Communication Protocol

See also API reference for Credit Transfer Check API

The DuitNow system supports communication via Web (HTTPS) and TCP/IP where participants are able to establish connectivity to the system securely. This section provides information on the two protocols and explanation on how the connectivity flow works in the RPP DuitNow system.

HTTP

HTTP is short for "Hypertext Transfer Protocol" and it is a request-response protocol that allows users to communicate data over web servers. HTTP works similar to other application services like SMTP and FTP. Like the performance of FTP, it transfers a document using service of TCP port. But it uses just one TCP connection (usually at Port 80) i.e. at data link and no individual Control Connection is used.

HTTP is a protocol which fetches resources such as HTML documents. It is used for exchanging data on the Web and is a client-server protocol which means requests are initiated by the recipient usually the Web browser.

The controls from the client-side delivered in a request message into the webserver. The web server sends the requested content at a response message. The HTTP doesn’t provide any security and makes use of SSL (Secure Socket layer) to club security in its communication.

How HTTP Works

HTTP provides users a way by distributing hypertext messages between servers and clients to interact. HTTP clients generally use Transmission Control Protocol (TCP) connection to communicate with servers.

The HTTP can carry out various tasks to be performed by Request methods such as but not limited to:

- **GET**: Requests a specific source in its entirety.
- **HEAD**: A specific resource with no body content.
- **POST**: Adds articles, messages and information to another page under an existing web resource.
- **PUT**: Directly modifies a current web source and creates a new URL if need be.
- **DELETE**: Eliminates a specified source.

Additional resources regarding HTTP:

1. **The Flow of HTTP Request**
2. **HTTP Request Methods**

TCP/IP

TCP (Transmission Control Protocol) provides reliable, ordered, and error-checked delivery of a stream of bytes between applications running on hosts. It states that a connection is established and maintained until the application data at each end have finished exchange. TCP breaks application data into packets. This packet delivers to the transport layer. Layer 4 manages flow control and provide error free data transmission and handles retransmission of dropped or garbled packets and acknowledges all packets that arrive.

IP (Internet Protocol) is the protocol for shifting packets around between hosts in the network layer.

NOTE
Prior to using this, participant is required to engage with PayNet on arrangement for the TCP/IP client connector to be installed at client site.

How TCP Works

TCP's job is to ensure that all data sent in a stream moves from client to server in a correct order and is intact. TCP uses a technique known as positive acknowledgement with retransmission, requiring the receiving end of a transmission to give a response as to what data has been received. The bytes sent can exactly match the bytes received. No data is altered or lost along the way.

Connection is established and a 3-way handshake is made. First, the source sends a SYN request packet to the server in order to start session establishment process. Then, the server sends a SYN-ACK packet to agree to the process. Lastly, the source sends an ACK packet to the target to confirm the process, after which the data can be sent.

Additional resources regarding TCP/IP:

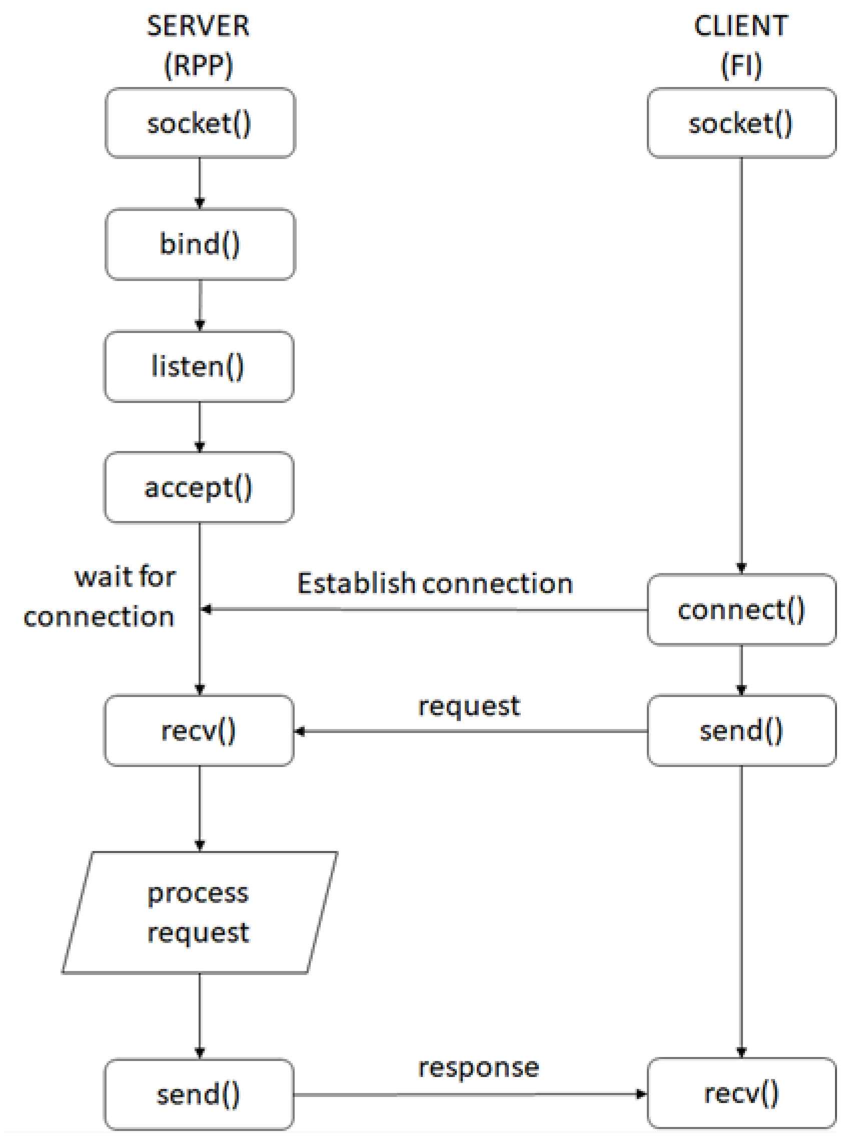
TCP Windowing

The TCP/IP Model

TCP/IP Client Server Model

Connection will be created at server side which is listening to specified port for client to establish the connection. FI will be a client side to establish the connection in order to send and receive message to/from RPP hub.

The following is the connectivity diagram for both client and server:



Messaging Flow

Connection Type

FI is required to create socket and establish a persistent connection (keep alive) to RPP Hub on the specified port given. Every message must be sent and received via the existing socket established before and it is not recommended to establish single-use connection every time sending the request.

Message Length

Messaging to RPP Hub should be started with sending the message length in the first four bytes, followed by the real message. This is required for every message format sent to RPP respectively.

This messaging format is not only limited to the request from client, but the same format will also be used by any message coming from Hub itself. Client side must be designed to handle this kind of message format in order to process messages from Hub.

Sending and Receiving Message

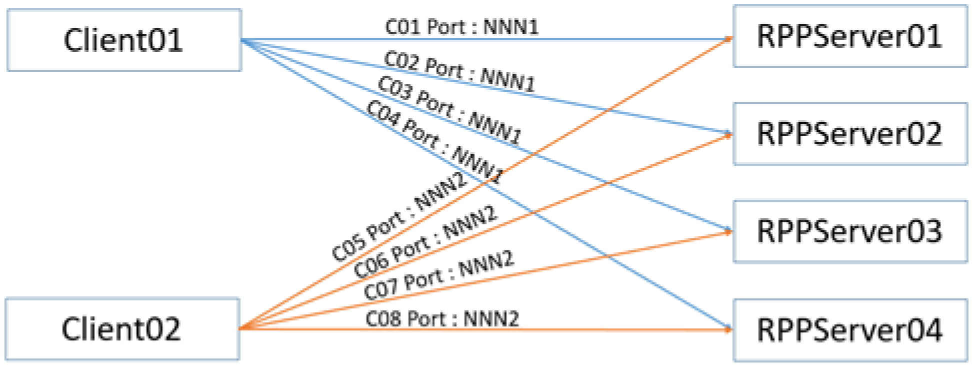
To send and receive message to/from hub, client should be running synchronize write and read to the existing connection stream.

Connection Handling

Connection Structure

Every FI will be assigned n number of ports (currently n=4). An FI can use up to 4 clients (1 port = 1 client) to connect to the RPP server. RPP Hub has 4 servers and every client must connect to every server and send messages on a round-robin basis to all 4 servers subject to availability of the servers.

The diagram below shows the sample connections for each PayNet server:



For FI that has 4 clients, the total number of connections will be 16. It is up to the FI to decide how many clients they would like to have depending on their requirements such as primary site and DR site.

Persistent Connection

As described in Connection Type, clients must be using the same keep-alive connection and should not close the connection every time they send a message to the Hub.

Connection Drop

Clients must be configured with an ability to re-connect if the connection is dropped.

Request Messages

All request messages must be sent on a round-robin basis to all 4 RPP servers from each client. E.g. When the first message goes via Port C01, the second message should go via C02, therefore the third and fourth will go via C03 and C04 respectively. The fifth message will be sent out via C01 again and so on.

Response Messages

Response messages from both RPP server and client must follow the same route/connection where the request/response was sent. E.g. if a request message from RPP server was sent via

Port C01, client must send the response to that request via C01 only. If the connection through C01 is somehow broken or lost, all pending messages can only be further continued when C01 is active again or otherwise those messages will be considered as lost. To handle timeouts for lost messages, please refer to latest FSD.

Message Timeout - Expired

Client should maintain a table of all requests sent to RPP server and received from RPP server. All requests must be dropped upon a configurable timeout. No responses must be sent or received if corresponding request is not found.

Incoming Request Messages:

Incoming Request Message	Channel	Request Timestamp	Request Drop Interval
Incoming message 01	C01	20190101235900	+10,000ms
Incoming message 02	C02	20190101235901	+10,000ms
Incoming message 03	C03	20190101235902	+10,000ms
Incoming message 04	C04	20190101235903	+10,000ms
Incoming message 05	C04	20190101235904	+10,000ms
Incoming message 06	C03	20190101235905	+10,000ms
Incoming message 07	C02	20190101235906	+10,000ms
Incoming message 08	C01	20190101235907	+10,000ms

Outgoing Messages:

Outgoing Message	Channel	Request Timestamp	Request Drop Interval
Outgoing message 01	C01	20190101235900	+10,000ms
Outgoing message 02	C02	20190101235901	+10,000ms
Outgoing message 03	C03	20190101235902	+10,000ms
Outgoing message 04	C04	20190101235903	+10,000ms
Outgoing message 05	C05	20190101235904	+10,000ms
Outgoing message 06	C06	20190101235905	+10,000ms
Outgoing message 07	C07	20190101235906	+10,000ms
Outgoing message 08	C08	20190101235907	+10,000ms

Digital Signature

All messages shall be signed using an Asymmetric (public key) Cryptography Mechanism. This involves calculating of RSA signatures on outbound messages and verifying RSA signatures on inbound messages. Signature and Public Key information of certificate used for signing must be placed in XML header (AppHdr) in the 'Sgntr' block. The signature digest is the encapsulated signature transformed in the general XML context. This would require **PKI Management.**

As RPP messages are in XML format, RPP require signature adhering to XML Signature which signatures are generated from a hash over the canonical form of a signature manifest. Requirements for the signature are:

- **Syntax**: XML Signature
- **Type**: Enveloped
- **Canonicalization Method**: Exclusive XML Canonicalization
- **Signature Type**: RSA
- **Hash Algorithm**: SHA256

NOTE

PayNet has 2 appointed vendors for procuring certificates. FI and PayNet must exchange Public Key (Certificates) beforehand.