DUBAI INTERNATIONAL AIRPORT
T1 DEPARTURE HALL D SORTATION BHS
HIGH LEVEL CONTROL SYSTEM

INTERFACE SPECIFICATION
APPLICATION PROTOCOL: TCPSERVER2CLIENT

Document No.:	BHS-504-10
Project No.:	S01C06221Y
Author:	XJ
Release Date:	09-Sep-2007
Revision:	1.01
Pages:	33

CONTENTS

1. REVISIONS

Version	Release	Date	Init.	Description
1	00	03-May-2007	XJ	Initial version.
1	01	09-Sep-2007	XJ	Add in the definition of new telegrams: • Parameter Changed Notification (PCN), 0106 • Parameter Changed Notification Acknowledge (PNA), 0107

2. PREFACE

2.1 DOCUMENT OBJECTIVE

The objective of this document is to specify the standard communication protocol between 1) the PLC Gateway Services to Business Data Handling Service, 2) SAC services to BHS Console Application that are used in the Dubai International Airport, Terminal 1 Departure Hall D Sortation BHS project.

For the application protocol all messages are defined and the protocol for exchanging them is specified.

2.2 SCOPE

It is the scope of this document to state all necessary information needed in order to implement the Item Tracking application protocol for SAC-PLC interface.

2.3 AUDIENCE

This interface specification is intended as a technical document specifying the protocols. Also the document serves, as the technical basis when deciding which standard transport protocols should be used in a particular solution.

2.4 DOCUMENT ORGANISATION

This Interface Specification contains the following chapters:
Chapter 3, "Introduction"
This chapter supplies an overview of the SAC-BSI protocol introducing primary concepts and communication partners.
Chapter 4, "Transport Protocol"
This chapter is in part a repetition of the major points in the [BagMessage interface] specification, but also a further specification of the open issues and / or protocol parameters.
Chapter 5, "Application Protocol"
This chapter describes the application protocol following the IATA Recommended Practice 1745.
Chapter 6, "Application Protocol Layer: Keep-Alive"
This chapter describes the keep-alive protocol.
Chapter 7 Error! Reference source not found.**, "REFERENCES"**
This chapter contains a list of the references used in the specification.

2.5 DOCUMENT LIMITATIONS

The transport protocol is described in [IR-102-01-1.01 IS_TP_FrameOnTCP].

2.6 DOCUMENT MAINTENANCE

This document is the one of the IR project document suites and maintained by Inter-Roller Engineering Limited.

3. INTRODUCTION

3.1 PLC GATEWAY SERVICE AND BUSINESS DATA HANDLING SERVICE

The SAC services are consisted of multiple PLC Gateway **(PLC-GW)** service for each PLC, the Central Business Data Handling service, the DCS Alive Status Monitoring Service, and the SSIMS-GW service. All of these services are belong to the SAC suite and normally running in the same SAC computer. Of cause, they are distributable design system and therefore can running in the different computers.

The PLC-GW service is responsible for the data collecting from PLCs. All BHS production data that are sent from PLCs will be collected by these PLC-GW services first according to the interface design specification (Refer to the [BHS-504-01-1.02 IS_BTS2PLC.doc]), and then the valid business data will be forwarded to the Central Business Data Handling service, which is running in the same computer or in the other computer, by the PLC-GW services. The communication protocol between PLC-GW services and PLCs can be many depend on the support of PLC hardware. The Omron FINS protocol is adopted in this project.

Beside the interface to the PLCs, the PLC-GW is also implemented as the TCP server to allow other services (e.g. Business Data Handling service and DCS Status Monitoring Service) and applications (e.g. BHS Console GUI Application) to connect to it.

The Business Data Handling **(BusinessHandler)** Service is responsible for the business data handling and replies the required data to the PLCs via PLC-GW services. There is only one BusinessHandler service on the system to centrally serve all PLC-GW services. It is implemented as the TCP client to communicate with the PLC-GW TCP server. The BusinessHandler service does not has the direct connection to the PLCs. If the BusinessHandler service need to send any data to PLC, it will send to the PLC-GW service and the PLC-GW service will send these data to the PLC. Regarding to the functionalities of BusinessHandler service, refer to the detail design specification of SAC [BHS-503-01-1.00 DDS_SAC.doc]

3.2 DCS STATUS MONITORING SERVICE AND PLC GATEWAY SERVICE

According to the project requirement, the SAC services need send the DCS alive status to PLCs. One DCS Status Monitoring **(DCSStatusMonitor),** therefore, is implemented in the Jinan BHS HLC system. This DCSStatusMonitor service will monitor the DCS alive status periodically and send the DCS status to the PLC-GW service through the TCP connection.

There is only one DCSStatusMonitor service is implemented in the system. It works as the TCP client and is responsible for opening the TCP connections to each PLC-GW service.

3.3 SAC SERVICES AND BHS CONSOLE APPLICATION

The SAC services suites are consisted of multiple PLC Gateway **(PLC-GW)** service for each PLC, the Central Business Data Handling service, the DCS Alive Status Monitoring Service, and the SSIMS-GW service. In order to monitor these services internal running status, each service is implemented as a TCP server to allow the BHS remote console application to make the TCP connection. Through this TCP connection, the console application can send the status request to the SAC services and the SAC services can reply its predefined running status parameters and values to the console application for the display.

3.4 PARAMETER CHANGED NOTIFICATION SERVICE

The BHS SAC system is consisted of number of components. These SAC components could be:

- SAC to PLC communication gateway services;
- SAC Sort Engine service;
- SAC Service Monitoring service;
- Departure Flight Allocation application;
- Manual Encoding Station application;
- BHS-FIDS Interface communication gateway service;
- BHS-CUTE Interface communication gateway service;
- And other BHS related applications...

Each of above components has its own configuration parameters. Some of the parameters are used to provide the flexibility to developer for fine tuning the application during the development, in-house testing or on-site test stage. Once the value of the parameters is finalized by the testing, they will not be changed during the BHS production time. Usually this type of parameters is called the “Internal” parameter. There is no any GUI will be provided to end user for them to change the “Internal” parameter settings.

There are two places are used to store parameter settings for SAC components:

- One is the XML configuration file;
- Another one is the database table [SYSCONFIG].

Each SAC component has its own XML configuration file, which is used to store the setting of all “Internal” and “Public” parameters.

The table [SYSCONFIG] in the SAC database is used to store the setting of those “Public” parameters. This database table can be accessed by all SAC components. It is the central place for all SAC applications and services to store their own “Public” parameters. When the setting of any “Public” parameters is changed by end user in the configuration GUI, the new setting will be updated into this database table, but not the XML configuration file.

When end user changes the setting of “Public” parameters, in order for them to be taken effect as soon as possible and without restart the application or service, the parameter related SAC component has to be notified about the change of setting. This can be done by [ParamCHGNotification] service.

[ParamCHGNotification] is a Windows service application to centralized monitors the parameter setting changes in the database table [SYSCONFIG]. If it detects that any parameter setting (the value of column “SYSVALUE”) is changed, it will send the “Parameter Changed Notification” (PCN) telegram to SAC components. Once receives the PCN, the SAC component needs to reload the parameter new settings from [SYSCONFIG] table and use it in future.

Once receives the PCN telegram, the SAC component needs send “Parameter Changed Notification Acknowledge” (PNA) telegram back to the notification service.

3.5 PROTOCOL LAYERS

The communication protocol between 1) PLC-Gateway Service and Business Data Handling Service; 2) DCS Status Monitoring Service to PLC-GW services, 3) SAC Services and BHS Console Application is the TCP protocol.

The transport protocol between each party consists of a number of logical layers on top of an Ethernet connection. The protocol layers are depicted in Figure 3.1.

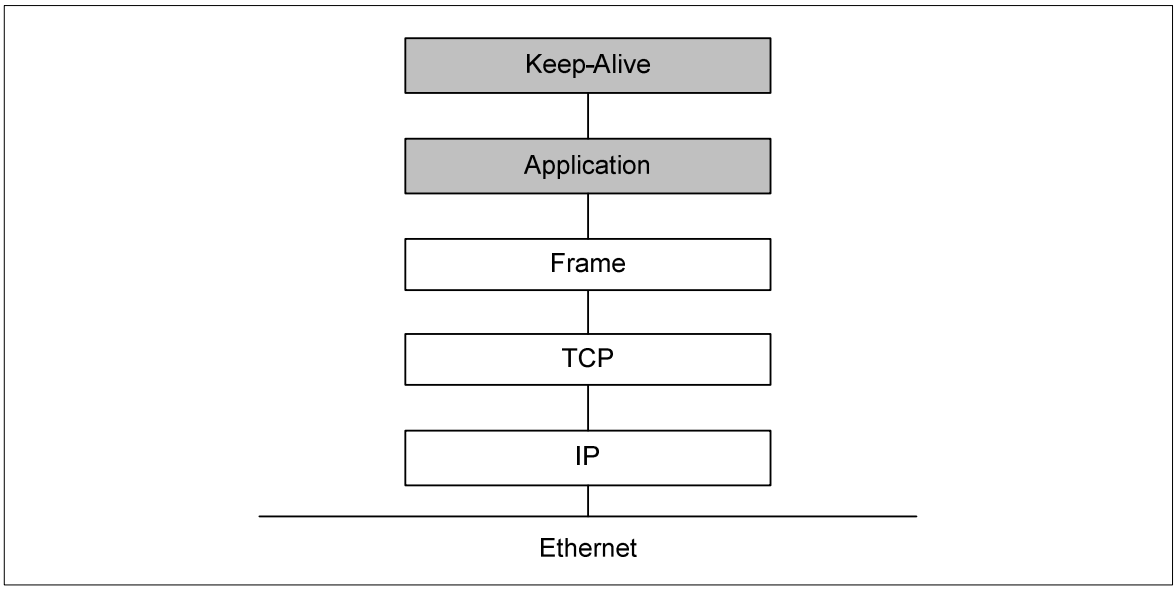

Figure 3.1: Overview of the transport protocol layers. The layers in the grayed boxes are within the specification scope for this document.

The following lists the main responsibilities for the layers in Figure 1:

- **Keep-Alive**: Ensures the connection validity through sending special keep-alive messages and monitoring the traffic on the TCP connection. Thus a connection with no traffic is considered a dead or broken connection. The keep-alive messages only can be sent after the application layer connection has been established.

- **Application**: The Application layer is the user of the Transmission Control Protocol and specifies the protocol for a particular project or product. The Application layer connection is established on top of the TCP protocol layer connection.

- **TCP**: Transmission Control Protocol (TCP). A connection oriented protocol that provides an reliable transport service for a user process.

- **IP**: Internet Protocol (IP). IP is the protocol that provides the packet delivery service for TCP (and others). The combination of TCP using IP is often referred to as TCP/IP.

In a layered protocol a data packet is passed through the layers in sequence. Each layer then has the possibility of adding / deleting or changing data according to the specifications. In some cases data packets is created or terminated within a layer and thus not transferred to the next layer.

Thus it very important that clear interfaces and responsibility division between the protocol layers are enforced.

3.6 COMMON APPLICATION MESSAGE FORMAT

The application protocol is a message based protocol. A message consists of a header, a data-block and a message end character (<ETX>).

Whenever possible all characters in the message header and data block should be in 7-bit ASCII format, in order to simplify its interpretation during testing of the interface. All values from 20Hex (included) to 7EHEX (included) are allowed for these two blocks. The data type shown in the following table is used in the BHS application data definitions to limit the contents of a specific field:

Field Type	Syntax	Range	Padding Rule	Not Available Rule
Alphanumeric	At least one character	[20-7E] Hex in the US ASCII character set.	Left justified, space filled	Filled with spaces (hex 0x20) if not available
Numeric	At least one character	[30-39] Hex in the US ASCII character set.	Right justified, zero filled	Filled with zero (hex 0x30) if not available
Date	YYYYMMDD	[30-39] Hex in the US ASCII character set.	Not applicable	Filled with spaces (hex 0x20) if not available
Time	HHMM	[30-39] Hex in the US ASCII character set.	Not applicable	Filled with spaces (hex 0x20) if not available

All fields in the application message header and data blocks are of fixed length according to the specified length given as number of characters. Values must therefore be padded (either preceding or succeeding the data) according to the field length and the above specified padding rule. Example, if the alphanumeric information does not take up the entire message field, spaces (20Hex) will be added after the alphanumeric characters. If the numerical information does not take up the reserved number of digits, the numerical information will be pre-fixed by zeroes (30Hex).

All messages must adhere to the format specified here. Messages that do not adhere to this format must be ignored by the receiving part.

4. TRANSPORT PROTOCOL

As illustrated in the Figure 3.1, the protocol layers between SAC and PLC's can be divided into 2 parts: Transport Protocols and Application Protocols.

The transport protocols are consisted of the protocols that are bottom of the Application protocol layer in Figure 3.1. And the application protocols include the Application protocol and Keep-Alive protocol.

4.1 TCP/IP PROTOCOL

IP is the protocol that provides the packet delivery service for TCP (and others). The combination of TCP using IP is often referred to as TCP/IP.

The specification of TCP is a standardized protocol and as such not part of this specification. For the TCP protocol specification the reader is referred to [RFC-793-TCP].

4.1.1 TCP/IP Protocol Parameters

Service Name	Application Code	IP Address	Port	Remark
BHS_SAC2PLC2GW	SAC2PLC2	192.168.10.58/24	24035	The TCP server IP and Port number for primary BHS_SAC2PLC2GW service which is running in SAC-COM1 server.
BHS_SAC2PLC2GW	SAC2PLC2	192.168.10.60/24	24035	The TCP server IP and Port number for secondary BHS_SAC2PLC2GW service which is running in SAC-COM2 server.
BHS_SAC2PLC3GW	SAC2PLC3	192.168.10.58/24	24036	The TCP server IP and Port number for primary BHS_SAC2PLC3GW service which is running in SAC-COM1 server.
BHS_SAC2PLC3GW	SAC2PLC3	192.168.10.60/24	24036	The TCP server IP and Port number for secondary BHS_SAC2PLC3GW service which is running in SAC-COM2 server.
BHS_SAC2PLC4GW	SAC2PLC4	192.168.10.58/24	24037	The TCP server IP and Port number for primary BHS_SAC2PLC4GW service which is running in SAC-COM1 server.
BHS_SAC2PLC4GW	SAC2PLC4	192.168.10.60/24	24037	The TCP server IP and Port number for secondary BHS_SAC2PLC4GW service which is running in SAC-COM2 server.
BHS_SAC2PLC5GW	SAC2PLC5	192.168.10.58/24	24038	The TCP server IP and Port number for primary BHS_SAC2PLC5GW service which is running in SAC-COM1 server.
BHS_SAC2PLC5GW	SAC2PLC5	192.168.10.60/24	24038	The TCP server IP and Port number for secondary BHS_SAC2PLC5GW service which is running in SAC-COM2 server.
BHS_SortEngine	SortEngn	192.168.10.58/24	24039	The TCP server IP and Port number for primary SAC Sort Engine service

				which is running in SAC-COM1 server.
BHS_SortEngine	SortEngn	192.168.10.60/24	24039	The TCP server IP and Port number for secondary SAC Sort Engine service which is running in SAC-COM2 server.
BHS_SvcMonitor	SvcMontr	192.168.10.58/24	24040	The TCP server IP and Port number for primary BHS Monitoring service which is running in SAC-COM1 server.
BHS_SvcMonitor	SvcMontr	192.168.10.60/24	24040	The TCP server IP and Port number for secondary BHS Monitoring service which is running in SAC-COM2 server.

5. APPLICATION PROTOCOL

5.1 APPLICATION LAYER CONNECTION ESTABLISHING

In order to maintain the reliability of production data transmission, the application layer connection between TCP server and TCP client is required to be established before the application message can be sent from the both parties.

Below Figure 5.1 and 5.2 illustrate the sequence diagrams of establishing of the application layer connection between the TCP server and TCP client.

5.1.1 TCP Server Application Layer Connection Establishment

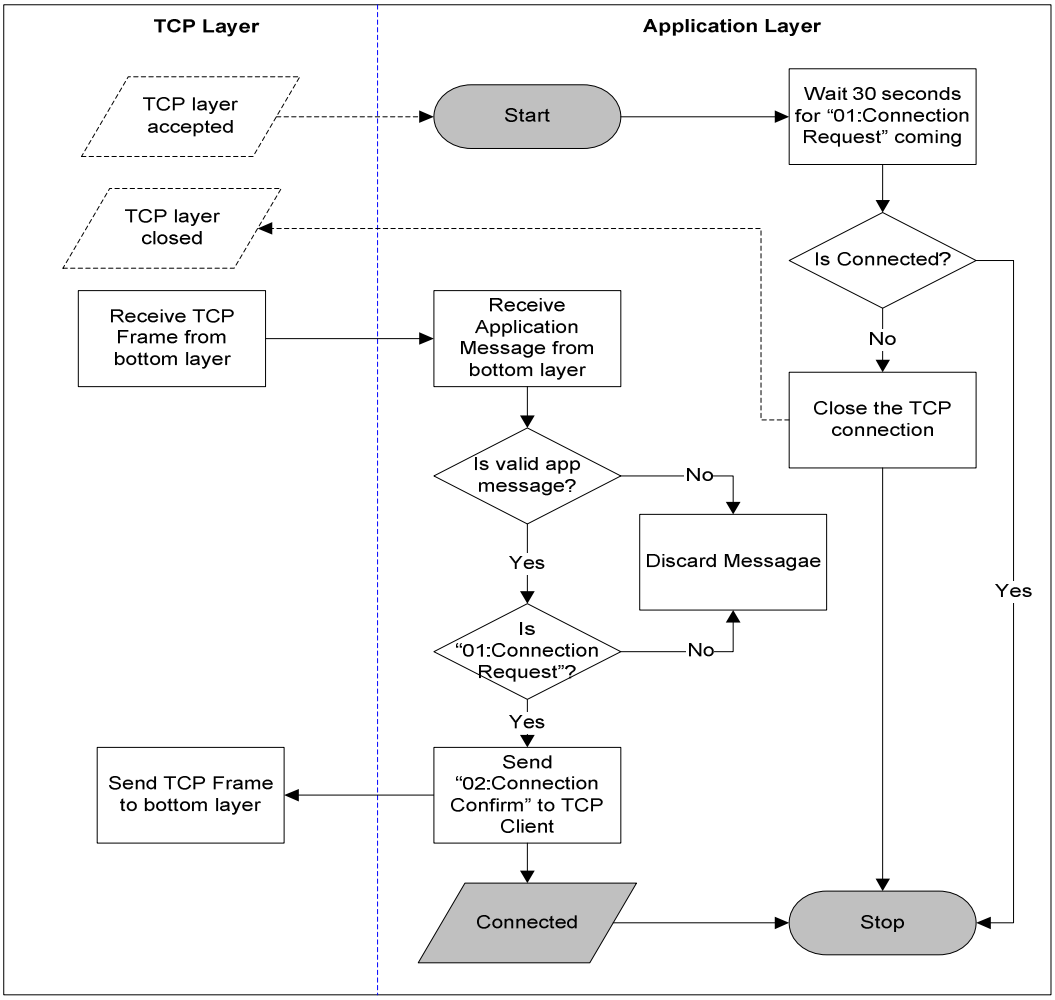

Figure 5.1: TCP Server application layer connection initialization sequence.

As illustrated in the Figure 5.1, the TCP server application will start the application layer connecting process once the bottom TCP layer connection to the remote TCP client has been accepted.

As per the design, the remote TCP client application should send the Connection Request **(CRQ)** message to TCP server immediately after the TCP layer connection has been opened. The TCP server

application will reply the Connection Confirm **(CCF)** message to TCP client upon receives the CRQ message. The application layer connection of TCP server side will be justified as connected after the CCF message successfully sent out.

If TCP layer connection has been accepted for **30 seconds** (configurable) Connection Request Waiting Timeout but the TCP server does not receive the CRQ message from TCP client, it will close the bottom TCP layer connection, release all allocated resources of it. And then wait for the remote TCP client to re-connect.

Except the CCF message, no other messages are allowed to be sent out from the TCP server side before the application layer connection is established.

If any reason cause the TCP server received more than one CRQ message, e.g. the returning of CCF message delay or it was lost at somewhere during the transmission to cause the TCP client re-initialize the connection after the re-connecting timeout, the TCP server must send the CCF message to the TCP client for every received CRQ messages.

5.1.2 TCP Client Application Layer Connection Establishment

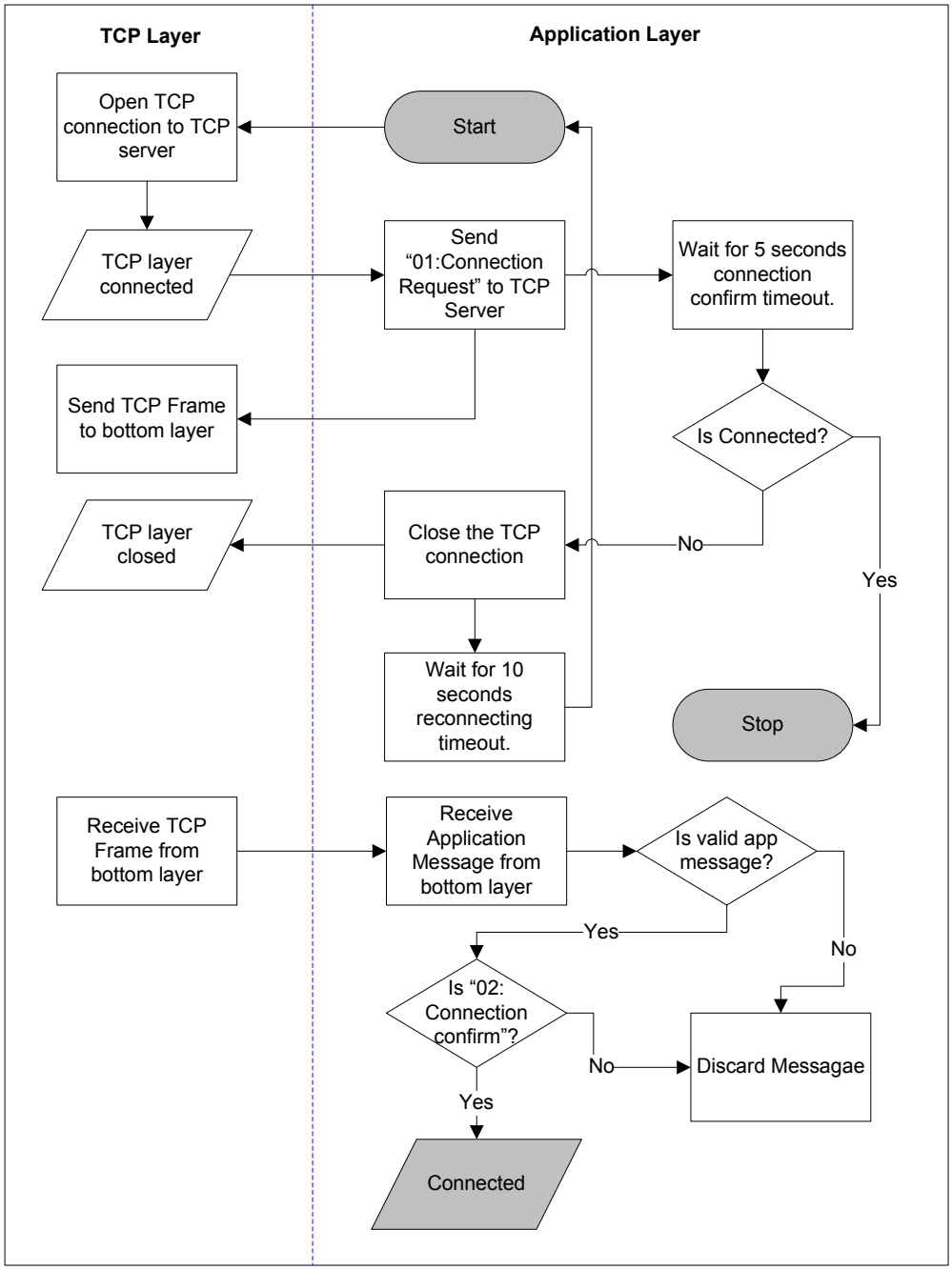

Figure 5.2: TCP client application layer connection initialization sequence.

As shown in the Figure 5.1, the TCP client application will initialize the application layer connection immediately after the TCP layer connection has been opened. It will send the Connection Request **(CRQ)** message to the TCP Server and waiting for the Connection Confirm **(CCF)** message is returned back from the TCP server application. The TCP client will justify the application layer connection is established upon receiving the CCF message.

If the application layer connection is not been connected after the CRQ message has been sent for **5 seconds** (configurable) connection-confirm timeout, then the TCP client application will close the bottom TCP layer connection. After the 10 **seconds** (configurable) re-connecting timeout, the same application layer connection establishing process will be started again.

Except the CRQ message, no other messages are allowed to be sent out from the TCP client side before the application layer connection is established.

After the connection established, one keep-alive process will be started at the both TCP server and client side to monitor the application layer connection status (Keep-Alive process will be described in great detail in later chapter). If the keep-alive process detects that the connection has been broken, it will stop the message sending and invoke the above application layer connection establishing process again after **10 seconds** (configurable) re-connecting timeout.

5.2 APPLICATION LAYER MESSAGES

Bellow is the list of all application layer messages could be exchanged between the TCP server and client applications through the TCP connection.

No.	Message Type	Message	Alias Name	Source	Destination	Acknowledge
1	0001	Connection Request	CRQ	TCP Client	TCP Server	No
2	0002	Connection-confirm	CCF	TCP Server	TCP Client	No
3	0101	Service Running Status Request	SRQ	TCP Client	TCP Server	No
4	0102	Service Running Status Reply	SRP	TCP Server	TCP Client	No
5	0103	Bag Events Reporting	BEV	TCP Server	TCP Client	**Yes**
6	0104	Service Start Command Message	STR	TCP Client	TCP Server	No
7	0105	Service Stop Command Message	STO	TCP Client	TCP Server	No
8	0106	Parameter Changed Notification	PCN	TCP Client	TCP Server	No
9	0107	Parameter Changed Notification Acknowledge	PNA	TCP Server	TCP Client	No
10	0090	Acknowledge	ACK	TCP Client	TCP Server	No
11	0099	Keep-Alive	SOL	Both	Both	No

As stated above some application messages must be acknowledged. Both sides can only send one acknowledge-required message before receive it's acknowledge. During the period of waiting for acknowledge, incoming messages from the remote side can still be received. This does not entitle the receiver to send another acknowledge-required message. However, the protocol allow the application to send messages that do not require acknowledge while waiting for an acknowledge message. In other word, an acknowledge-required message can only be sent if the previous acknowledge-required message has been acknowledged. All other acknowledge-unrequired messages can be transmitted at any time.

For acknowledge required message 22, if the sender does not receive acknowledge after **3 seconds** (configurable) acknowledge waiting timeout, the message will be resent to according to the pre-set number of times retry (**1 times**). If still no acknowledge is received after the maximum retry times, the application layer connection supposed has been broken. In this case,

- For the message sender side, it will stop the message sending for all messages and then close the TCP layer connection.
- For the receiver side, since there is no more messages will be received from the remote, it application and TCP layer connections will be closed by its keep-alive process.
- The messages that waiting in the outgoing queue of sender side will be re-sent after the connection has been established again.

Each message will be described in greater detail at below sections.

5.2.1 Application Layer Connection Confirm (0001)

Direction:

TCP Client => TCP Server

Alias Name:

CRQ

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0001	Telegram Type.
	4-7	Numeric	4	0020	Telegram Length.
	8-11	Numeric	4	0001	Telegram Sequence Number.
Data Fields	12-19	Alphanumeric	8	(SortEngn)	Client Application Code.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0001 – Telegram type, Application layer connection request telegram.

0020 – Telegram length, 20 bytes.

0001 – Sequence number, generated by TCP Client.

(SortEngn) – Client Application Code. Succeeding pad with space character (Hex: 0x20).

To inform the PLC which application is trying to make the connection to it. PLC is able to control which application can make the application layer connection.

There are following 2 valid application codes are defined for SAC-PLC communication gateway application in the Tocument Airport BHS project. They are:

"**SortEngn**" – SAC sort engine Application Code.

"**BHSCnsol**" – BHS Console Application Code.

Telegram Sample:

"00010020000 SortEngn*"*

Description:

Request for the application layer connection. This is the first telegram to be exchanged between TCP Client and TCP server interface.

After the transport layer (TCP and RFC1006) connections have been established, AC server will sends the Application Layer Connection Request telegram to PLC. This is the application layer connection handshake message used to establish the connection between the SAC application and PLC application.

The Connection Request telegram is the first application telegram need to be sent from SAC and it is expects to receive the Connection Confirm telegram (Telegram Type: 0002) from the PLC. SAC will send or receive other application telegrams only when "0002" telegram has been received. PLC will

send or receive other application telegrams only when “0001” telegram has been received and “0002” telegram has been sent out.

There are two instances of each PLC communication gateway (PLC-GW) application will be running at any time. One instance is running in the primary SAC-COM server hardware, and another one is running in the secondary SAC-COM server hardware. The identical Client Application Code will be set for these 2 instances.

At any time, only one application layer connection can be established for any Client Application Code. Hence, there will be only one of two instances of each PLC-GW application can successfully establish the connection to the PLC and exchange the application layer telegrams with PLC.

Once receives the “0001” telegram, PLC will check whether the application layer connection of the specified Client Application Code has already been established. If it has, PLC just ignores this “0001” telegram. If it hasn’t, the “0002” telegram will be generated and send to SAC to confirm the connection request.

The Connection Request telegram must be sent at least once after the transport protocol layer connection was established.

5.2.2 Application Layer Connection Confirm (0002)

Direction:
TCP Server => TCP Client

Alias Name:
CCF

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0002	Telegram Type.
	4-7	Numeric	4	0020	Telegram Length.
	8-11	Numeric	4	0001	Telegram Sequence Number.
Data Fields	12-19	Alphanumeric	8	(SortEngn)	Client Application Code.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0002 – Telegram type, Application layer connections confirm telegram.

0020 – Telegram length, 20 bytes.

0001 – Sequence number. The value "0001" of the received Connection Request telegram sequence number field will be echoed back to SAC by PLC.

(SortEngn) – Client Application Code. Succeeding pad with space character (Hex: 0x20).
PLC will echo back the same code in the "0001" telegram to SAC if the connection is accepted.

Telegram Sample:
*"000200200001*SortEngn"

Description:
Confirmation of application layer connection request.

Once receives the Application Layer Connection Request (0001) telegram, PLC must reply the Application Layer Connection Confirm (0002) telegram to SAC immediately.

If the "0002" telegram is not received within one time period (Connection Confirmation Timeout, Default: 3000ms) after the "0001" telegram has been sent to PLC, SAC will resend "0001" telegram to PLC. This process will be kept retried for preset number of times (Connection Request Retry Times, Default: 1times) before stop.

If the SAC still does not receive the "0002" telegram after the number of times retry, it will then issue the close command to transport layer to close the RFC1006 and TCP connection with PLC. After all layer connections have been closed, SAC will quite for a time period (Reconnect Timeout, Default: 10000ms) and then reinitialize the connections from the transport protocol layer to application protocol layer. The

above connection handshake will be repeated after the transport layer connections have been established again.

The Connection Confirm telegram must be sent by PLC for every received Connection Request telegram.

There are two instances of each PLC communication gateway (PLC-GW) application will be running at any time. One instance is running in the primary SAC-COM server hardware, and another one is running in the secondary SAC-COM server hardware. The identical Client Application Code will be set for these 2 instances.

At any time, only one application layer connection can be established for one Client Application Code. Hence, there will be only one of two instances of each PLC-GW application can successfully establish the connection to the PLC and exchange the application layer telegrams with PLC.

After receives the “0001” telegram, PLC will check whether the application layer connection of the specified Client Application Code has already been established. If it has, PLC just ignores this “0001” telegram. If it hasn’t, the “0002” telegram will be generated and send to SAC to confirm the connection request.

Parameters:

Parameter name	Default Values	Description
Connection Confirmation Timeout	3000	Millisecond.
Connection Request Retry Times	1	
Reconnect Timeout **(Same parameter as Keep-Alive protocol layer. 6: Application Protocol Layer: Keep-Alive)**	10000	Millisecond.

5.2.3 Running Status Request Message (0101)

Direction:
TCP Client => TCP Server

Alias Name:
SRQ

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0101	Telegram Type.
	4-7	Numeric	4	(0045)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	12-?	Alphanumeric	?	(IREL.Net.Transports.TCP.TCPServer)	The name list of class whose running status are required. • Use comma to delimitate the different classes, • Value **"ALL"** to request all classes of service running status.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0101 – Telegram type, Application layer connections confirm telegram.
(0045) – Telegram length
(1234) – Sequence number.
(IREL.Net.Transports.TCP.TCPServer) – The name list of class whose running status are required.

- Use comma to delimitate the different classes,
- Value **"ALL"** to request all classes of service running status.

Telegram Sample:
"010100451234 IREL.Net.Transports.TCP.TCPServer"

Description:
In order to monitor the SAC services status during the running time, each SAC service has been implemented one internal class status collecting and reporting mechanism to collect the pre-defined parameters and their values and send them to the BHS console GUI application upon the request.

The Service Running Status Request (SRQ) message will be sent by the BHS Console GUI application (TCP client). Once the SAC service (TCP Server) receives the request, it will collect the status parameters and send them to the console application.

These 2 messages are status monitoring messages, they are not critical for the BHS production. Hence there is no acknowledgement is required for them.

5.2.4 Running Status Reply Message (0102)

Direction:
TCP Server => TCP Client

Alias Name:
SRP

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0102	Telegram Type.
	4-7	Numeric	4	(0469)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number. Echo back the same Sequence field value of the 1001 message (SRQ).
Data Fields	12-?	Alphanumeric	?	?	XML string format service running status.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0102 – Telegram type, Application layer connections confirm telegram.
(0469) – Telegram length
(1234) – Sequence number.
(?) – XML string format service running status.

Telegram Sample:
"010204691234 <class_status><class name="BTS.Messaging.Handlers.BTSBizMessageHandler"><HandlingThreadCounter>4119097</HandlingThreadCounter><NoOfMsgsInIncomingQueue>1</NoOfMsgsInIncomingQueue><NoOfReceivedBRQMsg>16</NoOfReceivedBRQMsg><NoOfReceivedLTKMsg>15</NoOfReceivedLTKMsg><NoOfReceivedSCRMsg>8</NoOfReceivedSCRMsg><NoOfReceivedIPRMsg>4</NoOfReceivedIPRMsg><NoOfReceivedUnknownMsg>0</NoOfReceivedUnknownMsg><DCSAliveStatus>DD</DCSAliveStatus></class></class_status>"

Description:
In order to monitor the SAC services status during the running time, each SAC service has been implemented one internal class status collecting and reporting mechanism to collect the pre-defined parameters and their values and send them to the BHS console GUI application upon the request.

The Service Running Status Request (SRQ) message will be sent by the BHS Console GUI application (TCP client). Once the SAC service (TCP Server) receives the request, it will collect the status parameters and send them to the console application.

These 2 messages are status monitoring messages, they are not critical for the BHS production. Hence there is no acknowledgement is required for them.

5.2.5 Bag Event Message (0103)

Direction:
TCP Server <=> TCP Client

Alias Name:
BEV

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0103	Telegram Type.
	4-7	Numeric	4	(0024)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	10-?	Alphanumeric	?	(001100121234)	ItemTracking protocol message.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0103 – Telegram type, Acknowledge telegram.

(0024) – Telegram length,

(1234) – Sequence number.
The value is the echo back value of the sequence number field in the received acknowledgement required telegram, for example: Item Proceeded telegram.

(001100121234) - ItemTracking protocol message.
Sample here is the Chute Status Request message (Type: 0011) of ItemTracking procotol.

Telegram Sample:
"0103002412340011001212 34"

Description:
Once the valid BHS bag event message has been received by the SAC2PLC GW services (TCP Server), SAC2PLC GW service will encapsulate received bag event message into this Bag Event Message and send to SortEngine Service (TCP client) via TCP connection. Similarly, when SortEngine needs send ItemTracking message to SAC2PLC GW services, it will also encapsulate the outgoing data into BagEvent message first, and then send it to SAC2PLC GW services (TCP Server) via TCP connection.

This message is the BHS production related message and each message represents the different bag event. It is critical to the BHS production. Hence the acknowledgement is required from the message receiver. If the message sender does not receive the acknowledge after **3 seconds** (configurable) acknowledge waiting timeout, the same Bag Event Reporting Message will be resent to according to the pre-set number of times retry (**1 times**). If still no acknowledge is received after the maximum retry times, the application layer connection supposed has been broken. In this case, the message sender will stop any message sending and close its own application and TCP layer connection. And then wait for TCP client application to re-initialize the connection.

5.2.6 Service Start Command Message (0104)

Direction:

BHS Console Application (TCP Client) => BHS Service Monitor Service (TCP Server)

Alias Name:

STR

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0104	Telegram Type.
	4-7	Numeric	4	(0041)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	10-?	Alphanumeric	?	(BHS_SAC2PLC2GW,BHS_SAC2PLC4GW)	The name list of services that need to be started. • Use comma to delimitate the different service; • Value **"ALL"** request to start all service.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0104 – Telegram type, Acknowledge telegram.

(0041) – Telegram length,

(1234) – Sequence number.
The value is the echo back value of the sequence number field in the received acknowledgement required telegram, for example: Item Proceeded telegram.

(BHS_SAC2PLC2GW,BHS_SAC2PLC4GW) - The name list of services that need to be started.

Telegram Sample:

*"010400411234*BHS_SAC2PLC2GW,BHS_SAC2PLC4GW*"*

Description:

BHS Service Monitor (BHS_SvcMonitor) Service is running in the server machine to keep monitor the running status of each services that were registered in the XML configuration file. BHS_SvcMonitor service has the feature to start or stop the registered service upon receives the "Service Start Command" (0104, STR) message or "Service Stop Command" (0105, STO) message.

5.2.7 Service Stop Command Message (0105)

Direction:

BHS Console Application (TCP Client) => BHS Service Monitor Service (TCP Server)

Alias Name:

STO

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0105	Telegram Type.
	4-7	Numeric	4	(0041)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	10-?	Alphanumeric	?	(BHS_SAC2PLC2GW,BHS_SAC2PLC4GW)	The name list of services that need to be stopped. • Use comma to delimitate the different service; • Value **"ALL"** request to stop all service.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0105 – Telegram type, Acknowledge telegram.

(0041) – Telegram length,

(1234) – Sequence number.
The value is the echo back value of the sequence number field in the received acknowledgement required telegram, for example: Item Proceeded telegram.

(BHS_SAC2PLC2GW,BHS_SAC2PLC4GW) - The name list of services that need to be started.

Telegram Sample:

*"010500411234*BHS_SAC2PLC2GW,BHS_SAC2PLC4GW*"*

Description:

BHS Service Monitor (BHS_SvcMonitor) Service is running in the server machine to keep monitor the running status of each services that were registered in the XML configuration file. BHS_SvcMonitor service has the feature to start or stop the registered service upon receives the "Service Start Command" (0104, STR) message or "Service Stop Command" (0105, STO) message.

5.2.8 Parameter Changed Notification (0106)

Direction:

[ParamCHGNotification] Service (TCP Client) => SAC Component (TCP Server)

Alias Name:

PCN

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0106	Telegram Type.
	4-7	Numeric	4	0012	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0106 – Telegram type, Acknowledge telegram.

0012 – Telegram length,

(1234) – Sequence number. Its value will be each back from SAC component via Parameter Changed Notification Acknowledge (PNA) telegram.

Telegram Sample:

"010600121234"

Description:

[ParamCHGNotification] is a Windows service application to centralized monitors the parameter setting changes in the database table [SYSCONFIG]. If it detects that any parameter setting (the value of column "SYSVALUE") is changed, it will send the "Parameter Changed Notification" (PCN) telegram to SAC components. Once receives the PCN, the SAC component needs to reload the parameter new settings from [SYSCONFIG] table and use it in future.

In order to be notified by [ParamCHGNotification] service about the parameter setting changes, the component name, IP address and TCP port umber of the particular SAC component has to be pre-registered in the [ParamCHGNotification] service XML configuration file. The notification server will only send PCN telegram to those registered components.

There is a TCP connection is opened between notification service and each SAC components. The notification service acts as the TCP server, and initializes the TCP connection to the registered SAC components (TCP Servers).

Once receives the PCN telegram, the SAC component needs send "Parameter Changed Notification Acknowledge" (PNA) telegram back to the notification service. If the PNA is not received from the individual SAC component after the **"Notification Acknowledge Timeout" (configurable, default is 3 seconds)**, notification service resend the PCN telegram to that particular SAC component. This PCN resending process will be repeated for "**Notification Resending Times" (configurable, default is 3**

times). If the PNA is still not received after the retry, the application layer connection between [ParamCHGNotification] service and SAC component will be closed. The [ParamCHGNotification] service (TCP client) will re-initialize the connection. The same PCN telegram will be resent after the application layer connection is established.

5.2.9 Parameter Changed Notification Acknowledge (0107)

Direction:

SAC Component (TCP Server) => [ParamCHGNotification] Service (TCP Client)

Alias Name:

PNA

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0107	Telegram Type.
	4-7	Numeric	4	0012	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0107 – Telegram type, Acknowledge telegram.

0012 – Telegram length,

(1234) – Sequence number. It is the value in the received PCN telegram.

Telegram Sample:

"010700121234"

Description:

Upon receives the "Parameter Changed Notification" (PCN) telegram from [ParamCHGNotification] service, the SAC component is required to echo back the PCN sequence number via "Parameter Changed Notification Acknowledge" (PNA) telegram to the [ParamCHGNotification] service. Otherwise, the PCN telegram will be kept resending to SAC component.

5.2.10 Acknowledge (0099)

Direction:
TCP Server <=> TCP Client

Alias Name:
ACK

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0099	Telegram Type.
	4-7	Numeric	4	0012	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0099 – Telegram type, Acknowledge telegram.
0012 – Telegram length, 28 bytes.
(1234) – Sequence number.
The value is the echo back value of the sequence number field in the received acknowledgement required telegram, for example: Item Proceeded telegram.

Telegram Sample:
"009900121234"

Description:
Due to the critical to the SAC sortation control of some telegrams, for example the Item Proceeded telegram, the telegram sender must make sure the telegram is successfully received by receiver. Hence such items are designed to be the acknowledgement required telegram. After the telegram is sent, the sender must wait for the Acknowledge Telegram (0099), which contains the same sequence number as the original telegram. If no Acknowledge Telegram is returned from receiver within a certain time period (Acknowledgement Timeout, Default: 3000ms), the original telegram will be resent by sender. This process will be kept retried for preset number of times (Telegram Resend Times, Default: 3 times) before stop.

If the sender still does not receive the Acknowledge Telegram after the number of times retry, it will then stop all telegram sending, include the acknowledge un-required telegram and Keep-Alive telegram. If the sender is PLC, it then will wait for SAC to actively close the connections when the keep-alive receiving timeout. The SAC will re-establish the connection to PLC after certain time. If the sender is SAC, it then will close the connections immediately. After certain time (Connection Reinitialize Timeout, Default: 30000ms), the SAC will re-establish the connection to PLC.

After the application layer connection has been re-established, the sender will resend the original telegram to receiver. And the above procedure will be repeated until the sender receives the desired Acknowledge Telegram.

Parameters:

Parameter name	Default Values	Description
Acknowledgement Timeout	3000	Millisecond. **5.2.2 Application Layer Connection Confirm (0002).**
Telegram Resend Times	3	Times.
Connection Reinitialize Timeout	30000	Millisecond. Refer to section **5.2.2 Application Layer Connection Confirm (0002).**

6. APPLICATION PROTOCOL LAYER: KEEP-ALIVE

6.1 OVERVIEW

To ensure the application protocol layer connection validity both participating hosts must enforce a minimum level of traffic on the connection. This is done by means of a keep-alive telegram. Whenever a host has detected a quiet period (**Send Interval: 10seconds**) where it has not send anything for a configurable amount of time, it must send a keep-alive telegram to the other host. This ensures that no host is totally quiet for more than this configurable "keep-alive timeout".

Whenever a host finds it communication opponent quiet for more than this timeout, plus a margin, it must terminate the communication and disconnect. After an optionally pause (**Reconnect Timeout: 10seconds**) it must agree to reestablish the connection. Such connection related processes will only be issued by the application protocol layer.

The keep-alive protocol is illustrated in Figure 1 below.

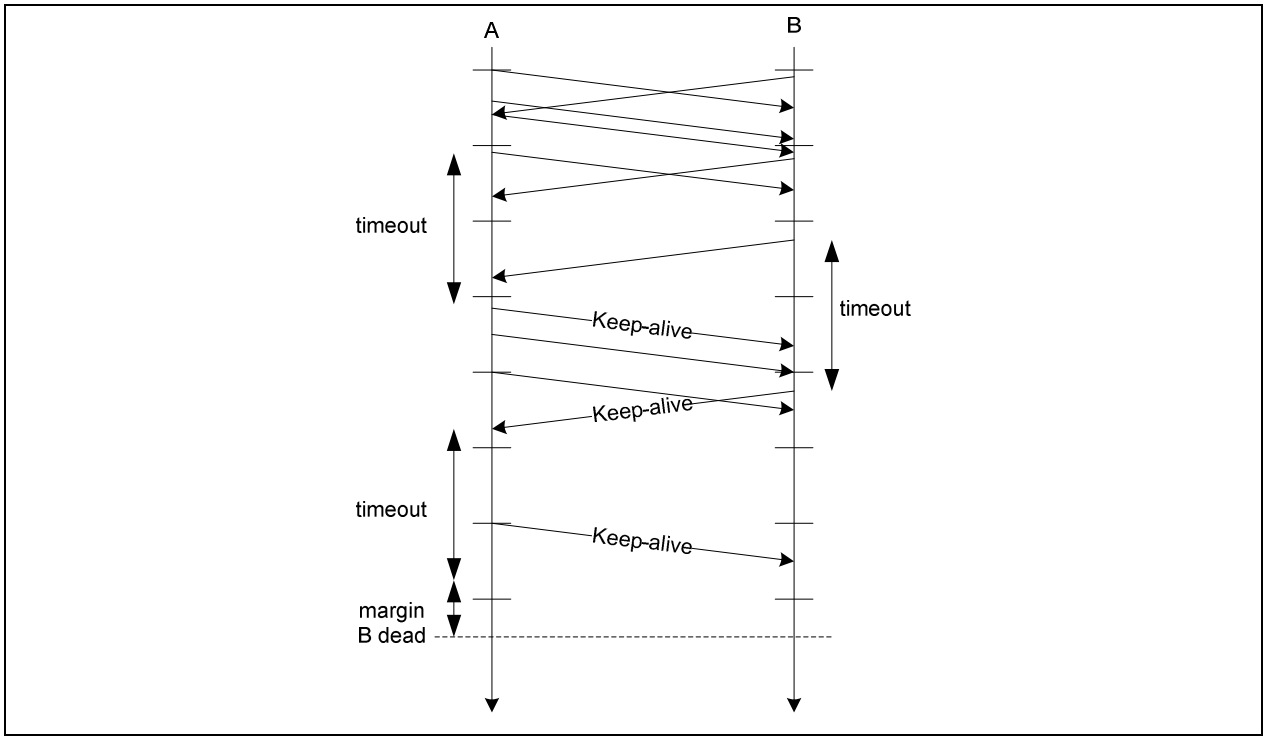

Figure 1: Ensuring connection validity by means of life sign telegrams. The unnamed directed lines indicate normal telegrams of random type, whereas the lines marked "keep-alive indicates telegrams of exactly that type.

As can be seen from the figure, peer A and peer B have a busy period, at the top of the figure, where they exchange telegrams regularly. At some point the communication ceases. At the bottom of the figure peer B fails to send either keep-alive or any other telegram resulting in A concluding the communication and deeming B dead.

The keep-alive telegram will be sent only when the bottom application layer connection has been established. And it will be stopped sending when application layer connected is interrupted.

6.2 Protocol Parameters

The table below lists the configurable parameters for the keep-alive protocol. These parameters must be decided and specified in the appropriate project documentation.

Parameter name	Values	Description
Receive_Timeout	25000	The timeout period in milliseconds. If keep-alive or other telegram is received within this period the connection must be closed and re-established.
Send_Interval	10000	The interval in milliseconds with which the keep-alive telegram are sent.
Reconnect_Timeout	10000	The timeout period in milliseconds. After the connection has been closed for this time period, the re-open connection will be performed.

6.3 Keep-Alive Telegram Definition

The keep-alive telegram is an acknowledgement un-required telegram.

Telegram Type	Telegram Name	Source	Destination	Acknowledge Required
0090	Keep-Alive Telegram	MES/PLC	PLC/MES	N

6.3.1 Keep-Alive Telegram (0090)

Direction:

MES <=> PLC

Alias Name:

SOL

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0090	Telegram Type.
	4-7	Numeric	4	0012	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.

Note: The value that is inside the brackets is the data sample of field. The value without brackets is the actual field data of the telegram.

0090 – Telegram type, Application layer connection request telegram.

0012 – Telegram length, 22 bytes.

(1234) – Sequence number, generated by telegram sender. The value varies according the sequence of telegram is created.

Telegram Sample:

"009000121234"

7. REFERENCES

Abbreviation	Reference
[BHS-503-01-1.00 DDS_SAC.doc]	SAC Detail Design Specification. Inter-Roller document ID: BHS-503-01-1.00 DDS_SAC
[BHS-503-04-1.00 DDS_DCSGW.doc]	DCS Gateway Detail Design Specification. Inter-Roller document ID: BHS-503-04-1.00 DDS_DCSGW
[BHS-503-07-1.00 DDS_IPAddress.doc]	BHS Network IP Address Assignment. Inter-Roller document ID: BHS-503-07-1.00 DDS_IPAddress
[BHS-503-09-1.00 DDS_SSIMSGW.doc]	SSIMS Gateway Detail Design Specification. Inter-Roller document ID: BHS-503-09-1.00 DDS_SSIMSGW
[BHS-504-01-1.10 IS_AP_ItemTracking.doc]	SAC to PLC interface Specification. Inter-Roller document ID: BHS-504-01-1.10 IS_AP_ItemTracking
[RFC-793-TCP]	RFC 793 – Transmission Control Protocol, DARPA Internet Program, Protocol specification, September 1982 http://www.faqs.org/rfcs/rfc793.html

8. APPENDIX

8.1 APPENDIX 1: BHS SOLUTION SUITE TELEGRAM LIST

Telegram Type	Telegram Alias Name	Telegram Name	Interface Protocol in Which Telegram is Defined
0001	CRQ	Application Layer Connection Request	[Item Tracking], [Manual Encoding], [TCPServer2Client]
0002	CCF	Application Layer Connection Confirm	[Item Tracking], [Manual Encoding], [TCPServer2Client]
0003	GID	GID Generated	[Item Tracking]
0004	ICR	Item Screened	[Item Tracking]
0005	ISC	Item Scanned	[Item Tracking]
0006	IRD	Item Redirect	[Item Tracking]
0007	ISE	Item Sortation Event	[Item Tracking]
0008	IPR	Item Proceeded	[Item Tracking]
0009	ILT	Item Lost	[Item Tracking]
0010	ITI	Item Tracking Information	[Item Tracking]
0011	CSR	Chute Status Request	[Item Tracking]
0012	CST	Chute Status Reply	[Item Tracking]
0013	IDR	Item Destination Request	[Item Tracking]
0099	ACK	Acknowledge	[Item Tracking], [Manual Encoding], [TCPServer2Client]
0101	SRQ	Running Status Request	[TCPServer2Client]
0102	SRP	Running Status Reply	[TCPServer2Client]
0103	BEV	Baggage Event	[TCPServer2Client]
0104	STR	Service Start Command	[TCPServer2Client]
0105	STO	Service Stop Command	[TCPServer2Client]
0106	PCN	Parameter Changed Notification	[TCPServer2Client]
0107	PNA	Parameter Changed Notification Acknowledge	[TCPServer2Client]
0201	IRY	Item Ready	[Manual Encoding]
0202	IEC	Item Encoded	[Manual Encoding]
0203	IRM	Item Removed	[Manual Encoding]