Xpress Flex API Integration

Introduction

This document details commands and methods for integrating a third party application with **Xpress Flex Connect** using a simple REST API.

**Xpress Flex Connect** is Taylor Technologies Inc.'s bridge application, allowing communication between **Xpress Flex**™ devices and water testing software.

**System Requirements**

* Windows 10
* 10 Mb available disk space (50 Mb if using sureTREATXpress)
* High Speed USB 2.0 Port for device connection
* 4 Gb RAM

**Application Requirements**

* Any application capable of http communication with localhost, including a browser-based application (HTML+JS+CSS). Taylor Technologies provides [xfConnect.js](https://w01.href.com/wiki/Xpress_Flex_Series_2_API#xfConnect.js) which fully implements the required REST API.

**Test Strip Names and Strip Codes**

When strips are manufactured by Taylor Technologies, a code is printed on the bottom of each bottle. The last 3 characters are the Strip Code for the strips inside the bottle. This enables the software to use an optimized calibration curve for each Strip Code for improved accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Strip** | **Analytes** | **Strip Codes** | **Qty in Bottle** |
| Flex1 | Fe | A01 .. A99 | 50 |
| Flex2 | S1, S2, S3, Bo and Cu | B01 .. B99 | 50 |
| Flex3 | FC, TC, pH, TA, TH, CYA | C01 .. C99 | 100 |
| Flex4 | PO4 | D01 .. D99 | 50 |

**Third Party Developer Assistance (planned)**

Taylor Technologies plans to provide a full simulation version of **Xpress Flex Connect** such that a third party application can be developed without the need for a physical device. Ideally this will simulate all error conditions plus all variations of the status response for complete testing

API

A third party application runs tests using an **Xpress Flex**™ device by sending HTTP GET requests to **Xpress Flex Connect**. Responses are returned as JSON, detailed below.

**Xpress Flex Connect** implements a Windows-based HTTP server using localhost bound to a unique port. The port has a default value of 61264 and can be overridden.

The API uses six commands which share a common base url.

* Base URL = **http://localhost:port/xf2/v1/{command}**

Responses occur over http (not https) and only on localhost.

The port used by XFConnect.exe can be modified in the XFConnectPort.txt file, which is created in the %ProgramData%\TaylorTechnologies\XFConnect folder when XFConnect.exe first runs. The javascript needs to use the same port as XFConnect.exe on any given Windows computer.

Client Animation Timing

If the third party water testing software is to provide animation for use with the **Xpress Flex**™ device, then the following key timing points need to be coded for :

* Start
* Prepare to Dip Strip, 3 sec
* Immerse Strip, 2 sec
* Insert Shuttle, 11 sec
* Analyzing, 60 sec ( or sooner if results are returned with the Status request )

Importantly, the **/newtest** request is sent at the start of the Prepare-to-Dip countdown.

Command Summary

* **about**: determines basic details about the hardware device, relevant software, strip codes and inventory
* **initialize**: cancels any tests in progress without raising an error, and returns all system details (same output as provided by the /about command)
* **inventory**: lists Strip Codes with expiry details, indicates which open bottle Strip Codes are active and optionally adjusts the number of test strips remaining in each bottle
* **newtest**: initiates a new water test with optional animation
* **status**: checks progress after a test has started. At the end of a test, the returned JSON object will contain the water test results.
* **cancel**: interrupts and stops a currently active test
* **update**: updates software EXE and/or data file based on recommended versions

Command Sequencing

The Xpress Flex device consists of a programmable board that controls a digital camera. The XFConnect software communicates with this programmable board over usb. Even though commands to the XpressFlex device have been implemented as HTTP GET requests, they cannot behave exactly as true HTTP GET requests over the web. Commands must be issued in the proper sequence with enough time between commands to allow for their proper interpretation. Avoid sending multiple requests to the device in an asynchronous manner where things could be called out of order.

Best practice for working with the XpressFlex device is:

1. Make the GET request
2. Receive the Response
3. Make another request …

The process for running a Water test is

1. Initialize -> Response
2. Newtest -> Response
3. Call Status repeatedly unlit results are present.
4. Cancel or Initialize will halt in progress tests

For best results, follow the paradigm used in the xfConnect.js library (see function xfNewWaterTest).

Command Detail

(note, response

*about*

The JSON details of the **about** command are shared with the **initialize** and **update** commands.

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**about** |

**Response**

{  
   "requestCommand":"about",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
   "details":{  
    "softwareVersion":"2.1.0.0",  
    "apiVersion":"1.0",  
    "dataVersion":"1.0",  
    "documentation":"[https://xpressflexseries2api.taylortechnologies.com](https://xpressflexseries2api.taylortechnologies.com/)",  
    "deviceName":"XpressFlex",  
    "deviceSeries":2,  
    "deviceSerialNo":"XF000001",  
    "deviceFirmware": "0.14",  
    "animationAvailable":true ❘ false,  
    "newDataAvailable":true ❘ false,  
    "newDataVersion":"1.1",  
    "newExeAvailable":true ❘ false,  
    "newExeVersion":"2.1.1.5",  
    "newFirmwareAvailable":true ❘ false,  
    "newFirmwareVersion":"0.15",  
    "exeName":"XPressFlexConnect.exe ❘ XPressFlexConnectService.exe ❘ sureTREATXpress.exe",  
   },  
   "flexStrips": {  
    "stripCodes": {  
     "Flex1" : [  
      {"code": "A01", "status", "Optimal ❘ PastBestByDate ❘ Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A03", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex2" : [  
      {"code": "B01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
     ],  
     "Flex3" : [  
      {"code": "C01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "C02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex4" : [  
      {"code": "D02", "status", "Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D03", "status", "PastBestByDate", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D04", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ]  
    },  
    "openBottleStripInventory": {  
     "Flex1" : { "code": "A01", "count": 49 },  
     "Flex2" : { "code": "B01", "count": 49 },  
     "Flex3" : { "code": "C02", "count": 99 },  
     "Flex4" : { "code": "D01", "count": 48 }  
    }  
  }

**Response Notes**

* When the device is disconnected, many values such as deviceSerialNo will be blank and should be ignored.
* For **StripCodes**, a maximimum of 3 will be valid for each strip.
* Within **openBottleStripInventory**, the **count** field provides the number of strips remaining in that bottle.

*initialize*

Calling **initialize** will cancel any test in process. The JSON response details of the **initialize** command are shared with the **about** command.

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**initialize** |

**Response**

{  
   "requestCommand":"initialize",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
   "details":{  
    "softwareVersion":"2.1.0.0",  
    "apiVersion":"1.0",  
    "dataVersion":"1.0",  
    "documentation":"[https://xpressflexseries2api.taylortechnologies.com](https://xpressflexseries2api.taylortechnologies.com/)",  
    "deviceName":"XpressFlex",  
    "deviceSeries":2,  
    "deviceSerialNo":"XF000001",  
    "deviceFirmware": "0.14",  
    "animationAvailable":true ❘ false,  
    "newDataAvailable":true ❘ false,  
    "newDataVersion":"1.1",  
    "newExeAvailable":true ❘ false,  
    "newExeVersion":"2.1.1.5",  
    "newFirmwareAvailable":true ❘ false,  
    "newFirmwareVersion":"0.15",  
    "exeName":"XPressFlexConnect.exe ❘ XPressFlexConnectService.exe ❘ sureTREATXpress.exe",  
   },  
   "flexStrips": {  
    "stripCodes": {  
     "Flex1" : [  
      {"code": "A01", "status", "Optimal ❘ PastBestByDate ❘ Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A03", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex2" : [  
      {"code": "B01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
     ],  
     "Flex3" : [  
      {"code": "C01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "C02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex4" : [  
      {"code": "D02", "status", "Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D03", "status", "PastBestByDate", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D04", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ]  
    },  
    "openBottleStripInventory": {  
     "Flex1" : { "code": "A01", "count": 49 },  
     "Flex2" : { "code": "B01", "count": 49 },  
     "Flex3" : { "code": "C02", "count": 99 },  
     "Flex4" : { "code": "D01", "count": 48 }  
    }  
  }

*inventory*

Call the inventory command to determine Strip Code information and Open Bottle Strip Inventory, or when a new bottle is started and/or when you wish to adjust the inventory because a test strip has been wasted.

There is a **grace period** of 2 months from the Best-By Date to the hard Expiry date.

The options for what inventory is available come from the Data Versions file, and is presented as a result of the inventory command.

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**inventory** |

**Parameters**

* Flex1=StripCode[,Quantity]
* Flex2=StripCode[,Quantity]
* Flex3=StripCode[,Quantity]
* Flex4=StripCode[,Quantity]

**Examples:**

* /xf2/v1/inventory?Flex1=A01&Flex2=B01&Flex3=C01&Flex4=D01

this indicates that all 4 bottles are being opened for the first time, using the first available batch of strip codes (A01, B01, C01, D01)

* /xf2/v1/inventory?Flex3=C02,100

this indicates that a new Flex3 bottle has been started, containing the default number of 100 strips.

* /xf2/v1/inventory

this will return the open bottle strip inventory without changing any quantities.

**Response**

{  
   "requestCommand":"inventory",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
   "flexStrips": {

    "stripCodes": {  
     "Flex1" : [  
      {"code": "A01", "status", "Optimal ❘ PastBestByDate ❘ Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A03", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex2" : [  
      {"code": "B01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
     ],  
     "Flex3" : [  
      {"code": "C01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "C02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex4" : [  
      {"code": "D02", "status", "Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D03", "status", "PastBestByDate", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D04", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ]  
    },  
    "openBottleStripInventory": {  
     "Flex1" : { "code": "A01", "count": 49 },  
     "Flex2" : { "code": "B01", "count": 49 },  
     "Flex3" : { "code": "C02", "count": 99 },  
     "Flex4" : { "code": "D01", "count": 48 }  
    }  
   }  
}

*status*

Call the status command until results are ready.

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**status** |

**Response**

{  
   "requestCommand":"status",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
   "results": {  
     "errorCode":0|n,  
     "errorMsg":"",  
     "strips: [  
       {  
       "stripId":"Flex1 | Flex2 | Flex3 | Flex4",  
       "stripCode":"A01"  
       },  
       { etc for each strip },  
     ],  
     "data: [  
       {  
         "analyteId": "Fe | Salt | Bo | Cu | FC | TC | pH | TA | TH | CYA | PO4",  
         "analyteName": "Iron | Salt | Borate | Copper | Free Chlorine | Total Chlorine | pH | Total Alkalinity | Total Hardness | Cyanuric Acid | Phosphate",  
         "valueStatus": "Exact | UnderRange | OverRange | OutOfRange | Invalid | Untested",  
         "decimals": 0 | 1,  
         "valueAsString": "1.3",  
         "units": "ppm | ppb| (blank for pH)",  
         "errorMsg": ""  
       },  
       { etc. for each analyte }  
       ]  
   },  
   "flexStrips": {  
    "openBottleStripInventory": {  
     "Flex1" : { "code": "A01", "count": 49 },  
     "Flex2" : { "code": "B01", "count": 49 },  
     "Flex3" : { "code": "C02", "count": 99 },  
     "Flex4" : { "code": "D01", "count": 48 }  
    }  
   }  
}

**Response Notes**

* ErrorCode 0 means there was no error. See error code table below.
* For untested analytes, valueStatus will be "Untested" and no other values will be given.
* The systemMode values "PrepareToDip" and "Dipping" only apply when animation=server

*newtest*

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**newtest** |

**Optional Parameter**

* animation=client|server

The default animation is "client."

**Optional Known Concentration Values**

When running diagnostics, provide additional information to be used for comparison against the new water test.

* reference=char(20)

Alphanumeric reference, up to 20 characters; required when providing known concentration values.

* sample=PoolSpaWater |PreparedSample | LabStandard |TapWater | DryTestStrip | BottledWater | DIWater | Unknown
* method=None | 2005Kit | Colorimeter | LabStandard | Spin | EyeReadTestStrip | Other
* temperature=75F|24C

provide the temperature as a number, followed by F or C for Fahrenheit or Celcius

* Pairs of known analyte=value measurements

e.g. Fe=1.0&FC=2.0

**Examples:**

* /xf2/v1/newtest

this defaults to animation=client

* /xf2/v1/newtest?animation=server

this indicates animation should be managed by the server

* /xf2/v1/newtest?reference=123&sample=PoolSpaWater&method=Spin&temperature=80F&Fe=1.0

**Response**

{  
   "requestCommand":"newtest ",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
}

*cancel*

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**cancel** |

**Response**

 {  
   "requestCommand":"cancel ",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
}

*update*

This command lets the client tell the **Xpress Flex Connect** software to update itself.

**Request**

|  |  |
| --- | --- |
| GET | http://localhost:61264/xf2/v1/**update** |

**Response**

The client should not rely on receiving a response for 4 to 10 seconds.

{  
   "requestCommand":"update",  
   "requestErrorCode": 0 ❘ n,  
   "requestErrorMsg":"",  
   "deviceConnected":true ❘ false,  
   "systemMode": "NoDevice ❘ ReadyHidden ❘ ReadyVisible ❘ PrepareToDip ❘ Dipping ❘  
      InsertingShuttle ❘ Analyzing ❘ Results ❘ UpdatingFirmware ❘ UnexpectedError",  
   "systemErrorCode": 0 ❘ n,  
   "systemErrorMsg":"",  
  
   "details":{  
    "softwareVersion":"2.1.0.0",  
    "apiVersion":"1.0",  
    "dataVersion":"1.0",  
    "documentation":"[https://xpressflexseries2api.taylortechnologies.com](https://xpressflexseries2api.taylortechnologies.com/)",  
    "deviceName":"XpressFlex",  
    "deviceSeries":2,  
    "deviceSerialNo":"XF000001",  
    "deviceFirmware": "0.14",  
    "animationAvailable":true ❘ false,  
    "newDataAvailable":true ❘ false,  
    "newDataVersion":"1.1",  
    "newExeAvailable":true ❘ false,  
    "newExeVersion":"2.1.1.5",  
    "newFirmwareAvailable":true ❘ false,  
    "newFirmwareVersion":"0.15",  
    "exeName":"XPressFlexConnect.exe ❘ XPressFlexConnectService.exe ❘ sureTREATXpress.exe",  
   },  
   "flexStrips": {  
    "stripCodes": {  
     "Flex1" : [  
      {"code": "A01", "status", "Optimal ❘ PastBestByDate ❘ Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "A03", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex2" : [  
      {"code": "B01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
     ],  
     "Flex3" : [  
      {"code": "C01", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "C02", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ],  
     "Flex4" : [  
      {"code": "D02", "status", "Expired", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D03", "status", "PastBestByDate", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} },  
      {"code": "D04", "status", "Optimal", "bestBy": "yyyymmdd", "expiry": "yyyymmdd"} }  
     ]  
    },  
    "openBottleStripInventory": {  
     "Flex1" : { "code": "A01", "count": 49 },  
     "Flex2" : { "code": "B01", "count": 49 },  
     "Flex3" : { "code": "C02", "count": 99 },  
     "Flex4" : { "code": "D01", "count": 48 }  
    }  
  }

**Response Notes**

* Action occurs only if there is a new software EXE or data file recommended.
* The user does not need to click any buttons. The auto-update process is automatic.
* The new software EXE and/or the new data file is downloaded. A splash screen is shown with captions indicating what is being downloaded.
* Backup copies of the software EXE and/or the data file are saved.
* Once the download is complete, the software relaunches the new version of itself. Any new data file will be loaded and used for subsequent water tests.

Request and System Error Codes

|  |  |  |
| --- | --- | --- |
| **Code** | **Message** | **Request, System** |
| 0 | Success (no error) |  |
| 100 | Update In Process | System |
| 101 | Update Required | System |
| 200 | Test in progress | Request /newtest |
| 201 | No test has been started | Request /cancel |
| 202 | Server animation not available | Request /newtest (while in animation=server mode ) |
| 203 | Device is not connected | Request /newtest |
| 300 | Invalid Set of Strip Codes | Request /inventory |
| 900 | Access Denied | Any invalid or malformed request |
| 901 | Unexpected Error | System |

Browser-Based interface to XpressFlex

Taylor Technologies provides xfConnect.js library to facilitate working with the XFConnect.exe API. This is an optional tool to make it easier to work with the API.

xfConnect.js

This JavaScript wraps the API commands for convenience. Instead of calling the API command sequence ( about, status, newtest, status... ), the developer can call **xfNewWaterTest('')** and implement the onResults event handler to handle and/or display the water test results.

**Promise-based JavaScript**

The xfConnect.js code uses modern yet widely supported promise-based syntax to implement the REST calls, handle errors and process responses.

**Minimal Coding**

1. **Properties**

There are two essential properties to configure first.

* 1. xfConnect.**animationInClient** = true | false; ( default true )

Set this to true when you want the countdown timers displayed in the browser.

* 1. xfConnect.**statusPollIntervalMs** integer

Set this to 0 when you do not want any automatic polling, or, set this to the number of milliseconds for automatic retries to look for status information or results.

1. **Methods**

There is one method to call to start a new water test.

* 1. **xfNewWaterTest('')**;

This will call [initialize](https://w01.href.com/wiki/Xpress_Flex_Series_2_API#initialize), [newtest](https://w01.href.com/wiki/Xpress_Flex_Series_2_API" \l "newtest) and [status](https://w01.href.com/wiki/Xpress_Flex_Series_2_API#status). The status command is called repeatedly, until results are ready or an error happens.

1. **Events**

Always customize the onError event handler. Either customize onSuccess or 7 detailed event handlers. (See Public Event Handlers below.)

* 1. **onError(error)**

This is called if there are error conditions.

* 1. **onSuccess(response)**

This is called for all circumstances other than an error.

Note: **onResults(response)** is called once the results are ready. The response object has everything required. The definition of the response can be seen above (see [status command](https://w01.href.com/wiki/Xpress_Flex_Series_2_API#status); look at results.data within the JSON).

**Response Object**

The response object provides pre-parsed information from the API Command response.

command: xfCommand,

mode: json.systemMode,

details: json.details,

results: json.results,

responseJson: atomicResponse.jsonOrText,

xhr: atomicResponse.xhr

**Error Object**

The error object includes the following properties:

xfCommand: string,

deviceConnected: boolean,

errorType: string,

errorCode: integer,

errorMsg: string,

responseJson: null,

xhr: xhrObject

**Detailed Feature Documentation**

The structure and functionality of xfConnect.js closely mirrors the API defined here.

**Public Properties**

* xfConnect.**animationInClient**, boolean, default true
* xfConnect.**apiPort** integer, default 61264
* xfConnect.**statusPollIntervalMs**, integer

**Public Methods**

* xfConnect.**about()**

optionally use this to display details such as the firmware version

* xfConnect.**cancel()**

optionally use this to enable the end-user to cancel a test after it has started

* xfConnect.**initialize()**
* xfConnect.**inventory(**params**)**
* xfConnect.**newtest()**
* xfConnect.**status()**

**Public Event Handlers**  
Once the response is available, these event handlers are called they are if assigned.

* xfConnect.**onAbout**
* xfConnect.**onCancel**
* xfConnect.**onError**
* xfConnect.**onInitialize**
* xfConnect.**onInventory**
* xfConnect.**onNewTest**
* xfConnect.**onResults**
* xfConnect.**onStatus**
* xfConnect.**onSuccess**

Optionally, the developer can connect all of these events except onError to a single JavaScript function which has a switch statement. Here is an example of a shared handler:

function on\_xfConnect\_Success( response ) {

switch(response.command) {

case "about":

break;

case "initialize":

break;

case "inventory":

break;

case "newtest":

break;

case "status":

break;

case "cancel":

break;

}

**Error Handler**

An example for the error handler follows:

function on\_xfConnect\_Error( error ) {

var

s;

/\*\*/

s = '';

switch( error.errorType ) {

case "apiConnectionError":

if ( error.xhr.status == 0 )

s = 'Xpress Flex Connect is not running'

else

s = error.errorMsg;

break;

case "noDeviceError":

s = error.errorMsg;

break;

case "apiResponseFormattingError":

s = error.errorMsg;

break;

case "apiUsageError":

s = error.errorMsg;

break;

default:

s = error.errorType + ':' + errorMsg;

}

alert( s );

}

Please see the xfConnect.js source code for more information.