PTERIAL AIRPORT LOGISTIC SUITE (PALS)
FUNCTION DESIGN SPECIFICATION (FDS)
MESSAGEROUTER APPLICATION

Document No.:	PGL-105-07
Project No.:	HLC-Standardization
Author:	XJ
Release Date:	10-Jun-2009
Revision:	1.01
Pages:	40

Responsible for the contents

Pteris Global Limited

Singapore

Revision

Version	Release	Date	Init.	Description
1	00	23-May-2009	XJ	Initial version.
1	01	10-Jun-2009	XJ	• Add in “Connection Status Notification” (CSNF) message. • Add in “Gateway Ready” (GWRD) message.

Contents

Table of Figure

1 DESIGN OVERVIEW

1.1 Objective

It is very common in the industrial control system that one application needs communicate to multiple other applications concurrently. Among of these remote applications, some of them are TCP server and some of them are TCP client. In order to communicate with them, local application has to playing both TCP server and client roles. If one control system is consist of multiple network applications, and each one need to communicate with each other, then the application architecture become very complicated.

In order to simplify the complexity of application architecture, software development and testing work, **MessageRouter** application is added in between any two of connection applications as the intermediate layer. Instead of having the direct connections between multiple applications, all network applications now connect to the MessageRouter application for message sending and receiving.

MessageRouter is one application who is playing TCP server role. It listens on the TCP port **0x6666** (or **26214**) and is able to accept and maintain multiple connections with more than one TCP client applications concurrently. It receives the Intermediate (**INTM**) message from connected applications, and then route INTM message to its **Desired Receiver** that is defined in the INTM message, or to its **Subscribed Receivers** that is defined in the XML configuration file.

By introducing MessageRouter, each network application, who needs communicate with multiple other applications, can work as the TCP client and only need open a single TCP connection to MessageRouter without compromise the multiple communication requirements.

1.2 Intermediate (INTM) Message

Due to the different network applications could be connected to MessageRouter. And each application may support different communication protocols. It is very difficult and also is not necessary for MessageRouter to understand the different protocols used by the applications connected to it. Hence, the Intermediate (INTM) message is designed to solve this problem.

Before the network applications sending message to MessageRouter, they are required to encapsulate its original outgoing message to INTM message, which has following format (the details of message format will be covered soon in later chapters):

Field	Byte No	Format	Length (char)	Sample Value	Description
Header Fields	0-3	Alphanumeric	4	0103	Telegram Type.
	4-7	Numeric	4	(0044)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	12-19	Alphanumeric	8	(SAC2PLC1)	The sender of original message.
	20-27	Alphanumeric	8	(SORTENGN)	The receiver of original message.
	28-31	Alphanumeric	4	(0011)	The type of Original message.
	32-?	Alphanumeric	?	(001100121234)	Encapsulated Original message.

By knowing original message **Sender**, **Receiver**, and **Original Message Type**, MessageRouter has been able to route incoming messages without bother the format of original message.

The receiver that is defined by the INTM message "Receiver" field is called "**Desired Receiver**". The value of "Original Message Type field will be used to identify the applications who has subscribed this type of message in the XML configuration file of MessageRouter application. Such applications are called "**Subscribed Receiver**". One message could be subscribed by multiple applications. One INTM message will be routed to its Desired Receiver and all Subscribed Receiver.

Hence, the INTM message is the only valid message recognized by MessageRouter for message receiving incoming messages and sending outgoing messages. All other format messages will be classified as the invalid message and be discarded immediately by MessageRouter.

1.3 Message Subscription

In some case, one message that is generated by one application (sender) may need to be sent to multiple applications (receivers). In the INTM message format, there is one single receiver can be assigned. But MessageRouter is still able to route this message to other receivers if they have been registered in the MessageRouter application XML configuration file, also called as Message Subscription.

Below is the example XML configuration of message subscription:

```
<nodes>
  <node>
    <name>SAC2PLC1</name>
    <messages>0101,0301,0302,0303</messages>
    <dependingNodes>SORTENGN</dependingNodes>
    <affectingNodes></affectingNodes>
  </node>
  <node>
    <name>SAC2PLC2</name>
    <messages>0101,0301,0302,0303</messages>
    <dependingNodes>SORTENGN</dependingNodes>
    <affectingNodes></affectingNodes>
  </node>
  <node>
    <name>SORTENGN</name>
    <messages>0101,0304,0305</messages>
    <dependingNodes></dependingNodes>
    <affectingNodes>SAC2PLC1,SAC2PLC2</affectingNodes>
  </node>
  <node>
    <name>PALSTEST</name>
    <messages>0101,0301,0302,0303,0304,0305</messages>
  </node>
</nodes>
```

Figure 1: Message subscription example

As illustrated above example, all SAC2PLC1, SAC2PLC2, SORTENGN, and PALSTEST 4 applications have subscribed for the message "0101". Once MessageRouter receives "0101" message, it will route it to these 4 applications immediately. The "0301" message will be route to SAC2PLC1, SAC2PLC2, and PALSTEST 3 applications, but not to SORTENGN application.

If one message has no any subscribers, then it will only be router to its desired receiver, not to subscribed receiver.

1.4 Application Code

In order for MessageRouter to properly route message, all connected network applications (message senders and receivers) must be identified by a unique name, called **Application Code**.

Application code is the string, which length can be from 3 to 8 characters. Any visible characters can be used for the application code, as long as the string is unique among all connected network applications.

The application code is also used for message subscription in the XML configuration file. The same application registered in following two sections, AppServer and MessageHandler, in the XML configuration must has the same name.

```
  <configSet name="PALS.Net.Filters.Application.AppServer">
    <threadInterval>100</threadInterval>
    <connectionRequestTimeout>3000</connectionRequestTimeout>
    <minSequenceNo>1</minSequenceNo>
    <maxSequenceNo>9999</maxSequenceNo>
    <clients>
      <!--The max length of client application code is 8.-->
      <appCode>PALSTEST</appCode>
      <appCode>SAC2PLC1</appCode>
      <appCode>SAC2PLC2</appCode>
      <appCode>SORTENGN</appCode>
    </clients>
  </configSet>

  <configSet name="BHS.Router.TCPServerChains.Messages.Handlers.MessageHandler">
    <sender>MSGROUTE</sender>
    <nodes>
      <node>
        <name>SAC2PLC1</name>
        <messages>0101,0301,0302,0303</messages>
        <dependingNodes>SORTENGN</dependingNodes>
        <affectingNodes></affectingNodes>
      </node>
      <node>
        <name>SAC2PLC2</name>
        <messages>0101,0301,0302,0303</messages>
        <dependingNodes>SORTENGN</dependingNodes>
        <affectingNodes></affectingNodes>
      </node>
      <node>
        <name>SORTENGN</name>
        <messages>0101,0304,0305</messages>
        <dependingNodes></dependingNodes>
        <affectingNodes>SAC2PLC1,SAC2PLC2</affectingNodes>
      </node>
      <node>
        <name>PALSTEST</name>
        <messages>0101,0301,0302,0303,0304,0305</messages>
      </node>
    </nodes>
  </configSet>
```

Figure 2: Application Code configuration example

In above XML configurations, there are 4 applications are registered as the remote client and have subscribed the different messages:

- **PALSTEST** - PALS tester that has single TCP connected to MessageRouter. It is designed to subscribe (receive) all messages sent by all applications for testing and debugging purpose.
- **SAC2PLC1** - PLC interface gateway application that interfaces to both MessageRouter and to PLC 1. It is designed to receive message sent by PLC and forward them to Sort Engine service via message routing service provided by MessageRouter.
- **SAC2PLC2** - PLC interface gateway application that interfaces to both MessageRouter and to PLC 2. It is designed to receive message sent by PLC and forward them to Sort Engine service via message routing service provided by MessageRouter.
- **SORTENGN** - PALS internal Sortation Engine service application that has single TCP connected to MessageRouter. It is responsible for business logic handling of original messages sent by PLCs.

1.5 DependingNodes & AffectingNodes Properties

Each application has two properties, **DependingNodes** and **AffectingNodes**. Their value could be single node name or multiple node names separated by comma as illustrated in the sample below.

```
<configSet name="BHS.Router.TCPServerChains.Messages.Handlers.MessageHandler">
  <sender>MSGROUTE</sender>
  <nodes>
    <node>
      <name>SAC2PLC1</name>
      <messages>0101,0301,0302,0303</messages>
      <dependingNodes>SORTENGN</dependingNodes>
      <affectingNodes></affectingNodes>
    </node>
    <node>
      <name>SAC2PLC2</name>
      <messages>0101,0301,0302,0303</messages>
      <dependingNodes>SORTENGN</dependingNodes>
      <affectingNodes></affectingNodes>
    </node>
    <node>
      <name>SORTENGN</name>
      <messages>0101,0304,0305</messages>
      <dependingNodes></dependingNodes>
      <affectingNodes>SAC2PLC1,SAC2PLC2</affectingNodes>
    </node>
    <node>
      <name>PALSTEST</name>
      <messages>0101,0301,0302,0303,0304,0305</messages>
    </node>
  </nodes>
</configSet>
```

Figure 3: DependingNodes & AffectingNodes configuration example

If one application want to connect to MessageRouter, it has to wait until it's all depending nodes have connected to the MessageRouter first. E.g. Sortation Engine application has to connect to MessageRouter first, before all SAC2PLC gateway applications can open the connection to

MessageRouter. Otherwise, MessageRouter will actively close the application layer and socket layer connection to any SAC2PLC gateway application immediately once it is opened.

If one application lost the connection to MessageRouter, then MessageRouter will close its connections to all affecting nodes of disconnected application immediately.

This design is used to solve the deadlock problem in the scenario of Engine application lost the connection to MessageRouter but its associated Gateway applications are still connected to MessageRouter. When this case is happened, the PLC will still sending message to PLC Gateway application, but these message will no longer be routed to Sortation Engine application for sortation control.

There are few rules need to be followed when configure “DependingNodes” & “AffectingNodes” properties. They are:

- Any one application is not allowed to open the connection to MessageRouter if its depending node has not connected to MessageRouter yet.
- When one application lost the connection to MessageRouter, the connections between its affecting nodes and MessageRouter will close immediately by MessageRouter.
- <dependingNode> and <affectingNode> could contain more than one node names that are separated by comma.
- <dependingNode> and <affectingNode> can not point to the application itself.
- If one node name has been assigned to <dependingNode>, then it is not allowed to be reassigned to <affectingNode> any more.
- <dependingNode> and <affectingNode> can be omitted in the XML configuration. If do so, the empty string will be assigned to them.

1.6 Message Routing Rules

Following rules are defined for MessageRouter application to do the message routing:

- INTM message will always be routed to its “Desired Receiver” first before they are routed to “Subscribed Receiver”;
- INTM message will not be routed to its “Desired Receiver” if the same application code is assigned to both “Sender” and “Receiver” fields. But its will still be routed to its “Subscribed Receiver”.
- If message has been routed to its “Desired Receiver”, then it will not be routed to the same application twice even though this application has the subscription in the XML configuration file;
- If the receiver has not connected to MessageRouter, then the message that is being routed to this particular receiver will be discarded immediately;

1.7 Connection Status Notification (CSNF)

By using MessageRouter as the intermediate layer, there is no direct connection between any pair of interface parties, for example between PLC Gateway and Sort Engine services. When Engine service connected to MessageRouter, Gateway service may not connected to MessageRouter too. It may result in some Engine initialization tasks, which are required to be performed upon Engine-Gateway connection is opened, failure. Hence, the “Connection Status Notification” (CSNF) message is designed to address this requirement.

Whenever the connection between an application and MessageRouter is opened or closed, MessageRouter will produce the CSNF message and send to relevant interface parties (Engine and Gateway) automatically. There are below 2 scenarios of producing CSNF message:

- **When application (A) is connected to MessageRouter –**

  MessageRouter will check whether this application **A** has depending node and affecting nodes. If doesn’t have, then no CSNF message will be produced and sent.

  If application **A** has depending or affecting nodes, then MessageRouter will check the connection status of those nodes. If is there any depending or affecting nodes has connected to MessageRouter, then MessageRouter will

  1) Generate CSNF message that includes **A**’s application code and send to connection opened depending or affecting nodes to inform them that application **A** is online;
  2) Generate CSNF message that includes application code of connection opened depending or affecting nodes’ and send to application **A** to inform it that its associated depending/affecting node is online.

  Hence, each connected depending or affecting node will receive one CSNF message, but application **A** could receive multiple CSNF message if there are multiple connected depending or affecting nodes.

  If application **A** has depending or affecting nodes, but none of them is connected to MessageRouter, then no CSNF message will be produced.

- **When application (A) is disconnected to MessageRouter –**

  MessageRouter will check whether this application **A** has depending nodes. If doesn’t, then no CSNF message will be produced. Affecting node will not be checked as they will be disconnected immediately by MessageRouter upon application **A** is disconnected.

  If application **A** has depending nodes, then MessageRouter will check the connection status of those nodes. If there is any depending nodes has connected to MessageRouter, then MessageRouter will generate CSNF message that includes **A**’s application code and send to connection opened depending nodes to inform them that application **A** is offline. No CSNF message is sent to application **A** as it has been disconnected to MessageRouter.

  If application **A** has depending nodes, but none of them is connected to MessageRouter, then no CSNF message will be produced.

1.8 Gateway Ready Notification (GRNF)

In the PGL interface application design, all of external devices (e.g. PLC), PALS Gateway, and PALS Engine, PALS MessageRouter applications are required to participate in one interface communication. Gateway connects to both external devices and MessageRouter as well. It receives message sent by PLC and forwards to MessageRouter for routing message to its final receiver, which is Engine application, for business logic handling. Or it receives messages (e.g. Sorting Destination Reply), which is produced by Engine application and sent to MessageRouter, from MessageRouter and forward to external devices. Hence, Gateway application always maintains two connections, one to external device, and another to MessageRouter. Engine application needs to maintain single connection to MessageRouter only.

Gateway application can be classified as ready for communication only when its two connections to both external device and MessageRouter are opened successfully. Before Gateway is connected to external devices, Engine application should not send any message to external device. In order to achieve this, "**Gateway Ready Notification**" (**GRNF**) message is designed.

Upon successfully connected to external device and MessageRouter, Gateway application will produce a GRNF message and send to its associated Engine application. Engine application can only send messages, which need to be forwarded to external device by Gateway, after receives the GRNF message from Gateway.

1.9 Challenges

In the control system architecture, MessageRouter is playing the critical mission. As per the control system configuration of respective projects, the number of applications connected to it may be many.

For example, if one BHS project has 40 PLCs, then there will be minimum 40 SAC2PLC gateway applications, plus one SAC SortEngine application, totally 41 applications, need connected to the MessageRouter.

Another example, if one Catering project has 10 PLCs, 8 Cranes, 20 CMS GUI Workstations, then there will be 10 CMS2PLC gateway applications, 1 PLC Engine application, 8 CMS2Crane gateway application, 1 Crane Engine application, 20 CMS GUI applications, and 1 CMS SCADA engine, totally 41 applications, need connected to the MessageRouter.

Hence, to maintain the high performance and reliability will be the essential and challenge requirements. Hence, the thoroughly testing and continuously improvement is required.

1.10 Limitations and Future Improvements

The current MessageRouter application design has following limitations and needs further improvement for it to become a "**perfect**" MessageRouter.

- **Message Queuing –**

  In the current MessageRouter design and implementation, in order to maintain the high performance, all messages will be discarded immediately if its receiver is not connected to MessageRouter at the time when message is sent.

  To prevent the loss of messages in the case of interruption of network connection between MessageRouter and message receivers, all outgoing message may need to be buffered into a queue. And resending those messages to its receiver upon the connection is recovered.

2 ARCHITECTURE DESIGN

2.1 Communication System Architecture

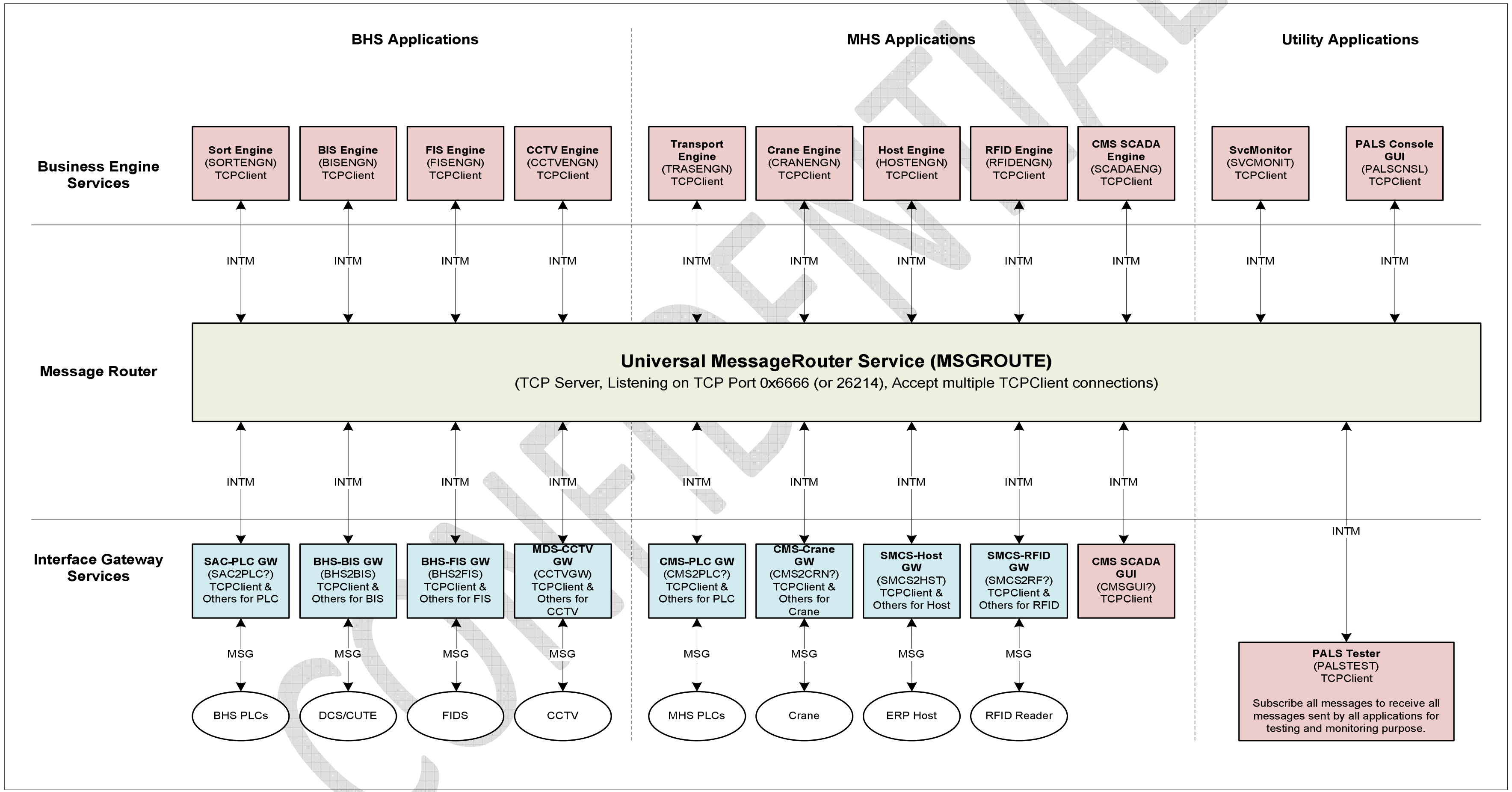
BHS Applications
MHS Applications
Utility Applications
Business Engine Services
Sort Engine (SORTENGN) TCPClient
BIS Engine (BISENGN) TCPClient
FIS Engine (FISENGN) TCPClient
CCTV Engine (CCTVENGN) TCPClient
Transport Engine (TRASENGN) TCPClient
Crane Engine (CRANENGN) TCPClient
Host Engine (HOSTENGN) TCPClient
RFID Engine (RFIDENGN) TCPClient
CMS SCADA Engine (SCADAENG) TCPClient
SvcMonitor (SVCMONIT) TCPClient
PALS Console GUI (PALSCNSL) TCPClient
INTM
Message Router
Universal MessageRouter Service (MSGROUTE)
(TCP Server, Listening on TCP Port 0x6666 (or 26214), Accept multiple TCPClient connections)
Interface Gateway Services
SAC-PLC GW (SAC2PLC?) TCPClient & Others for PLC
BHS-BIS GW (BHS2BIS) TCPClient & Others for BIS
BHS-FIS GW (BHS2FIS) TCPClient & Others for FIS
MDS-CCTV GW (CCTVGW) TCPClient & Others for CCTV
CMS-PLC GW (CMS2PLC?) TCPClient & Others for PLC
CMS-Crane GW (CMS2CRN?) TCPClient & Others for Crane
SMCS-Host GW (SMCS2HST) TCPClient & Others for Host
SMCS-RFID GW (SMCS2RF?) TCPClient & Others for RFID
CMS SCADA GUI (CMSGUI?) TCPClient
PALS Tester (PALSTEST) TCPClient
Subscribe all messages to receive all messages sent by all applications for testing and monitoring purpose.
MSG
BHS PLCs
DCS/CUTE
FIDS
CCTV
MHS PLCs
Crane
ERP Host
RFID Reader

Figure 4: Communication System Architecture

2.2 MessageRouter Application Architecture

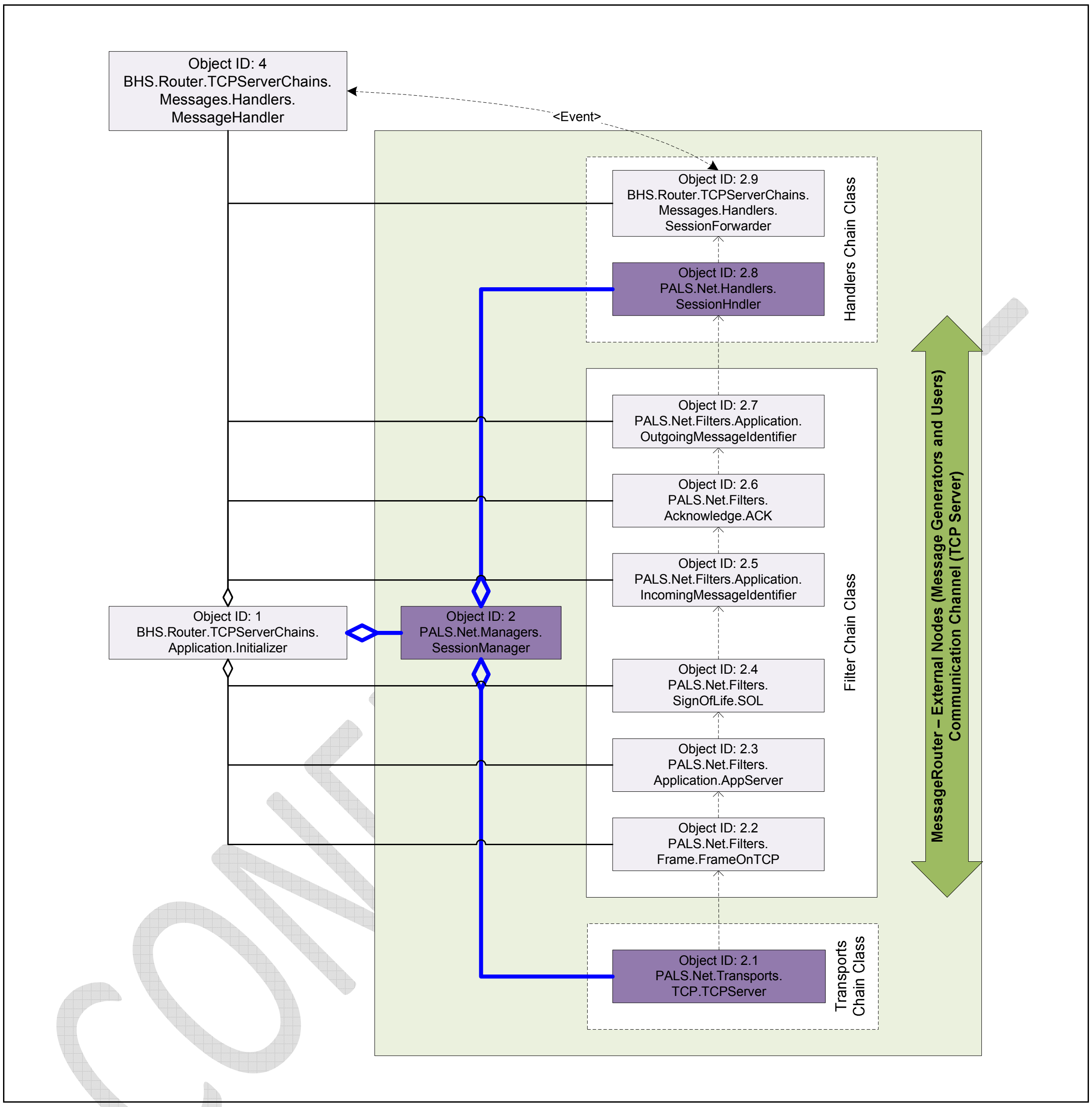

Figure 5: MessageRouter Application Architecture

3 COMMINUCATION PROTOCOL

3.1 Protocol Layers

The communication protocol between MessageRouter and network applications is 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 6: Overview of the transport protocol layers.

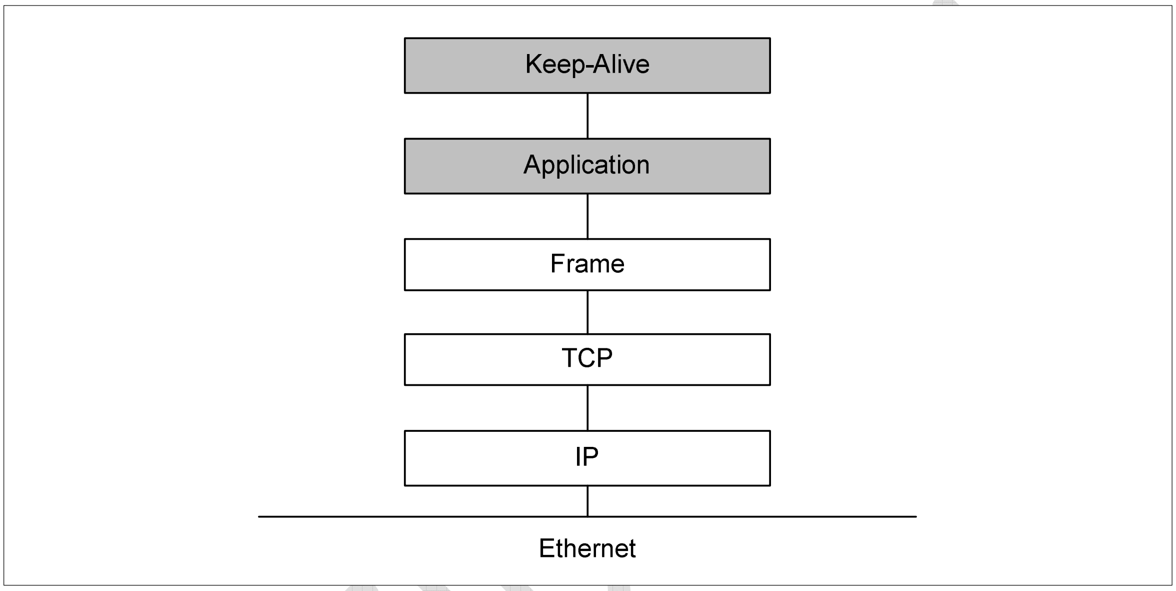

Figure 6: Overview of the transport protocol layers.

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 a 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.

3.2 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 MessageRouter and network application 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
PALS.MessageRouter	MSGROUTE	TBA	26214	MessageRouter Service Application, TCP Server

PALS.SAC2PLC?GW	SAC2PLC?	TBA	Auto	SAC-PLC Gateway Service Applications.
				? – represents the running number
				TCP Client for MessageRouter connection;
				TCP Server/Client for PLC connection;

PALS.SortEngine	SORTENGN	TBA	Auto	SAC Sortation Engine Service Applications. TCP Client for MessageRouter connection;

PALS.BHS2BISGW	SAC2BIS	TBA	Auto	BHS-BIS Gateway Service Applications.
				TCP Client for MessageRouter connection;
				TCP Server/Client for BIS connection;

PALS.BISEngine	BISENGN	TBA	Auto	BIS Engine Service Applications. TCP Client for MessageRouter connection;

PALS.BHS2FISGW	SAC2FIS	TBA	Auto	BHS-FIS Gateway Service Applications.
				TCP Client for MessageRouter connection;
				TCP Server/Client for FIS connection;

PALS.FISEngine	FISENGN	TBA	Auto	FIS Engine Service Applications. TCP Client for MessageRouter connection;

PALS.MDS2CCTVGW	CCTVGW	TBA	Auto	MDS-CCTV Gateway Service Applications.
				TCP Client for MessageRouter connection;
				TCP Client for CCTV controller connection;

PALS.CCTVEngine	CCTVENGN	TBA	Auto	CCTV Engine Service Applications. TCP Client for MessageRouter connection;

PALS.CMS2PLC?GW	CMS2PLC?	TBA	Auto	CMS-PLC Gateway Service Applications.
				? – represents the running number
				TCP Client for MessageRouter connection;
				TCP Server/Client for PLC connection;

PALS.TransEngine	TRASENGN	TBA	Auto	Conveyor Transport Engine Service Applications. TCP Client for MessageRouter connection;

PALS.CMS2Crane?GW	CMS2CRN?	TBA	Auto	CMS-Crane Gateway Service Applications.
				? – represents the running number
				TCP Client for MessageRouter connection;
				TCP Client for Crane connection;

PALS.CraneEngine	CRANENGN	TBA	Auto	Crane Engine Service Applications. TCP Client for MessageRouter connection;

PALS.SMCS2HostGW	SMCS2HST	TBA	Auto	SMCS-Host Gateway Service Applications.
				TCP Client for MessageRouter connection;
				TCP Server/Client for Host connection;

PALS.HostEngine	HOSTENGN	TBA	Auto	Host Engine Service Applications. TCP Client for MessageRouter connection;

PALS.SMCS2RFID?GW	SMCS2RF?	TBA	Auto	SMCS-RFID Gateway Service Applications.
				? – represents the running number
				TCP Client for MessageRouter connection;
				TCP Client for RFID reader connection;

PALS.RFIDEngine	RFIDENGN	TBA	Auto	RFID Engine Service Applications. TCP Client for MessageRouter connection;
PALS.CMSSCADAGUI	CMSGUI?	TBA	Auto	CMS SCADA GUI Applications. ? – represents the running number TCP Client for MessageRouter connection;
PALS.SCADAEngine	SCADAENG	TBA	Auto	CMS SCADA Engine Service Applications. TCP Client for MessageRouter connection;
PALS.SvcMonitor	SVCMONIT	TBA	Auto	PALS Service Monitor Service Applications. TCP Client for MessageRouter connection;
PALS.ConsoleGUI	PALSCNSL	TBA	Auto	PALS Console GUI Applications. TCP Client for MessageRouter connection;
PALS.Tester	PALSTEST	TBA	Auto	PALS Network Tester GUI Applications. TCP Client for MessageRouter connection;

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 7and Figure 8 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

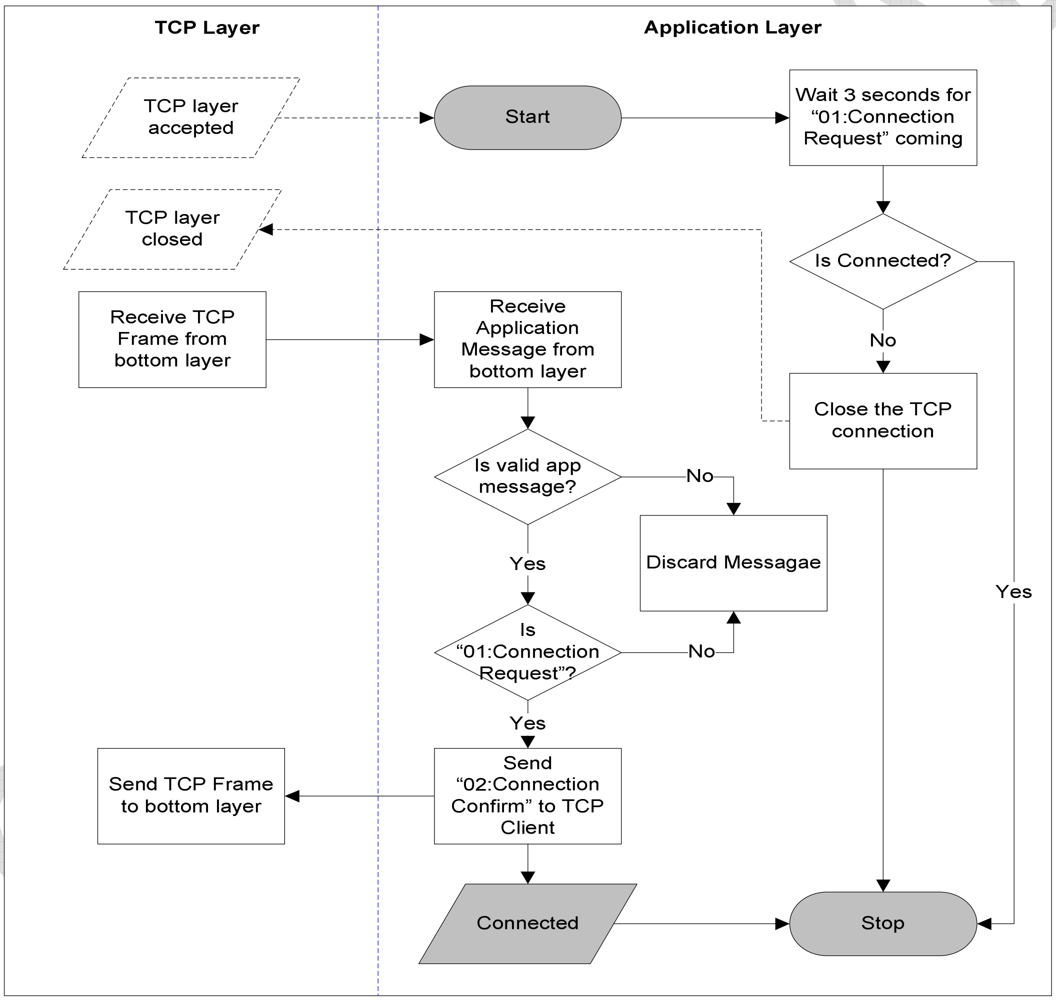
TCP Layer
Application Layer
TCP layer accepted
Start
Wait 3 seconds for "01:Connection Request" coming
TCP layer closed
Is Connected?
Receive TCP Frame from bottom layer
Receive Application Message from bottom layer
No
Close the TCP connection
Is valid app message?
No
Discard Messagae
Yes
Yes
Is "01:Connection Request"?
No
Yes
Send TCP Frame to bottom layer
Send "02:Connection Confirm" to TCP Client
Connected
Stop

Figure 7: TCP Server application layer connection initialization sequence.

As illustrated in the Figure 7, 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 **3 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

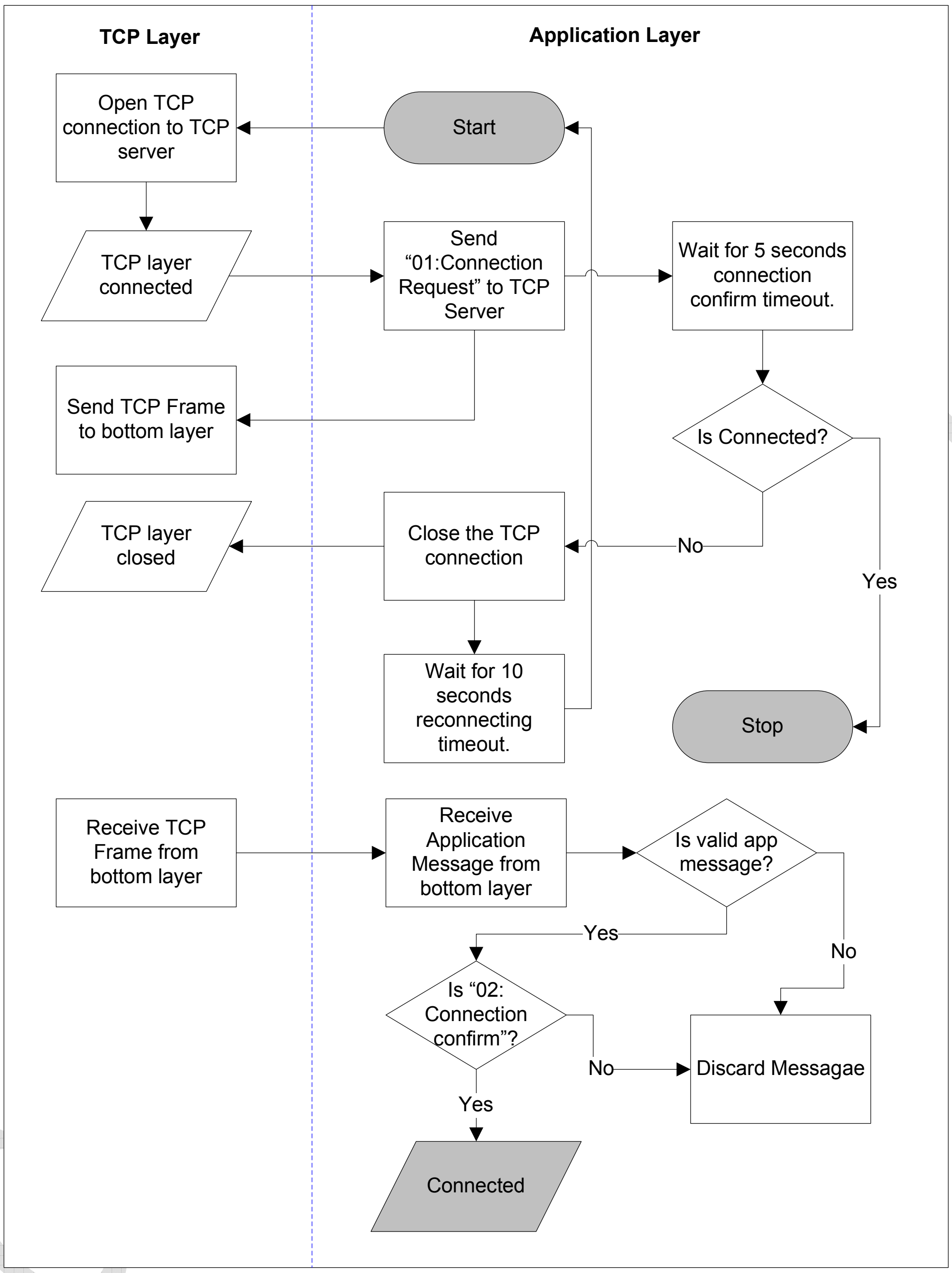
TCP Layer
Application Layer
Open TCP connection to TCP server
Start
TCP layer connected
Send "01:Connection Request" to TCP Server
Wait for 5 seconds connection confirm timeout.
Send TCP Frame to bottom layer
Is Connected?
TCP layer closed
Close the TCP connection
No
Yes
Wait for 10 seconds reconnecting timeout.
Stop
Receive TCP Frame from bottom layer
Receive Application Message from bottom layer
Is valid app message?
Yes
No
Is "02: Connection confirm"?
No
Discard Messagae
Yes
Connected

Figure 8: TCP client application layer connection initialization sequence.

As shown in the Figure 8, 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 20 **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 **20 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	Application Connection Request	CRQ	TCP Client	TCP Server	No
2	0002	Application Connection-confirm	CCF	TCP Server	TCP Client	No
3	0101	Service Running Status Request (for future implementation)	SRQ	TCP Client	TCP Server	No
4	0102	Service Running Status Reply (for future implementation)	SRP	TCP Server	TCP Client	No
5	0103	Intermediate Message	INTM	Both	Both	**Yes**
6	0108	Connection Status Notification Message	CSNF	Message Router	Depending and Affecting Nodes	**Yes**
7	0090	Acknowledge	ACK	Both	Both	No
8	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, 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 Request (0001)

Direction:

TCP Client => TCP Server

Alias Name:

CRQ

Acknowledgement Required:

No

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.

Refer to **4.1.1: TCP/IP Protocol Parameters** for application code of different applications..

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

Acknowledgement Required:

No

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:

“000200200001SORTENGN”

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)**	20000	Millisecond.

5.2.3 Intermediate Message (0103)

Direction:

TCP Server <=> TCP Client

Alias Name:

INTM

Acknowledgement Required:

Yes

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0103	Telegram Type.
	4-7	Numeric	4	(0044)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	12-19	Alphanumeric	8	(SAC2PLC1)	The sender of original message.
	20-27	Alphanumeric	8	(SORTENGN)	The receiver of original message.
	28-31	Alphanumeric	4	(0011)	The type of Original message.
	32-?	Alphanumeric	?	(001100121234)	Original 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.
(0044) –	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.
(SAC2PLC1) –	The sender of original message. Succeeding pad with space character (Hex: 0x20) if the sender code less than 10 characters.
(SORTENGN) –	The receiver of original message. Succeeding pad with space character (Hex: 0x20) if the sender code less than 10 characters. If the message is sent to MessageRouter for routing to desired receiver, then the desired receiver name shall be included here, instead of MessageRouter name.
(0011) –	The type of Original message. Due to the original message could come from various source, their message format (protocol) could be different. To include original message type here is to simplify the original message type identification done by Message Router. So that the MessageRouter can route the message by using original message type as defined by this field without bother the message format of original messages.

(001100121234) – Original message. The full data package of original application layer message.

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

Telegram Sample:

"010300401234SAC2PLC1SORTENGN0011001100121234"

Description:

Intermediate Message (INTM) is used for exchanging message between Interface Gateway services and business logic handler (Engine) services, or between above two services to MessageRouter service.

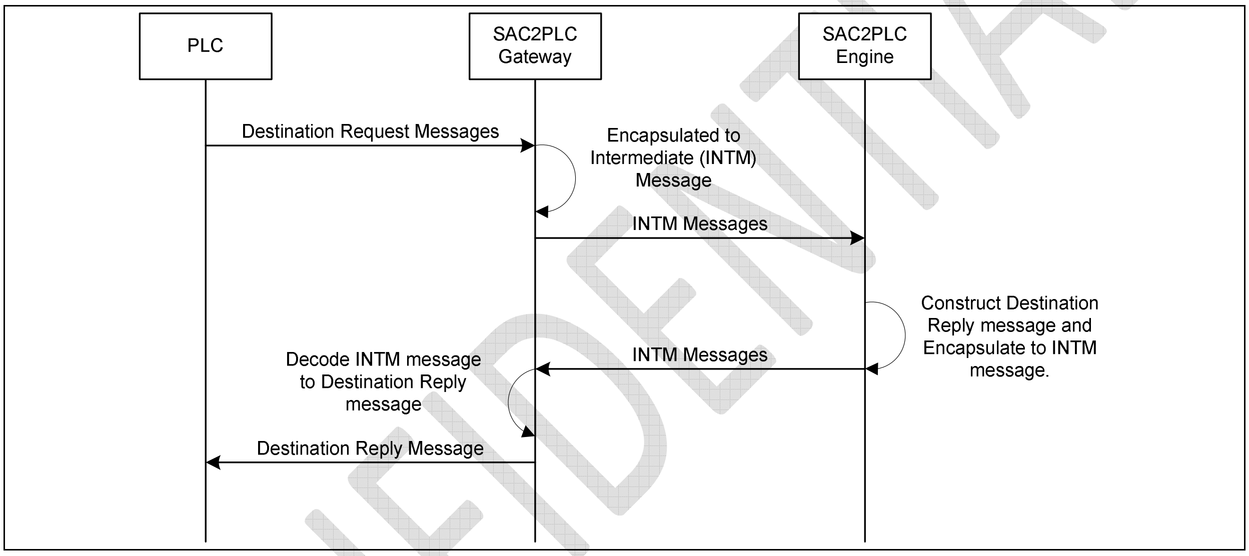
PLC
SAC2PLC Gateway
SAC2PLC Engine
Destination Request Messages
Encapsulated to Intermediate (INTM) Message
INTM Messages
Construct Destination Reply message and Encapsulate to INTM message.
INTM Messages
Decode INTM message to Destination Reply message
Destination Reply Message

Figure 9: Scenario 1 (Exchange INTM message between Gateway-Engine services).

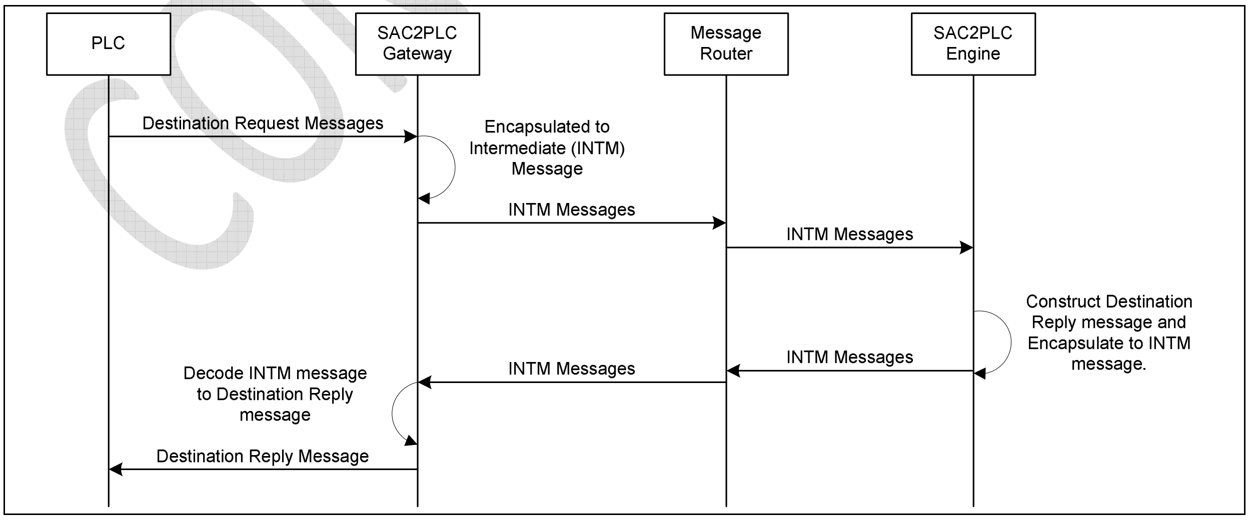
PLC
SAC2PLC Gateway
Message Router
SAC2PLC Engine
Destination Request Messages
Encapsulated to Intermediate (INTM) Message
INTM Messages
INTM Messages
Construct Destination Reply message and Encapsulate to INTM message.
INTM Messages
INTM Messages
Decode INTM message to Destination Reply message
Destination Reply Message

Figure 10: Scenario 2 (Exchange INTM message between Gateway-Router-Engine services).

The incoming application layer business message received from external devices (e.g. PLCs, CCTV, etc.) will be encapsulated to Intermediate (INTM) message by Gateway service before they are transferred to Engine Services or MessageRouter service. Gateway service will extract the original message from INTM message and send to external devices.

Due to one Engine service or MessageRouter service services multiple Gateway service, instead of just forwarding the original message from Gateway to Engine or Router services, the Intermediate Message, therefore, has to be used to inform Engine or Router servers the original message sender and its desired receiver.

MessageRouter service will route the INTM message between various connected services depends on the type of original message. In order to simplify the original message type identification work, the constructors of INTM message is required to assign the original message type value to one field of INTM message separated from original message field. It enables MessageRouter to identify the original message type without knowing its message format.

INTM is Acknowledgement required message.

5.2.4 Connection Status Notification Message (0108)

Direction:

MessageRouter => Connected Applications

Alias Name:

CSNF

Acknowledgement Required:

Yes

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0108	Telegram Type.
	4-7	Numeric	4	(0022)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	12-19	Alphanumeric	8	(SAC2PLC1)	Application code of application whose connection is just opened.
	20-21	Alphanumeric	2	(01)	Status of connection.

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.

0108 –	Telegram type, Acknowledge telegram.
(0020) –	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.
(SAC2PLC1) –	Application code of application whose connection is just opened.
(01) –	Connection event. There are two valid values: • **00** – Connection Closed; • **01** – Connection Opened.

Telegram Sample:

"010800221234SAC2PLC101"

Description:

By using MessageRouter as the intermediate layer of network communications, the original message sender and its final receiver has no direct connection. Therefore, they will not be notified when its dependent communication party (message receiver) open or lost the network connection to

MessageRouter. In this case, the message sender will still keep sending message to MessageRouter, but they are not be able to route to receiver any more.

In another scenario, some applications are required to perform some initialization tasks or sending initialization message at the time when it connected to its final interface, e.g. sending current MDS alarms to CCTV controller or sending current CCTV alarms to CCTV Engine whenever CCTV Gateway is connected to CCTV Engine applications. Such requirements will become the issues if there is no direct connection between both final end interface parties.

Hence, the "Connection Status Notification" (CSNF) message is designed to address this requirement. Whenever the connection between an application and MessageRouter is opened or closed, MessageRouter will produce the CSNF message and send to relevant interface parties (Engine and Gateway) automatically. There are below 2 scenarios of producing CSNF message:

- **When application (A) is connected to MessageRouter –**

  MessageRouter will check whether this application **A** has depending node and affecting nodes. If doesn't have, then no CSNF message will be produced and sent.

  If application **A** has depending or affecting nodes, then MessageRouter will check the connection status of those nodes. If is there any depending or affecting nodes has connected to MessageRouter, then MessageRouter will

  1) Generate CSNF message that includes **A**'s application code and send to connection opened depending or affecting nodes to inform them that application **A** is online;
  2) Generate CSNF message that includes application code of connection opened depending or affecting nodes' and send to application **A** to inform it that its associated depending/affecting node is online.

  Hence, each connected depending or affecting node will receive one CSNF message, but application **A** could receive multiple CSNF message if there are multiple connected depending or affecting nodes.

  If application **A** has depending or affecting nodes, but none of them is connected to MessageRouter, then no CSNF message will be produced.

- **When application (A) is disconnected to MessageRouter –**

  MessageRouter will check whether this application **A** has depending nodes. If doesn't, then no CSNF message will be produced. Affecting node will not be checked as they will be disconnected immediately by MessageRouter upon application **A** is disconnected.

  If application **A** has depending nodes, then MessageRouter will check the connection status of those nodes. If there is any depending nodes has connected to MessageRouter, then MessageRouter will generate CSNF message that includes **A**'s application code and send to connection opened depending nodes to inform them that application **A** is offline. No CSNF message is sent to application **A** as it has been disconnected to MessageRouter.

  If application **A** has depending nodes, but none of them is connected to MessageRouter, then no CSNF message will be produced.

CSNF message is sent from MessageRouter to those applications connected to it. Since it is sent by MessageRouter directly, instead of Routed by it, CSNF message will not be encapsulated into INTM message format.

5.2.5 Gateway Ready Notification Message (0109)

Direction:

Gateway Application => Engine Application

Alias Name:

GRNF

Acknowledgement Required:

Yes

Format:

Field	Byte No	Format	Length (char)	Value	Description
Header Fields	0-3	Alphanumeric	4	0109	Telegram Type.
	4-7	Numeric	4	(0020)	Telegram Length.
	8-11	Numeric	4	(1234)	Telegram Sequence Number.
Data Fields	12-19	Alphanumeric	8	(SAC2PLC1)	Application code of Gateway application whose connection is just opened.

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.

0109 –	Telegram type, Acknowledge telegram.
(0020) –	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.
(SAC2PLC1) –	Application code of interface Gateway application who is ready for communication.

Telegram Sample:

"010900201234SAC2PLC1"

Description:

In the PGL interface application design, all of external devices (e.g. PLC), PALS Gateway, and PALS Engine, PALS MessageRouter applications are required to participate in one interface communication. Gateway connects to both external devices and MessageRouter as well. It receives message sent by PLC and forwards to MessageRouter for routing message to its final receiver, which is Engine application, for business logic handling. Or it receives messages (e.g. Sorting Destination Reply), which is produced by Engine application and sent to MessageRouter, from MessageRouter and forward to external devices. Hence, Gateway application always maintains two connections, one to external device, and another to MessageRouter. Engine application needs to maintain single connection to MessageRouter only.

Gateway application can be classified as ready for communication only when its two connections to both external device and MessageRouter are opened successfully. Before Gateway is connected to external devices, Engine application should not send any message to external device. In order to achieve this, “**Gateway Ready Notification**” (**GRNF**) message is designed.

Upon successfully connected to external device and MessageRouter, Gateway application will produce a GRNF message and send to its associated Engine application. Engine application can only send messages, which need to be forwarded to external device by Gateway, after receives the GRNF message from Gateway.

GRNF message is sent from Gateway application to Engine application only. It is required to be routed by MessageRouter, therefore, CRNF message needs to be encapsulated into INTM message by Gateway before sending out to MessageRouter.

5.2.6 Acknowledge (0099)

Direction:

TCP Server <=> TCP Client

Alias Name:

ACK

Acknowledgement Required:

No

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 (Reconnecting Timeout, Default: 20000ms), 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.
Reconnection Timeout	20000	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: 20seconds**) 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 11 below.

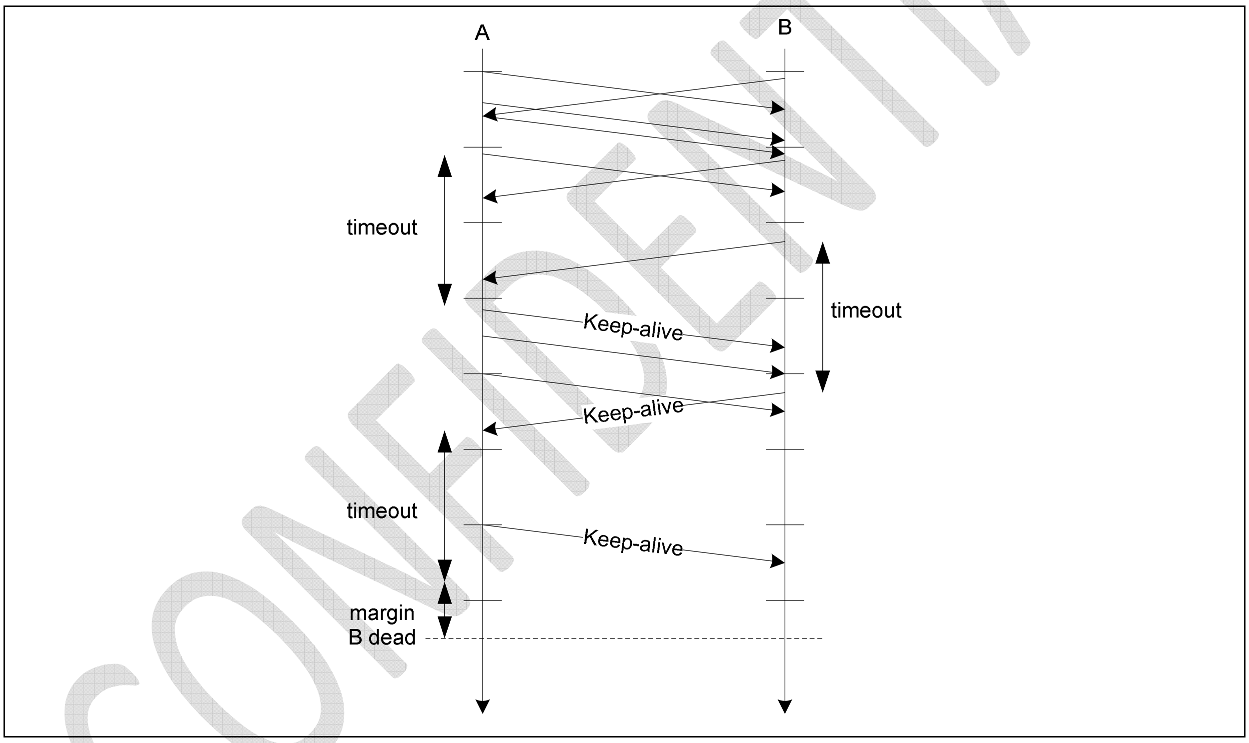

Figure 11: Ensuring connection validity by means of life sign telegrams. The unnamed directed lines indicate normal telegrams of random type, whereas the lines marked “keep-alive indicates telegrams of exactly that type.

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 not 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	20000	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

Acknowledgement Required:

No

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”