QRC

QRC (Q-Sys Remote Control) is an Unicode based TCP/IP control protocol. The client connect to the core (or emulator) on port 1710 (likely to be changed) and sends JSON RPC 2.0 (http://www.jsonrpc.org/specification) null terminated commands.

Error Codes

The following error codes can be returned as the code value in a JSON-RPC error object.

Code	Details
-32700	Parse error. Invalid JSON was received by the server.
-32600	Invalid request. The JSON sent is not a valid Request object.
-32601	Method not found.
-32602	Invalid params.
-32603	Server error.
2	Invalid Page Request ID
3	Bad Page Request - could not create the requested Page Request
4	Missing file
5	Change Groups exhausted
6	Unknown change croup
7	Unknown component name
8	Unknown control
9	Illegal mixer channel index
10	Logon required

NoOp

Simple 'do nothing' method useful for making sure the socket is left open

```
{
  "jsonrpc":"2.0",
  "method":"NoOp",
  "params":{
  }
}
```

Logon

```
	{
	  "jsonrpc":"2.0",
	  "method":"Logon",
	  "params":{
	    "User":"username",
	    "Password":"1234"
	  }
	}
```

Status

The status of the core is automatically sent when a client connects to the QRC port as well as whenever the status changes.

```
	{
	  "jsonrpc":"2.0",
	  "method":"EngineStatus",
	  "params":{
	    "State":"Active",
	    "DesignName":"SAF-MainPA",
	    "DesignCode":"qALFilm6IcAz",
	    "IsRedundant":false,
	    "IsEmulator":true
	  }
	}
```

If for some reason the client wants to request the current status they can use the StatusGet method

DataTypes

EngineStatus

State	one of the following strings - "Idle", "Active", "Standby"
DesignName	name of the currently running design
DesignCode	GUID of the currently running design
IsRedundant	true if the design is configured to be a redundant design
IsEmulator	true if design is currently running in the emulator

Methods

StatusGet

returns the EngineStatus of the core.

```
	{
  "jsonrpc": "2.0",
  "method": "StatusGet",
  "id": 1234,
  "params": 0
}
Response
{
  "jsonrpc":"2.0",
  "id":1234
  "result":{
    "Platform":"Core 500i",
    "State":"Active",
    "DesignName":"SAF-MainPA",
    "DesignCode":"qALFilm6IcAz",
    "IsRedundant":false,
    "IsEmulator":true,
    "Status":{
      "Code":0
      "String":"OK"
    }
  }
}
```

Control

DataTypes

ControlValue

Name	name of control, relative to component
Value	value of control. Can be number, string or boolean
String	string representation of control - only used in responses
Ramp	optional ramp time used to set the control

Methods

Control.Set

params	single ControlValue

Example

```
{
 "jsonrpc": "2.0",
 "id": 1234,
 "method": "Control.Set",
 "params": {
```

```
           "Name": "MainGain",
           "Value": -12
         }
        }
```

Control.Get

params	array of Named Control strings
result	array of ControlValues

Example

```
        {
         "jsonrpc": "2.0",
         "id": 1234,
         "method": "Control.Get",
         "params": ["MainGain"]
        }
        RESPONSE
        {
         "jsonrpc": "2.0",
         "id": 1234,
         "result": [
           {
             "Name": "MainGain",
             "Value": -12
           }
          ]
        }
```

```
        {
         "jsonrpc": "2.0",
         "id": 1234,
         "method": "Control.Get",
         "params": ["MainGain", "MainMute"]
        }
        RESPONSE
        {
         "jsonrpc": "2.0",
         "id": 1234,
         "result": [
           {
             "Name": "MainGain",
             "Value": -12
             "String" : "-12.0dB"
           },
           {
             "Name": "MainMute",
             "Value": false,
             "String" : "Unmuted"
           }
          ]
        }
```

Component Control

DataTypes

Component

Name	name of component
Controls	array of ControlValues

Methods

Component.Get

Gets one or more controls on a named component

```
        {
          "jsonrpc": "2.0",
          "id": 1234,
          "method": "Component.Get",
          "params": {
            "Name": "My APM",
            "Controls": [
              { "Name": "ent.xfade.gain" }
            ]
          }
        }
        RESPONSE
        {
          "jsonrpc": "2.0",
          "result": {
            "Name": "My APM",
            "Controls": [
            {
              "Name": "ent.xfade.gain",
              "Value": -100.0,
              "String": "-100.0dB"
              "Position": 0
            }
            ]
          }
        }
```

Component.Set

params	Component object

Examples

Set a single control on a single component

```
        {
         "jsonrpc": "2.0",
         "id": 1234,
         "method": "Component.Set",
         "params": {
           "Name": "My APM",
           "Controls": [
             {
               "Name": "ent.xfade.gain",
               "Value": -100.0,
               "Ramp": 2.0
             }
           ]
         }
        }
```

Multiple controls on a single component

```
        {
         "jsonrpc": "2.0",
         "id": 1234,
         "method": "Component.Set",
         "params": {
           "Name": "My APM",
           "Controls": [
             {
               "Name": "ent.xfade.gain",
               "Value": -100.0,
               "Ramp": 2.0
             },
             {
               "Name": "bgm.xfade.gain",
               "Value": 0.0,
               "Ramp": 1.0
             }
           ]
         }
        }
```

Component.GetComponents

params	NA

Examples

Gets a list of all named components in the design, along with their type and properites

```
        {
          "jsonrpc": "2.0",
          "method": "Component.GetComponents",
          "params": "test",
          "id": 1234
        }
        RESPONSE
        {
            "jsonrpc": "2.0",
            "result": [
                {
                    "Name": "APM ABC",
                    "Type": "apm",
                    "Properties": []
                },
                {
                    "Name": "My Delay Mixer",
                    "Type": "delay_matrix",
                    "Properties": [
                        {
                            "Name": "n_inputs",
                            "Value": "8"
                        },
                        {
                            "Name": "n_outputs",
                            "Value": "8"
                        },
                        {
                            "Name": "max_delay",
                            "Value": "0.5"
                        },
                        {
                            "Name": "delay_type",
                            "Value": "0"
                        },
                        {
                            "Name": "linear_gain",
                            "Value": "False"
                        },
                        {
                            "Name": "multi_channel_type",
                            "Value": "1"
                        },
                        {
                            "Name": "multi_channel_count",
                            "Value": "8"
                        }
                    ]
                }
            ],
            "id": 1234
        }
```

Change Groups

Add controls to change group via Named Controls

```
        {
          "jsonrpc": "2.0",
          "id": 1234,
          "method": "ChangeGroup.AddControl",
          "params": {
            "Id": "my change group"
            "Controls" : [
              "some control", "another control"
            ]
          }
        }
```

Add controls to change group via Named Component

```
        {
          "jsonrpc": "2.0",
          "id": 1234,
          "method": "ChangeGroup.AddComponentControl",
          "params": {
            "Id": "my change group"
            "Component" : {
              "Name": "My Component",
              "Controls": [
                { "Name": "gain" },
                { "Name": "mute" }
              ]
            }
          }
        }
```

Remove controls from change group via Named Controls

```
        {
          "jsonrpc": "2.0",
          "id": 1234,
          "method": "ChangeGroup.Remove",
          "params": {
            "Id": "my change group"
            "Controls" : [
              "some control"
            ]
          }
        }
```

Poll change group

```
        {
          "jsonrpc": "2.0",
          "id": 1234,
          "method": "ChangeGroup.Poll",
          "params": {
            "Id": "my change group"
          }
        }
        RESPONSE
        {
          "jsonrpc": "2.0",
          "id": 1234,
          "result": {
            "Id": "my change group",
            "Changes": [
              {  // Named control return value
                "Name": "some control",
                "Value": -12
                "String": "-12dB"
              },
              { // Named component return value
                "Component": "My Component",
                "Name": "gain",
                "Value": -12
                "String": "-12dB"
              }
            ]
          }
        }
```

Destroy change group

```
{
  "jsonrpc": "2.0",
  "id": 1234,
  "method": "ChangeGroup.Destroy",
  "params": {
    "Id": "my change group"
  }
}
```

Invalidate change group - causes all controls to be resent

```
{
  "jsonrpc": "2.0",
  "id": 1234,
  "method": "ChangeGroup.Invalidate",
  "params": {
    "Id": "my change group"
  }
}
```

Clear change group - removes all controls

```
{
  "jsonrpc": "2.0",
  "id": 1234,
  "method": "ChangeGroup.Clear",
  "params": {
    "Id": "my change group"
  }
}
```

Set up automatic polling

```
{
  "jsonrpc": "2.0",
  "id": 1234,
  "method": "ChangeGroup.AutoPoll",
  "params": {
    "Id": "my change group"
    "Rate": 5
  }
}
WILL RESULT IN FOLLOWING UPDATE EVERY 5 SECONDS
{
  "jsonrpc": "2.0",
  "id": 1234,
  "result": {
    "Id": "my change group",
    "Changes": [
      {  // Named control return value
        "Name": "some control",
        "Value": -12
        "String": "-12dB"
      },
      { // Named component return value
        "Component": "My Component",
        "Name": "gain",
        "Value": -12
        "String": "-12dB"
      }
    ]
  }
```

```
      }
```

Mixer Control

The mixer control API uses a string specification to determine which inputs and outputs to apply changes to. The syntax supports either space or comma separated numbers, ranges of numbers or all (*). It's supports negation of selection with the ! operator. Here's a few examples

*	everything
1 2 3	channels 1, 2, 3
1-6	channels 1 through 6
1-6 9	channels 1 through 6 and 9
1-3 5-9	channels 1 through 3 and 5 through 9
1-8 !3	channels 1 through 8 except 3
* !3-5	everything but 3 through 5

DataTypes

CrossSpec

Name	name of mixer
Inputs	string specification of mixer inputs
Outputs	string specification of mixer outputs
Value	value to set to control
Ramp	ramp time to use for control set

InputSpec

Name	name of mixer
Inputs	string specification of mixer outputs
Value	value to set to control
Ramp	ramp time to use for control set

OutputSpec

Name	name of mixer
Outputs	string specification of mixer outputs
Value	value to set to control
Ramp	ramp time to use for control set

CueSpec

Name	name of mixer
Cues	string specification of mixer cues
Value	value to set to control
Ramp	ramp time to use for control set

InputCueSpec

Name	name of mixer
Cues	string specification of mixer cues
Inputs	string specification of mixer inputs
Value	value to set to control
Ramp	ramp time to use for control set

Methods

Mixer.SetCrossPointGain

params	CrossSpec

Mixer.SetCrossPointDelay

params	CrossSpec

Mixer.SetCrossPointMute

params	CrossSpec

Mixer.SetCrossPointSolo

params	CrossSpec

Mixer.SetInputGain

params	InputSpec

Mixer.SetInputMute

params	InputSpec

Mixer.SetInputSolo

params	InputSpec

Mixer.SetOutputGain

params	OutputSpec

Mixer.SetOutputMute

params	OutputSpec

Mixer.SetCueMute

params	CueSpec

Mixer.SetCueGain

params	CueSpec

Mixer.SetInputCueEnable

params	InputCueSpec

Mixer.SetInputCueAfl

params	InputCueSpec

Examples

Set all crosspoints on mixer to -100db over 5 seconds

```
{
 "jsonrpc": "2.0",
 "method": "Mixer.SetCrossPointGain",
 "id": 1234,
 "params": {
   "Mixer": "Parade",
   "Inputs": "*",
   "Outputs": "*",
   "Value": -100.0,
   "Ramp": 5.0
 }
}
```

Mute inputs 4-6

```
{
  "jsonrpc": "2.0",
  "method": "Mixer.SetInputMute",
  "params": {
    "MixerName": "Parade",
    "Inputs": "4-6",
    "Value": true,
    "Ramp": 0.0
  },
  "id": 1234
}
```

Loop Player Control

The loop player control API allows a remote system to cue up file playback into a named loop player.

DataTypes

FileSpec

Name	name of file to play
Mode	mono | stereo
Output	output of loop player to play file out of

JobSpec

Name	name of loop player

StartTime	time of day in seconds of when to start job
Files	array of FileSpecs
Loop	if true will automatcially loop file
Seek	optional time in seconds to seek into each file before playback

Methods

LoopPlayer.Start

params	JobSpec

```
{
    "jsonrpc": "2.0",
    "method": "LoopPlayer.Start",
    "params": {
        "Files": [
            {
                "Name": "Audio/mainloop.wav",
                "Mode": "mono",
                "Output": 1
            }
        ],
        "Name": "test",
        "StartTime": 62600,
        "Loop": false,
        "Log": true
    },
    "id": 1234
}
```

LoopPlayer.Stop

params	JobSpec

```
{
    "jsonrpc": "2.0",
    "method": "LoopPlayer.Stop",
    "params": {
        "Name": "test",
        "Outputs": [ 1, 3, 4 ],
        "Log": true
    },
    "id": 1234
}
```

LoopPlayer.Cancel

params	JobSpec

```
{
    "jsonrpc": "2.0",
    "method": "LoopPlayer.Cancel",
    "params": {
        "Name": "test",
        "Outputs": [ 1, 3, 4 ],
        "Log": true
    },
    "id": 1234
}
```