STARNET COMMUNICATIONS

FAST X

Protocol Extensions With ClientComm

Background

FastX 3.1 implements the ClientComm channel which is a general purpose protocol channel that can be used to send small messages between the FastX session server and a FastX client (desktop or browser) that are not part of the core remote desktop (which includes screen updates, keyboard/mouse operations, and copy/paste). This channel allows messages to be sent from custom user-defined scripts on the server to the client. Depending on the message type, the client can "break out" of the core protocol and execute applications on the client OS.

Examples of ClientComm use include:

- Alerts (pop-up messages) on the client window, outside of the X11 desktop.
- Open a browser window on the client (useful when the session server does not have Internet access).
- Support of session sharing.
- File transfer.
- Any application where a user wants to click on a button in the linux session and it have some effect on the client.

Example Use Case

Many FastX users run their Session Servers on a closed network. FastX clients connect to the session server via a VPN or external gateway. The FastX client machine has internet access. The Session server does not.

This setup poses a problem when trying to access online help documentation. Clicking on the help button will bring up a browser on the Linux machine that will always fail since the Linux machine is not connected to the internet. The clientcomm channel solves this problem.

Say the documentation is located at *https://www.my-online-help.com*
An administrator can set up a script on the linux machine that runs a command:

/usr/lib/fastx/3/tools/clientcomm --type=fx-url https://www.my-online-help.com

The end user can click on a button inside the FastX session that runs the above command. This message will be sent to the client. The clientcomm channel will decode the message and then bring up a browser on the client side going to that URL.

Using the ClientComm Channel

In a default installation the clientcomm access script is found in */usr/lib/fastx/3/tools/clientcomm*

clientcomm is a command line script

Command Line Arguments

--id=<fastx session ID>

The default session ID is the environment variable FASTX_SESSION_ID

--config_path=<path>

Used if the FastX config directory is in a non-standard location. This can also use the FX_CONFIG_DIR environment variable.

--type=<ClientComm type>

All ClientComm messages use a type name. This defaults to "fx-notify". FastX client versions 3.1.21 and later implement "fx-notify" and "fx-url".

--list

Specifying this makes this tool list all the FastX clients currently connected to this session; instead of sending a ClientComm message.

--clients=<client1,client2,...>

Specify which client(s) are the recipient of this ClientComm message. This can be a client ID as shown by running this tool with the "--list" option, or a client role shown below. When this option includes a comma (e.g. "master,controller") then this message is sent as a notification (with no response possible). If this option is not specified, then the message will be sent as a notification to all clients. If this option does not have a comma, then the message is directed to that client only, and a response is expected. It is not allowed to specify "viewer" or "inactive" without a comma.

Client Roles

"master"

The client that "owns" this session. Currently, this is the only role.

"controller"

When sharing a session, the master may pass keyboard/mouse control to another client, which will take the "controller" role.

"viewer"

This client can see the session display, but can not send keyboard/mouse input.

"inactive"

When a guest client first connects, it is assigned this role, which cannot see or interact with the session. It can still receive ClientComm sessages, however.

"master-viewer"

When the master has passed keyboard/mouse control to a different client, it takes this role.

<message>

Arguments after the option are sent as the data part of the message.

If the "--clients" option is used without a comma, the message is sent directed, and the client will send a response. For example, a script could call clientcomm like this:
response=$(/usr/lib/fastx/3/tools/clientcomm --clients=master --type=fx-notify '{"message":"Save the file?", "buttons":["Yes","No","Abort"]}')
The current desktop client only works with some buttons including ok, open, save, cancel, yes, no, abort and a few others. Future versions function with anything.

ClientComm Default Types

The clientcomm channel uses a string as the “type” name, which is used to route the various messages. This document describes the protocol for the various types.

FastX ships with predefined message types. FastX clientcomm types are have the fx- prefix

fx-notify

Fx-notify is used to send a short message to be shown to the user.

Data field

The data field **may** be a JSON-encoded object.

If it is not a JSON-encoded object, then the client will assume it is a plain UTF-8 encoded text string and display it in the default manner.

If it is a JSON-encoded object, then it must have a “message” member. (If this member is missing or blank, the behavior is undefined, however in this case the desktop client will display the entire data field, which will be helpful for debugging, although a very poor user interface.)

The meaning of the members of this object are as follows:

message (string): The message to be displayed, in UTF-8 encoding.

title (string): The title of the window (popup) used to display the message, if a window is used.

timeout (number): Number of milliseconds before the message disappears

icon (string): The type of icon to use for the window. It may be one of:

- none (No icon)
- information
- question
- warning
- critical (not recommended)

buttons (array of strings): If this is present, the buttons listed will be shown to the user. The button he presses will be returned in a message that contains an object with a string “button”. The client may treat the button names as case-insensitive. Possible buttons are:

“Ok”, “Open”, “Save”, “Cancel”, “Close”, “Discard”, “Apply”, “Reset”, “RestoreDefaults”, “Help”, “SaveAll”, “Yes”, “YesToAll”, “No”, “NoToAll”, “Abort”, “Retry”, “Ignore”. Obviously, many of these are not appropriate for a FastX session. Including a string that is not in this list will have undefined behavior.

Message number

The “fx-notify” message may be sent as a notification (with no response), or as a query. To indicate the latter, the ClientComm “number” field will be non-zero, and the “*buttons*” field described above should be used.

Example

```
/usr/lib/fastx/3/tools/clientcomm --id=$FASTX_SESSION_ID --type=fx-notify
\{ \"title\": \"MY NOTIFICATION\", \"message\": \"hello there\",
\"icon\":\"information\", \"timeout\": 60000, \"buttons\": [ \"close\"] \}
```

fx-url

Fx-url will give a url that will be opened by the client browser.

Data field

The data field is EITHER a STRING or a JSON encoded object

String

If the data field is a STRING, then the string is the URL to open

JSON

The JSON encoded object has the following properties

- url -- the URL to open

Example

```
/usr/lib/fastx/3/tools/clientcomm --id=$FASTX_SESSION_ID --type=fx-url
http://www.starnet.com
```

fx-share

Used for session sharing

Data field

The data will always be a JSON encoded object. If the data cannot be decoded as such, then a NACK must be sent (if NUMBER is non-zero), and the “name” field in the “error” object will be “format”.

The JSON objects sent will be similar to JSON-RPC, except that “jsonrpc” and “id” are not included (because the NUMBER field is used as the message id). Clients and server will use “method”:string for requests and notifications

ClientComm Custom Types

Administrators can create custom clientcomm types. The administrator must define both the client and server messages, as well as create a way to process the message once it reaches the client.

Web Client

The web client supports custom clientcomm types. Any clientcomm type that the FastX Web Client does not understand will be sent as a message to the parent window. Administrators should wrap the FastX Web Client in an iframe and listen for the window.onmessage event