INTEG Websocket Interface Specification

1. Abstract

The JANOS Web Server listens for connections from clients that are running one of the many popular browser programs. Typically ports 80 and 443 (for secure TLS/SSL communications) are open for connection although those are configurable through the JNIOR Registry. In addition to the default HTTP Protocol a connection may also utilize the Websocket Protocol as described in this document. The Web Server ports are shared by these two protocols. This provides for access to status information and control commands that previously were only available through the JNIOR Protocol. While the JNIOR Protocol remains a viable option for these functions the Websocket approach offers seamless integration into the dynamic web page environment. This capability is new to the Series 4 JNIOR products (Models 410, 412, and 414).

2. Introduction

2.1 Background

In order to remotely control the JNIOR you need the ability to obtain I/O status and to affect changes in I/O condition. In the earlier Series 3 JNIOR this was accomplished through the JNIOR Protocol[1] made available through a TCP/IP connection typically on port 9200. This is a documented binary protocol that requires special programming external to the JNIOR for its use. Care is also required to allow access to the specific port through routers and firewalls. Once successfully implemented the JNIOR Protocol not only provided I/O status and control mechanisms. It also opened access to the JNIOR Registry[2] and thereby the ability to configure and manage the product.

In addition to the JNIOR Protocol it was also necessary to access the JNIOR Command Line through Telnet. Care again is required to allow access to the Telnet port (Port 23) through routers and firewalls. The Command Line is also accessible using a serial connection to the RS-232 port on the JNIOR. This *Console* connection provides tools for monitoring I/O status and affecting I/O conditions as well as use of various kinds of diagnostics. Furthermore in this environment the product can be fully configured in all aspects including the network parameters. In addition this is where application programs can be executed which extend the functionality of the JNIOR product.

Management of the JNIOR also requires the manipulation of files in the local file system. While files may be manipulated through Console connection transfer to/from an external system is done using FTP. Again care must be taken to allow access to the FTP command port (Port 21) through routers and firewalls. FTP typically opens/accepts data connections which must also be accommodated by the network.

With the introduction of the Series 4 JNIOR running the JANOS operating system the various I/O and management requirements covered by these other protocols can be additionally handled through a single Web Server connection. Access to the Web Server is typically through ports 80 and 443. The latter connection providing for TLS/SSL up to 256-bit security. While these ports would also need to be accommodated by routers and firewalls this is a much more standard requirement and often routine request for IT personnel. This consolidation of functionality is accomplished using the Websocket Protocol[3] as specified by the Internet Engineering Task Force (IETF) in combination with JANOS server-side scripting. This can result in a fully functional browser-based dynamic website providing JNIOR monitoring and control. The example being the configuration pages provided with the product. These Javascript[TM] based dynamic web pages have replaced the Java-based applets used by the Series 3 JNIOR products.

2.2 Protocol Overview

Most computer languages today accommodate programmatic connection to Web Servers in one fashion or another. It makes sense since the majority of applications developed today involve networking and therefore access to the vast range of data available through Internet. These web application needed some form of bi-directional communication between the client and server. For a time programmers attempted to get the job done through an abuse of the HTTP Protocol. A simpler solution has been provided in the form or a Websocket API[4] which has been quickly accommodated. As a result most web-based programming environments support Websocket connections and the programmer can utilize them as easy as any other web protocol.

Briefly, the client makes a connection to a JANOS Web Server port. This port expects a valid HTTP connection but is also shared by the Websocket protocol. Transparently behind the scenes the connection issues the appropriate HTTP headers requesting an 'upgrade' to the Websocket protocol. When the handshake is complete the connection will be ready to handle bi-directional websocket messaging. The JANOS Web Server supports a built-in websocket service with messaging that can be used to monitor, control and manage the Series 4 JNIOR. The built-in service employ JSON[5] message formatting.

To provide additional flexibility the JANOS Websocket connection can, through a parameter in the URL, be redirected to an application running on the JNIOR. In the case websocket messages are routed through the JANOS inter-process messaging mechanism to the application program. The program uses the same messaging system to provide replies and messages outward through the websocket connection. In this fashion a completely custom messaging system can be implemented.

2.3 Security

Any protocol providing control and management functions must employ some form of security preventing unauthorized access and disturbance. In addition to being available through a TLS/SSL secure connection the built-in JANOS Websocket implementation requires authentication. The authentication handshake must be successfully completed before any operations for monitoring, control and management will be allowed. This login uses active JANOS user accounts and subsequent operations adhere to the account permissions assigned by the administrator.

To facility the seamless use of the Websocket protocol in the implementation of dynamic web pages a mechanism is provided that utilizes any website authentication completed by the browser to pre-authorize the websocket connection. This insures that only a single entry of login credentials is required to bring up a fully functional and secure dynamic website served by the JNIOR. Note that custom application running on the JNIOR that serve websocket connections are free to implement or ignore any kind of authentication requirement.

3. Connection

3.1 Built-in Websocket Service

Once a connection to a configured Web Server port (default 80 and 443) is made and upgraded from the default HTTP Protocol to the Websocket Protocol, traffic must conform to the websocket specification. Except in the case discussed in Section 3.2 the JANOS Web Server uses a built-in server to handle all websocket messages then received. These must use the JSON format and the connection must be authenticated.

To initialize communications the client should send a blank or empty message. The following is acceptable.

```
{
  "Message":""
}
```

The connection will proceed depending on the authentication requirements established by JNIOR configuration and the environment making the connection (browser, application, etc.).

3.1.1 Default Permissions

The Series 3 JNIOR (Models 310, 312, and 314) were shipped where by default web pages were not protected by login. The login requirement encountered when running the applets was the result of security in the JNIOR Protocol. Access to web pages could be controlled by permissions set on individual files or folders. For instance, removing the read attribute (R) from the /flash/www folder would force the browser to ask for login credentials and thus protect the pages. Unfortunately a second login would then be required by the JNIOR Protocol which has to be separately protected since control and configuration was possible though it. Modifying folder permissions involved a console command (CHMOD) which tended to be unfamiliar to everyone.

The Series 4 JNIOR (Models 410, 412, and 414) use a new Registry key WebServer/Login to control web page access. This key by default is set to TRUE. The dynamic configuration pages therefore also request login credentials but a second login is not required due to the Websocket Protocol implementation which will be discussed in a moment. The Websocket Protocol replaces the JNIOR Protocol just as the dynamic configuration pages replace the applets which have been dropped. You may still control access to areas of the JANOS website using file and folder permissions if desired. That would only be necessary should you disable the WebServer/Login requirement.

3.1.2 WebServer Login Enabled

With the Web Server Login requirement enabled any access to the JANOS website is challenged using the standard 401 Unauthorized response. The JANOS Web Server provides the necessary parameters so the browser can request the user's login credentials. If the proper credentials are entered and verified by JANOS the page is promptly served. A session ID is assigned.

Subsequently the Authorization information is supplied with requests for other pages required from the website. The JANOS Web Server recognizes the association between those credentials and the original login and therefore doesn't challenge each and every page. When the browser then moves to open a Websocket connection it uses the temporary session ID for authentication. A second is not required. This can be done because all of these connections are handled by the Web Server. This unlike the JNIOR Protocol which is a separate server entirely and cannot be passed shared authentication information.

Once you have authenticated for the website, you can create Websocket connections in the browser session without an additional login step. Immediately after opening the Websocket connection you will receive a "Monitor" message and you are good to go.

3.1.3 WebServer Login Disabled

If you set the WebServer/Login key to FALSE and assuming that permissions on files and folders have not been modified retaining the default Read Access flag, the browser will not need to request your login credentials. When a Websocket is then made there are no preauthorized credentials. The login handshake in the Websocket connection will be required before you may proceed using the websocket. This behavior assumes that the Websocket/Anonymous Registry key has not been defined or is set to 0.

Upon opening the connection a "monitor" message will not be provided. The application needs to send the blank message and will receive basically the 401 Unauthorized error. The application will then need to request the user's credentials and calculate the Auth-Digest response on its own. This is the same procedure performed by the browser. The dynamic configuration pages supplied with the JNIOR provide for this requirement. The Javascript can be used for reference.

Once the user credentials are processed the handshake can be completed and will proceed as follows.

```
{
  "Message":""
}
```

```
{
  "Message":"Error",
  "Text":"401 Unauthorized",
  "Nonce":"5d894efb48e1c3bc074fe78e7a5f"
}
```

```
{
  "Auth-Digest":"jnior:65f2d1cb66ef63f7d17a764f3a2f2508"
}
```

```
{
  "Message":"Authenticated",
  "Administrator":true,
  "Control":true
}
```

A "Monitor" message will likely immediately follow. This might even be received before the "Authenticated" message. That is the asynchronous nature of the connection. Please feel free to contact INTEG for assistance in implementing the digest calculation.

3.1.4 Anonymous Operation

If the Websocket/Anonymous Registry key is set to a valid user ID (1 is generally the JNIOR Administrator's ID), no login will be required. The USERS console command can be used to determine the available user IDs. This, however, is extremely dangerous. Any application can then make a fully functional Websocket connection. This gives anyone access to the unit with the ability to raise havoc with controls and to modify JNIOR configuration. This is not recommended. If there is physical security, meaning that access is only available to personnel on the local network and all those can be trusted, then this setting may be of use. Otherwise you are allowing anonymous access to this connection at your own risk.

It is important to note that even if you have WebServer/Login set to TRUE and have to enter username and password to bring up the website, the websocket interface is <u>not secure</u> if anonymous access is enabled. A separate application or copy of the website can get full control of your JNIOR.

With the anonymous key set to a valid user ID, the websocket connection will not require login. Immediately after making the connection you will receive the "Monitor" message. The connection is then available for full use.

3.2 Custom Server Applications

The JANOS operating system supports an inter-process communications mechanism similar to Windows(R). A Java application can be written to run in the JANOS environment (built solely against etc/JanosClasses.jar) which can exchange messages with the Web Server process. A Websocket connection can be opened and redirected to communicate with the application which can then act as a custom server. This is achieved by URL parameter. For instance the following URL will establish such a connection.

```
ws://10.0.0.201&app=1030
```

The inter-process messaging system uses a numeric ID to identify messages. JANOS system messages all use an ID less than or equal to 1023. This leaves IDs of 1024 and greater free for user assignment. The **APP** parameter in this URL defines a message ID to be used for all incoming websocket messages on this connection. The presence of the **APP** parameter tells the JANOS Web Server not to utilize the built-in websocket server for this connection but to pass all messages received onto the inter-process message system with the defined ID.

The application server is written to open a message listener and to collect all messages with (in this case) an ID of 1030. Note that an even number should be used for all incoming messages. These messages may be formatted as desired by the application. There is no requirement and JSON is may not even be used. The application server can then formulate a response or any other transmission placing the content onto the inter-message system with an ID one greater than assigned by **APP**. In this case that would be 1031. The Web Server processes user messages forwarding them back out through any websocket connection assigned to the ID one less (or 1030).

It is important to note that more than one websocket connection can be made assigning the same **APP** number. A single custom server will receive all such messages. It is provided additional information pertaining to the socket so it may handle the separate sources and respond specifically to each connection using the common reply ID (APP + 1).

If the custom server application is not running on the JNIOR the websocket will not receive a response to its messages. It may keep trying. There presently is no mechanism to automatically start the custom server upon detection of a websocket connection. The custom server application should be started at boot using an appropriate Run key.

Please feel free to contact INTEG for assistance if you are interested in writing a custom server of this kind.

4. Messaging

This section describes the various bi-directional messages and replies supported by the built-in Websocket Server. These communications are handled by the JANOS Web Server process.

The JANOS server implementation is not Master-Slave however there are a number of 'Requests' that have 'Responses' which is typical for such a server. In addition to this, unsolicited messages may be received from the server. These provide immediate notification for changes in I/O status and updates in configuration settings for instance. Any use of this websocket implementation must handle the presence of unsolicited messages. Care is also required to pair responses with the associated requests as messaging order is not guaranteed. Optional Meta data supplied with a Request is returned with the Response unmodified. This can then be used to identify each response and the action it then requires.

This Websockets interface utilizes the same set of JSON Object as does the JANOS Management Protocol (JMP). Please refer the JANOS Management Protocol Specification for further detail. (See JANOS Management Protocol (JMP)). JMP messages are transmitted in a 2-element JSON Array format. This Websockets interface transfers only the JSON Object associated with the message. The blocking here is handled by the low level Websockets logic.

References

[1] JNIOR Protocol Specification, INTEG Process Group, Inc.

[2] Registry Key Assignments, INTEG Process Group, Inc.

[3] RFC 6455, Internet Engineering Task Force Standard Track, ISSN: 2070-1721, Dated December 2011. https://tools.ietf.org/html/rfc6455#ref-WSAPI

[4] Hickson, I., "The WebSocket API", W3C Working Draft WD-websockets-20110929, September 2011, http://www.w3.org/TR/2011/WD-websockets-20110929/. Latest version available at http://www.w3.org/TR/websockets/.

[5] The JSON Data Interchange Standard, http://www.json.org/

[6] Wireshark Network Protocol Analyzer, https://www.wireshark.org/