Lucas and WCS Communications Document

PR2732 – Southeast

Document Information

|  |  |
| --- | --- |
| File Name | PR2732\_Southeast\_Lucas\_WCS\_Communications\_Document\_v1.0 |
| Created | 03/23/2020 |
| Last Edited | 03/23/2020 |
| # of Pages |  |

Revision History

|  |  |  |
| --- | --- | --- |
| Ver# | Revision  Date | Revision Reason  (PR#,CR#,Defect#) |
| 1.0 | 03/23/2020 | Original Document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# *WCS File Interface*

The WCS will communicate with WM via Ethernet TCP/IP utilizing a shared directory on the WCS server to receive download from Lucas.

For the purposes of this document, the following definitions are important:

|  |  |  |
| --- | --- | --- |
| Term | Definition | Protocol |
| Download | The transfer of data from Lucas to the WCS | Shared directory hosted on the WCS server |

## General Clarifications

1. Only one transaction (message) per line within the file; each record in the file will end with a line feed.
2. To prevent the WCS system from picking up an incomplete file, it is very important for Lucas systems to transfer the file from their system to the WCS with a “.TMP” extension. When Lucas has detected the file transfer is complete, Lucas will rename the file (now on the WCS server) with a “.DAT” extension.



## 

## Cartonized Data Message

Lucas will place a file in a *TBD* network share directory on the WCS server containing multiple records for a single route. Each record will contain the picking requirements for a store/order/stop.

### *Download Filename*

The filename for each file has the following format:

**ASSIGNMENT-YYYYMMDDHHNNSSIII.DAT**

|  |  |  |
| --- | --- | --- |
| **ASSIGNMENT-** |  | Constant indicating this file will contain container assignments. |
| **YYYYMMDDHHNNSSIII** |  | Year, Month, Day, Hour, Minute, seconds and milli-seconds |
| **.DAT** |  | Fixed value for the file extension (renamed with this extension after transfer) |

The WCS will poll for any file with a “.DAT” extension in chronological download order.

### *Record Format*

During processing of a file, the WCS will perform the following checks.

1. Each record in the file will be terminated after the last field with a Line Feed (LF) with no preceding delimited.
2. Unique assignment IDs cannot cross multiple files. All cartons for a particular assignment ID must be in the same file.
3. Routes-Stops may span multiple files.
4. Though fixed length fields are utilized, pipe (“|”) delimiters between fields will be utilized to facilitate debug/troubleshooting and increase general raw data “readability” by human eyes.

The message data will be as follows. Note, all fields are fixed length. All data (numeric or alpha) that does not fill the max length will be left justified and space padded to the end of the field.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| Record Identifier | 8 | Alpha | Constant “CARTONIZ” for container |
| Route Number | 6 | Alpha | The Route for this container. A Route corresponds to a truck, which also corresponds to a dock door. |
| Stop Number | 4 | Alpha |  |
| Container ID | 15 | Alpha | The specific container for this pick. |
| Assignment ID | 25 | Alpha | The assignment for this container. An assignment ID is created by Jen and consists of a route+stop+area |
| Pick Area | 6 | Alpha | Concatenation of 3 digit stype and 3 digit pick area. |
| Pick Type | 10 | Alpha | For Full Cases, if available, a description of the product will be sent, if not available will be filled in with “Full Case”.  For Split Cases, this will always be filled in with “Split Case”. |
| Jurisdiction | 6 | Alpha | Needed for Cig Stamping (Cigarettes Only otherwise filled with 6 spaces) |
| # of Cartons | 2 | Alpha | Number of cig cartons in a container, needed for cig stamping (Cigarettes Only otherwise filled with 2 spaces) |

Example:

CARTONIZ|307604|1123|307604112-005 |307604112 |006003|SPLIT CASE|113205|20*<LF>*

\* NOTE, <LF> is not the text “<LF>” but the Line Feed character (0x0A)

# *TCP Socket Transfer*

The WCS supports standard (non-encrypted) ASCII TCP/IP Socket communications.

## Sockets Protocol

A common protocol is utilized for both the download and upload sockets. The general protocol is as follows:

1. Client transmits message (single transaction) to server.
2. Client waits for response from server (ACK).
3. When client receives the response from the server, the client may then proceed to the next transaction.
4. If no response is received within five (5) seconds, client closes the connection and attempts to re-connect to the server application.
5. Upon reconnection to the server application, the transaction which failed to receive a response will be resent.

Each server must recognize when a client has closed an open connection. In this case, the server must then begin listening on its listening socket to establish a new socket interface to the client.

Two (2) basic message formats are utilized for both download and upload communications. These message formats are listed as follows:

1. Data Message
2. Keep Alive Message
3. Message Acknowledgement Message

These message formats are defined/described below:

Note that all messages are comprised of ASCII formatted data only. Other than STX, ETX, ACK, and NAK, all characters should be ASCII printable data.

## Keep Alive Message

The Keep Alive message is utilized to inform the server that the client is still “up and running”. This message is sent every fifteen (15) seconds when no data is available for transmission. The format of this message is shown as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message Sequence Number | 5 | Numeric | Sequential one up counter for the message. Used by listening system to Acknowledge message receipt. |
| Msg ID | 8 | Alpha | Constant “KEEPALIV”. |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*12 |KEEPALIV*<ETX>*

The server will respond to the client with the following response:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message Sequence Number | 5 | Numeric | Sequential one up counter for the message. Used by listening system to Acknowledge message receipt. |
| Msg ID | 8 | Alpha | Constant “ACKNOWLE”. |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

The message sequence number in the Acknowledgement must match the message sequence number sent in the keep alive message.

Example:

*<STX>*12 |ACKNOWLE*<ETX>*

## Data Message

The Data message is utilized to send data transactions from the client to the server. This message is sent whenever data becomes available for transmission. The format of this message is shown as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message Sequence Number | 5 | Numeric | The rotating sequence number. The client generates this value (first message = 0, second message = 1, etc.). |
| Msg ID | 8 | Alpha | The 8 character message Identifier. |
| Data Record | Varies |  | Data record to be transmitted to the server. Data records are defined in the following section. |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*15 |<DATA>*<ETX>*

Detailed Examples provided in detail portion below.

The server will respond to the client with the following response:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message Sequence Number | 5 | Numeric | The rotating sequence number. The server should always respond with the same value that was transmitted by the client. |
| Msg ID | 8 | Alpha | Constant “ACKNOWLE”. |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

The message sequence number in the Acknowledgement must match the message sequence number sent in the data message.

Example:

*<STX>*15 |ACKNOWLE*<ETX>*

If the server fails to send a message acknowledgement message in response to a data message, the client should disconnect after a timeout period (5 seconds), reestablish the connection and send the message again. The client should NOT send the next queued message until the previous message has been acknowledged.

The server must acknowledge receipt of the message from the client. The server must NOT use the processing status determine if an acknowledgement is sent or not. ALL messages from the client must be acknowledge regardless of errors in the data content or processing disposition of the message on the server side. It is very important for this to be followed as the client will attempt to resend a message indefinitely if a message acknowledgment is not received from the client.

Data messages include the following message types:

|  |  |  |
| --- | --- | --- |
| **Message Type** | **Description** | **Sender** |
| ADDCONTA | New Container Message | Lucas |
| CONTCOMP | Container Complete Message | Lucas |
| ASGNCOMP | Assignment Complete Message | Lucas |
| ORDRCOMP | Order Complete Message | Lucas |
| ROUTCOMP | Route Complete Message | Lucas |

### *New Container Message*

Lucas will send a TCP socket message to the WCS. One record will be sent in each transmission framed with the <STX> and <ETX> characters.

#### *Trigger*

Lucas will send the WCS the New Container message after processing the SAP work file, has created assignments and cartonized the data.

#### *Record Format*

The message data will be as follows. Note, All fields are fixed length. All data (numeric and alpha) that does not fill the max length will be left justified and space padded to the end of the field.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message Sequence Number | 5 | Numeric | Sequential one up counter for the message. Used by receiving system to Acknowledge message receipt (See Message Acknowledgement section) |
| Msg ID | 8 | Alpha | Constant “ADDCONTA” |
| Route Number | 6 | Alpha | The Route for this container. |
| Stop Number | 4 | Alpha | The stop within the route for this container. |
| Container Id | 15 | Alpha | The unique container ID. |
| Assignment Id | 25 | Alpha | The assignment ID for this container. |
| Pick Area | 6 | Alpha | Concatenation of 3 digit stype and 3 digit pick area. |
| Pick Type | 10 | Alpha | For Full Cases, if available, a description of the product will be sent, if not available will be filled in with “Full Case”.  For Split Cases, this will always be filled in with “Split Case”. |
| Jurisdiction | 6 | Alpha | Needed for cig Stamping (Cigarettes Only otherwise filled with 6 spaces) |
| # of Cartons | 2 | Alpha | Number of cig cartons in a container, needed for cig stamping (Cigarettes Only otherwise filled with 2 spaces) |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*12 |ADDCONTA|307604|1123|SC307604112-006|307604112 |006003|SPLIT CASE|113205|20*<ETX>*

\* NOTE, <STX> is not the text “<STX>” but the start transmission character (0x02). <ETX> is not the text “<ETX>” but the end transmission character (0x03).

### *Container Complete Message*

Lucas will send a TCP socket message to the WCS. One record will be sent in each transmission framed with the <STX> and <ETX> characters.

#### *Trigger*

Lucas will send the WCS the Container Complete message once a picker has decided the container(s) can/will hold no more items and a new container must be created to hold the items remaining in the assignment.

#### *Record Format*

The message data will be as follows. Note, All fields are fixed length. All data (numeric and alpha) that does not fill the max length will be left justified and space padded to the end of the field.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message  Sequence Nbr | 5 | Numeric | Sequential one up counter for the message. Used by receiving system to Acknowledge message receipt (See Message Acknowledgement section) |
| Msg ID | 8 | Alpha | Constant “CONTCOMP” |
| Container Id | 15 | Alpha | The unique container ID. |
| Assignment Id | 25 | Alpha | The assignment ID for this container. |
| QC Flag | 1 | Numeric | 0 = false, 1 = true |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*14 |CONTCOMP|SC307604112-006|307604112 |1*<ETX>*

\* NOTE, <STX> is not the text “<STX>” but the start transmission character (0x02). <ETX> is not the text “<ETX>” but the end transmission character (0x03).

### *Assignment Complete*

Lucas will send a TCP socket message to the WCS. One record will be sent in each transmission framed with the <STX> and <ETX> characters.

#### *Trigger*

Lucas will send the WCS the Assignment Complete message when a picker has picked all the items for the assignment.

#### *Record Format*

The message data will be as follows. Note, All fields are fixed length. All data (numeric and alpha) that does not fill the max length will be left justified and space padded to the end of the field.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message  Sequence Number | 5 | Numeric | Sequential one up counter for the message. Used by receiving system to Acknowledge message receipt (See Message Acknowledgement section) |
| Msg ID | 8 | Alpha | Constant “ASGNCOMP” |
| Assignment Id | 25 | Alpha | The assignment ID that is complete. |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*574 |ASGNCOMP|307604112 *<ETX>*

\* NOTE, <STX> is not the text “<STX>” but the start transmission character (0x02). <ETX> is not the text “<ETX>” but the end transmission character (0x03).

### *Order Complete Message*

Lucas will send a TCP socket message to the WCS. One record will be sent in each transmission framed with the <STX> and <ETX> characters.

#### *Trigger*

Lucas will send the WCS the Order Complete message once the “Stop” has been picked.

#### *Record Format*

The message data will be as follows. Note, All fields are fixed length. All data (numeric and alpha) that does not fill the max length will be left justified and space padded to the end of the field.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message  Sequence Number | 5 | Numeric | Sequential one up counter for the message. Used by receiving system to Acknowledge message receipt (See Message Acknowledgement section) |
| Msg ID | 8 | Alpha | Constant “ORDRCOMP” |
| Route | 6 | Alpha | The Route for the order |
| Stop | 4 | Alpha | The Stop for the Route |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*3560 |ORDRCOMP|30760|4112*<ETX>*

\* NOTE, <STX> is not the text “<STX>” but the start transmission character (0x02). <ETX> is not the text “<ETX>” but the end transmission character (0x03).

### 

### *Route Complete Message*

Lucas will send a TCP socket message to the WCS. One record will be sent in each transmission framed with the <STX> and <ETX> characters.

#### *Trigger*

Lucas will send the WCS the Route Complete message once all stops have been picked for a route.

#### *Record Format*

The message data will be as follows. Note, All fields are fixed length. All data (numeric and alpha) that does not fill the max length will be left justified and space padded to the end of the field.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Max Length** | **Type** | **Note** |
| STX | 1 | ASCII | Constant binary 2 Start of transmission character |
| Message  Sequence Number | 5 | Numeric | Sequential one up counter for the message. Used by receiving system to Acknowledge message receipt (See Message Acknowledgement section) |
| Msg ID | 8 | Alpha | Constant “ROUTCOMP” |
| Route | 6 | Alpha | The Route to be completed |
| ETX |  | ASCII | Constant binary 3 End of transmission character |

Example:

*<STX>*3560 |ROUTCOMP|307601*<ETX>*

\* NOTE, <STX> is not the text “<STX>” but the start transmission character (0x02). <ETX> is not the text “<ETX>” but the end transmission character (0x03).