reserved to identify messages originating in BagMessage
- <≡ is end of element delimiter, <CR> <LF>
- 'ADR' is a special Message Header designating the start of an address stamp update msg.
- <AS n> is a 2-4 character address stamp
- 'ENDADR' is a special Message Trailer designating the end of a BSI ADR message

Examples:

BSI	BSI Address Stamp
ADR	Standard message identifier
SR	First address stamp
AA	Second address stamp
LH1	Third address stamp
LH2	Fourth address stamp
ENDADR	End of message identifier

If this message is received by the baggage system on channel 1, then it can expect to receive all messages stamped with 'SR', 'AA', 'LH1', and 'LH2' on channel 1. Furthermore, it will not

receive messages with any other address stamp other than those mentioned above. All address stamps assigned to that channel by previous ADR messages are no longer valid.

Note that a baggage system should receive an ADR message on a channel before receiving any baggage messages from carrier hosts.

Note that the internal address stamp, 'BSI' is always valid on any channel.

5.1.3 BPM considerations

The primary example where bi-directional communications are used is when a BSU wishes to send a BPM to an airline DCS (or other application, such as baggage tracking).

The following section relates specifically to the processing of ADR messages and the sending of BPMs at LHR. It may be applicable at other sites where hot-standby operation is used.

After startup of the BagMessage Servers, and following a log-in message from a BSU, the BagMessage Servers send an ADR message, stating which airline DCSs are being processed by that BagMessage Server. The primary BagMessage Server sends a ADR message listing all connected airlines and the secondary server sends an empty ADR message. The primary BagMessage Server may send several ADR messages as sessions are established between the BagMessage system and the DCS systems. If any new DCS connects or a subsequent BagMessage fail-over occurs, the affected (and still live) BagMessage Server(s) issue a new ADR message listing all the DCS being processed by the BagMessage Server(s) at that time.

i.e. after fail-over, the standby BagMessage Server establishes sessions with the airline DCS and sends an ADR message to all logged on BSUs. No ADR message is sent from the failed primary BagMessage Server until it is brought back into service, when it will re-takeover the airline DCS sessions and send out an appropriate ADR message.

In other words, the most recent ADR message from any BagMessage Server must be taken to supersede all previous ADR messages. A baggage system should store the latest ADR status and read this every time it is restarted and use this information to determine to which BSI to send BPMs.

The following general algorithm has been devised:

1. On start-up, the BSU sends BPMs to the BagMessage Server that was set as active in the latest ADR message received.

2. Check from which BagMessage Server each message (BSM, BUM etc) is received and if this is different to that designated in step 1 change the BagMessage Server to which to send BPMs.

3. If a new ADR message is received and it denotes a change in the active BagMessage Server make that change for sending BPMs and update the stored ADR file.

4. Go to step 2.

5.1.3.1 BPMs and "Dummy Airlines"

There may be circumstances where a BSU generates a BPM without a BSM having been received previously. The BSU would not know which airline id to use in the address stamp. For example, such a case arises if a through-checked transfer bag arrives at a sortation system before the passengers present themselves at the transfer desk. While such a BPM is not usually required by an airline DCS, the BSU may wish to send the BPM for delivery to a baggage tracking system. In such cases, and in coordination with the local system administrator, a "dummy" airline id can be used (e.g. "XX", or "SYM").

5.1.3.2 BPMs and DCS

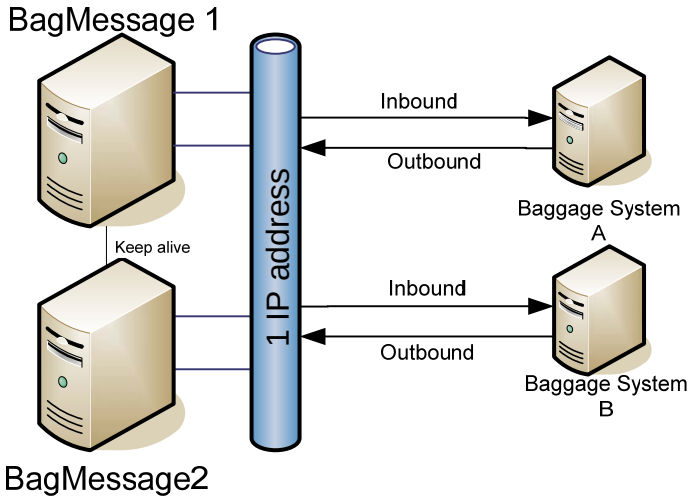
Not all airlines are capable of accepting BPMs. BagMessage can be configured to accept BPMs from a BSU, but not to relay them on to an airline DCS.

5.1.3.3 BPMs and Tracking Systems

In many cases, BPMs generated by a BSU must be delivered to a tracking system. This may be in addition to sending the same BPM to the airline DCS, if it will be accepted. BagMessage may be configured to 'copy' all BPMs received for a specific airline (including "dummy" airlines) to a specified BSU, such as a Tracking database.

5.1.3.4 BPMs and Multiple Connections between BagMessage and BSU

BSUs which both receive BSMs and send BPMs shall be programmed to receive BSMs on one TCP/IP connection, and send BPMs on another. The figure below illustrates this configuration:



In this configuration, the BSU programmer must consider the following:

- The connection to BagMessage for receiving BSMs will *only* be used for receiving BSMs and exchanging other protocol messages (such as 'keep alive' and ADR messages). BPMs must not be sent on this connection.
- Likewise, the connection with BagMessage for BPMs will *only* be used for sending BPMs.
- There will be no indication in the protocol that a connection with BagMessage is used for BSMs or BPMs: this information must be contained in the BSU's configuration.

The BSU will exchange two sets of ADR messages: one on the BagMessage BSM connection, and one on the BagMessage BPM connection. The ADR messages instruct the BSU how to address BPMs being sent to BagMessage. Each BPM set to BagMessage must contain the appropriate ADR address so that BagMessage can successfully distribue the message.TCP/IP Interface Specification

6 TCP/IP Interface Specification

This Chapter is for use only by those applications that require a TCP/IP interface to BagMessage

SITA does not allocate IP addresses to the baggage system computer(s) or airline DCS. This is the responsibility of baggage system supplier and airlines. Note also that a default routing will need to be established in the baggage system computer or airline DCS.

A baggage system computer or airline DCS is configured for IP with an address and a subnet mask. Each device with an address/mask sits on a subnet and all valid IP addresses within that subnet are calculable based on the IP address and subnet mask. Therefore, any device can send packets to any other device within the same subnet.

However, a device has no way of calculating valid IP addresses on other subnets. For this case, the protocol supports configuration of a default gateway (to get out of the subnet). Packets destined for a non-calculable IP address are forwarded to the default gateway address. From there, it is up to the gateway (the main distribution router in the BagMessage environment) to forward the packet to the proper subnet or notify the sender that destination is unreachable.

For example, the command line on a DEC machine running VMS and for Ethernet default routing is:

SET ROUTE 57.0.0.0 /GATEWAY = 159.245.100.102

6.1 Security

With the use of TCP/IP, SITA takes measures to ensure only authorized data traffic is permitted on the links to the baggage system computer. As a general policy, each baggage system computer is connected to BagMessage system via back-to-back routers over a dedicated link. Tight packet filtering is used to exclude all traffic except baggage related messages. The filtering is done by source & destination IP address and source & destination port. The routing tables ensure that only traffic for the baggage system computer appears on the link, and that the link is not cluttered with traffic for other baggage system computers.

6.2 Testing

For the testing of a TCP/IP interface, a BagMessage simulator program is available from SITA. It runs on a PC and needs a LAN interface to the baggage system computer or a WAN/LAN interface to the airline DCS.

6.3 Information exchanged

In preparation for the on-site installation of a BagMessage system connection to a baggage system, the following data items need to be exchanged:

6.3.1 Information Supplied by Server:

Data Item	Example
Login Name	LHR3GEC001
Password	LHR3GEC001
IP address of Primary Server	LHR1BSIU01 57.1.26.31

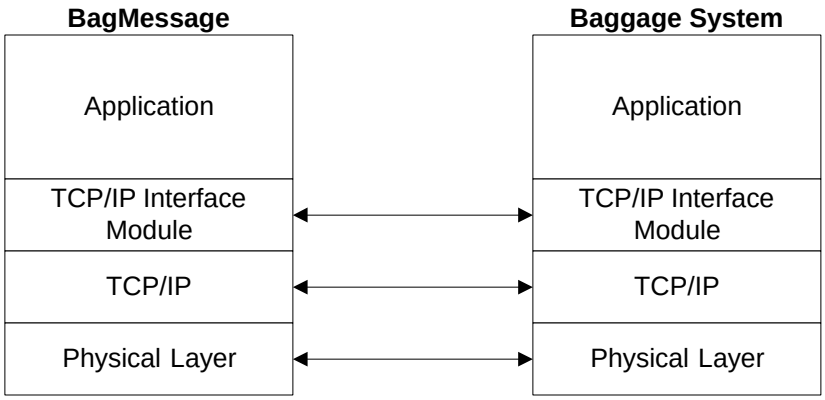
Port Number on Primary Server	8005
IP address of Secondary Server	LHRBSIU02 57.1.26.32
Port Number on Secondary Server	8006

6.3.2 Information Supplied by Client

IP address of Primary Connection
IP address of Secondary Connection

6.4 General considerations

BagMessage uses all the standard facilities of TCP/IP. This document describes the TCP/IP Interface Module which performs the function of transmitting messages on behalf of a calling program. The calling program loads the message into a buffer, specifies the length of the message, then calls the TCP/IP Interface Module. The Interface Module returns an error or status message to the calling program.



The following general considerations apply:

- BagMessage is a concurrent-connection oriented server (it 'listens' for connection requests) that can distribute data to one or more baggage system clients (which request connections to BagMessage). BagMessage can also be a TCP/IP client, connecting to a specific IP Address and port. Therefore, each baggage system can be configured to be either a TCPIP Client or Server.
- As BagMessage is developed as a service that provides and manages redundancy, it is preferred that BagMessage be configured as the Client and thus be allowed to manage and recover sessions as required.
- When a baggage system is configured to be a TCPIP Server, it should accept more than 1 connection on the same port. This is done by doing a second 'listen' command when a connection is established.
- All messages sent from BagMessage to baggage systems are complete -- no message assembly is required above TCP/IP,

- All message sent from the baggage system to BagMessage must be complete (two-way communication option, only), with no message assembly required,
- BagMessage will, optionally, request an acknowledgment for each message sent to a baggage system,
- BagMessage will acknowledge each message which contains an acknowledge request (two-way communication option, only),
- A Baggage System client must log into BagMessage before it can send or receive data.
- A Baggage System server will accept a login request before it can send or receive data.
- A Baggage System should allow for multiple connections. It is recommended that high traffic systems should support many connections to distribute the load. For redundancy, each connection to a primary BagMessage system needs another connection to a backup BagMessage system. For example, a high traffic system can have 3 connections to primary BagMessage system. It also has 3 more connections to backup BagMessage system in the event a BagMessage system has a failure. That would make 6 connections total.

NOTE: This is not to be confused with redundancy within the Baggage System itself. In most cases, a Baggage System will consist of two machines a primary and a backup. Following the example from above, the primary will have 6 connections and the backup another 6. Both sets of connections will receive the same messages, but every one of the 6 connections within each machine will receive unique messages.

All messages defined here pass over a TCP session, established between the Baggage System client and BagMessage. This session must be established before any of the messages in the protocol defined in this document are exchanged.

6.5 TCP/IP Host Considerations

The Baggage System Interface described within this chapter is exactly the same as for a TCPIP Host. The only difference is the TCP/IP Host must be a TCPIP Server. This is because of the redundancy within BagMessage. The backup system only allows host connections to be established after the primary has failed. Since the backup system will have a different IP address, the TCPIP Host will not be able to reconnect to the backup. Therefore, the BagMessage backup system must initiate the connection.

6.6 Message Structure

All messages exchanged between TCP/IP Interface Modules have a header followed by optional data. The header is structured as follows:

```
typedef struct TCP_IP_MSG
{
        char            appl_id[8];
        u_short version;
        u_short type;
        u_short message_id_number;
        u_short data_length;
        char    reserved[4];
}
```

Hex Offset	Decimal Offset	Size in Bytes	Field Name

0000	0	8	appl_id
0008	8	2	version
000a	10	2	type
000c	12	2	message_id_number
000e	14	2	data_length
0010	16	4	reserved

The header record fields: version, type, message_id_number, and data_length are stored least significant byte first (PC storage convention). On sending data via the TCP/IP socket it also arrives LSB first. These fields are unsigned values whose range is 0 to 65535. In accordance with general programming practice, the structure should be filled with nulls (MEMSET) before filling in fields to ensure unused characters are set to null.

char appl_id[8]

8 ASCII characters used to uniquely identify the application on the network. For example, BagMessage at Heathrow may use an appl_id of 'LHR_BSI'. This identifier can be used by BagMessage to direct specific messages to a specific baggage system, and also helps identify message senders for debugging. This string will be assigned by SITA. If it is not 8 characters long it should be null padded.

Example: header.appl_id = "LHR_BSI";

u_short version

Serves to identify which version of header is being sent. This document deals only with version 2.

Example: #define VERSION_2 2

header.version = VERSION_2;

u_short type

Identifies the type of message being sent. Valid types include:

```
#define LOGIN_RQST          1
#define LOGIN_ACCEPT        2
#define LOGIN_REJECT        3
#define DATA                4
#define ACK_DATA            5
#define ACK_MSG         6
#define NAK_MSG         7
#define STATUS          8
#define DATA_ON         9
#define DATA_OFF        10
#define LOG_OFF         11
```

The purpose and construction of these messages is defined in the sections below.

u_short message_id_number

Each message is given a message_id number. This may is used when acknowledging receipt of a message and may be used to detect duplicate transmission of messages.

u_short data_length

Specifies the number of data characters following the header

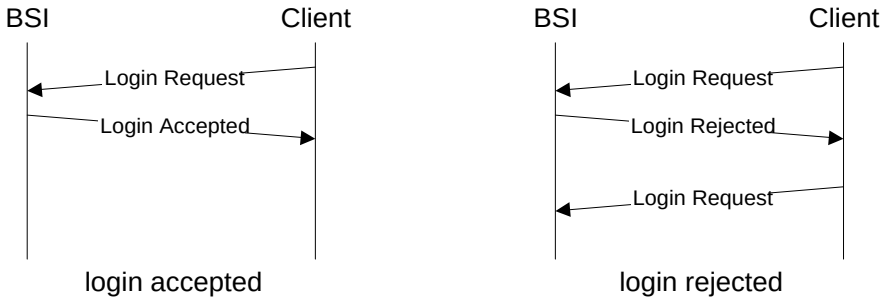
char reserved[4]

4 bytes are reserved for future use. They should be filled with nulls until otherwise defined.

6.7 Login Request

Before a baggage system client can send or receive data from BagMessage, it must first log in. To log in, it sends a log in message that is either accepted or rejected. If rejected, the client should attempt to log in again after a certain amount of time. Note that BagMessage can be configured as either a client of a server, depending on individual site requirements. If configured as a server it expects a login request and responds with a login response while if configured as a client it will send a login request and expect a login response from the attached baggage system.

The login sequence is as follows:



login accepted

login rejected

A login request message has the following construction:

```
header.application_id = “LHR_BRS”;
header.version = VERSION_2;
header.type = LOGIN_RQST;
header.message_id_number = n;
header.data_length = strlen(“password”);
data = “password”;
```

Where:

- The application_id field uniquely identifies the client. It will be provided by SITA at implementation time.
- The version field identifies the message version used by the client
- The type field indicates that this message is a login request
- The message_id_number field uniquely identifies the message. Any number may be used in the login request and the login response will echo back this number.
- The data_length field indicates the length of the password
- The data field contains a valid password in ASCII. The password is not encrypted and will be supplied by SITA.

6.8 Login Accept / Reject

A login response message has the following construction:

```
header.application_id = “LHR_BSI”;
header.version = VERSION_2;
header.type = LOGIN_ACCEPT;                    // or LOGIN_REJECT
header.message_id_number = n;
header.data_length = 0;
```

- The application_id field uniquely identifies BagMessage.
- The version field identifies the message version used by the client
- The type field indicates that this message is a login response and can contain either LOGIN_ACCEPT or LOGIN_REJECT
- The message_id_number field echoing the message_id number sent in the login request.
- The data_length field indicates the login response message contains no data

6.9 Logoff Message

When a connection between BagMessage or the baggage system client is closed by either party, a LOGOFF message should be sent. No response to a LOGOFF message is required. A logoff request message has the following construction:

```
header.application_id = “LHR_BRS”;
header.version = VERSION_2;
header.type = LOGOFF;
```

```
header.message_id_number = n;
header.data_length = 0;
```

- The application_id field uniquely identifies the client. It will be provided by SITA at implementation time.
- The version field identifies the message version used by the client.
- The type field indicates that this message is a logoff.
- The message_id_number field indicates the id number of the message.
- The data_length field indicates that no data is sent with a logoff message.

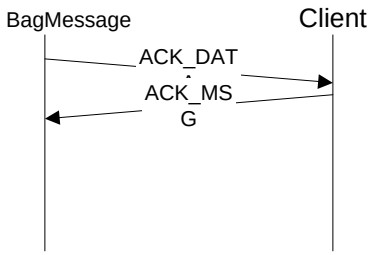
6.10 Data

When BagMessage or client sends data, it encloses it in a data message. When the sender transmits the data, it can either send the data in a DATA message, in which case no acknowledgment is required, or in an ACK_DATA message, in which case the receiver must immediately (before any other messages are sent on the same channel) send an acknowledgment in response.

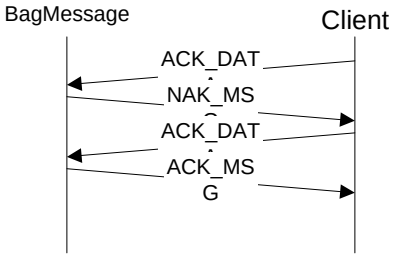
The acknowledgment to an ACK_DATA message can be either a positive ACK_MSG, meaning that the message has been properly received, or a negative NAK_MSG, meaning that the message has been received, but is not properly formed.

The sender waits until receiving the ACK_MSG or NAK_MSG before transmitting the next message. If the sender does not receive an acknowledgment within a certain amount of time (typically one second), it resends the data.

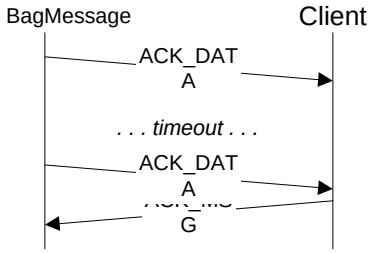
The valid data sequences are as follows:



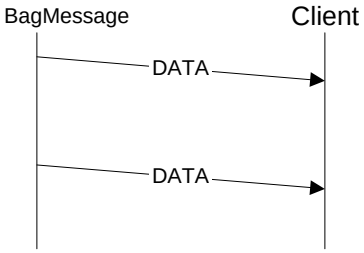
ACK_DATA Message
ACKed



ACK_DATA Message
NAKed and resent



ACK_DATA Message
resent after timeout



DATA Messages
(no ack required)

6.11 DATA Message

A DATA message has the following construction:

```
header.application_id = “LHR_BSI”;
header.version = VERSION_2;
header.type = DATA;
header.message_id_number = send_id;
header.data_length = strlen(message);
data = message;
```

- The application_id field uniquely identifies the sender (BagMessage)
- The version field identifies the message version used by the client
- The type field indicates that this message is a DATA message (no acknowledgment required)
- The message_id_number field indicates the message id number of the sent data
- The data_length field indicates the length of the message
- The data field contains the message. This is typically one of the IATA Baggage Messages defined in reference #1 (IATA RP 1745) such as BSM, BUM etc. See also reference #3 (IATA RP 1797b) and section 5 of this document for details on the ADR messages that may also be received and the ADR prefix to baggage messages that may (or may not) be present.

6.12 ACK_DATA Message

An ACK_DATA message has the following construction:

```
header.application_id = “LHR_BSI”;
header.version = VERSION_2;
header.type = ACK_DATA;
header.message_id_number = send_id;
header.data_length = strlen(message);
data = message;
```

- The application_id field uniquely identifies the sender (BagMessage)
- The version field identifies the message version used by the client
- The type field indicates that this message is an ACK_DATA message (acknowledgment required)
- The message_id_number field indicates the message id number of the message
- The data_length field indicates the length of the message
- The data field contains the message. This is typically one of the IATA Baggage Messages defined in reference #1 (IATA RP 1745) such as BSM, BUM etc. See also reference #3 (IATA RP 1797b) and section 5 of this document for details on the ADR messages that may also be received and the ADR prefix to baggage messages that may (or may not) be present.

6.13 Acknowledgment Message

An acknowledgment message may be positive (ACK) or negative (NAK). An acknowledgment message has the following construction:

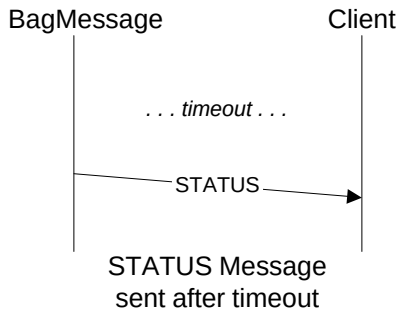
```
header.application_id = “LHR_BRS”;
header.version = VERSION_2;
header.type = ACK_MSG;                    // or NAK_MSG
header.message_id_number = receive_id;
header.data_length = 0;
```

- The application_id field uniquely identifies the receiver
- The version field identifies the message version used by the client
- The type field indicates that this message is a ACK_MSG message (or NAK_MSG)
- The message_id_number field indicates the message id number which is being ACKed or NAKed. If BagMessage receives an ACK for an unexpected id number it will resend the data message again. If a NAK is received, BagMessage resends the message. The message id number in the resent message may not be the same as in the original data message.
- The data_length field indicates that acknowledgment messages contains no data
- If a message requiring an acknowledgment is not acknowledged, BagMessage will resend the message after a configurable timeout. This will be repeated ad infinitum, or until BagMessage is informed by the network software that the link to the client is down. On restoral of the link BagMessage will start with the unacknowledged message.

6.14 STATUS

When either BagMessage or client has no data to send for a certain amount of time (typically 30 seconds), it must send a STATUS message to keep the session open. No response is required to a STATUS message. This message can be used as a ‘heartbeat’ to indicate that the partner system is up and running, although not supplying data. If the client is receiving data from BagMessage that does not require an ACK, it must still send a status message to keep the connection open.

The STATUS message is sent as follows:



STATUS Message
sent after timeout

A STATUS message has the following construction:

```
header.application_id = “LHR_BSI”;
```

header.version = VERSION_2;
header.type = STATUS;
header.message_id_number = n;
header.data_length = 0;

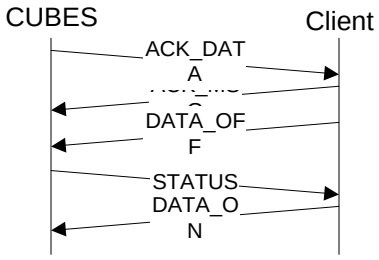
- The application_id field uniquely identifies the sender (BagMessage)
- The version field identifies the message version used by the client
- The type field indicates that this message is a STATUS message
- The message_id_number field indicates the message id number of the message
- The data_length field indicates that STATUS messages do not contain data
- If BagMessage does not receive data or a status message within a timeout period BagMessage closes the connection to the baggage system.

6.15 FLOW Control

When the client needs to temporarily halt the in-flow of data, it sends an DATA_OFF message to it's sending partner. When it is ready to receive data again, it sends a DATA_ON message.

After receiving a DATA_OFF message, BagMessage or client will not send data, but may still send status messages to keep the circuit alive.

Flow control message are sent as follows:



Flow Control

6.15.1 DATA_OFF Message

A DATA_OFF message is *never* sent by BagMessage.

A DATA_OFF message has the following construction:

header.application_id = "LHR_BRS";
header.version = VERSION_2;
header.type = DATA_OFF;
header.message_id_number = n;
header.data_length = 0;

- The application_id field uniquely identifies the sender
- The version field identifies the message version used by the client
- The type field indicates that this message is a DATA_OFF message
- The message_id_number field indicates the message id number of the message

- The data_length field indicates that DATA_OFF messages do not contain data

6.15.2 DATA_ON Message

A DATA_ON message is *never* sent by BagMessage.

A DATA_ON message has the following construction:

```
header.application_id = “LHR_BRS”;
header.version = VERSION_2;
header.type = DATA_ON;
header.message_id_number = n;
header.data_length = 0;
```

- The application_id field uniquely identifies the sender (Baggage Reconciliation)
- The version field identifies the message version used by the client
- The type field indicates that this message is a DATA_ON message
- The message_id_number field indicates the message id number of the message
- The data_length field indicates that DATA_ON messages do not contain data

6.15.3 Message Id Numbers

The message sender identifies each message uniquely with an id number. There should be no expectation by the receiver they run in sequence, or that they start at a particular number.

When resending a message, it is up to the sending application to decide if it resends the message with the same id number, or if it uses a new one. In either case, if the resend is received correctly, the received id number will be echoed back to the sender.

The protocol does not cater for resending of messages already acknowledged by the receiver. The protocol also has no provision to realign the id numbers.

7.0 DOCUMENT SIGN OFF

Functional Specification Name	Filename

Accepted by: SITA	**Accepted by:** CRTC
Signature…… Name…….. Date……...…….	Signature…… Name…….. Date…….