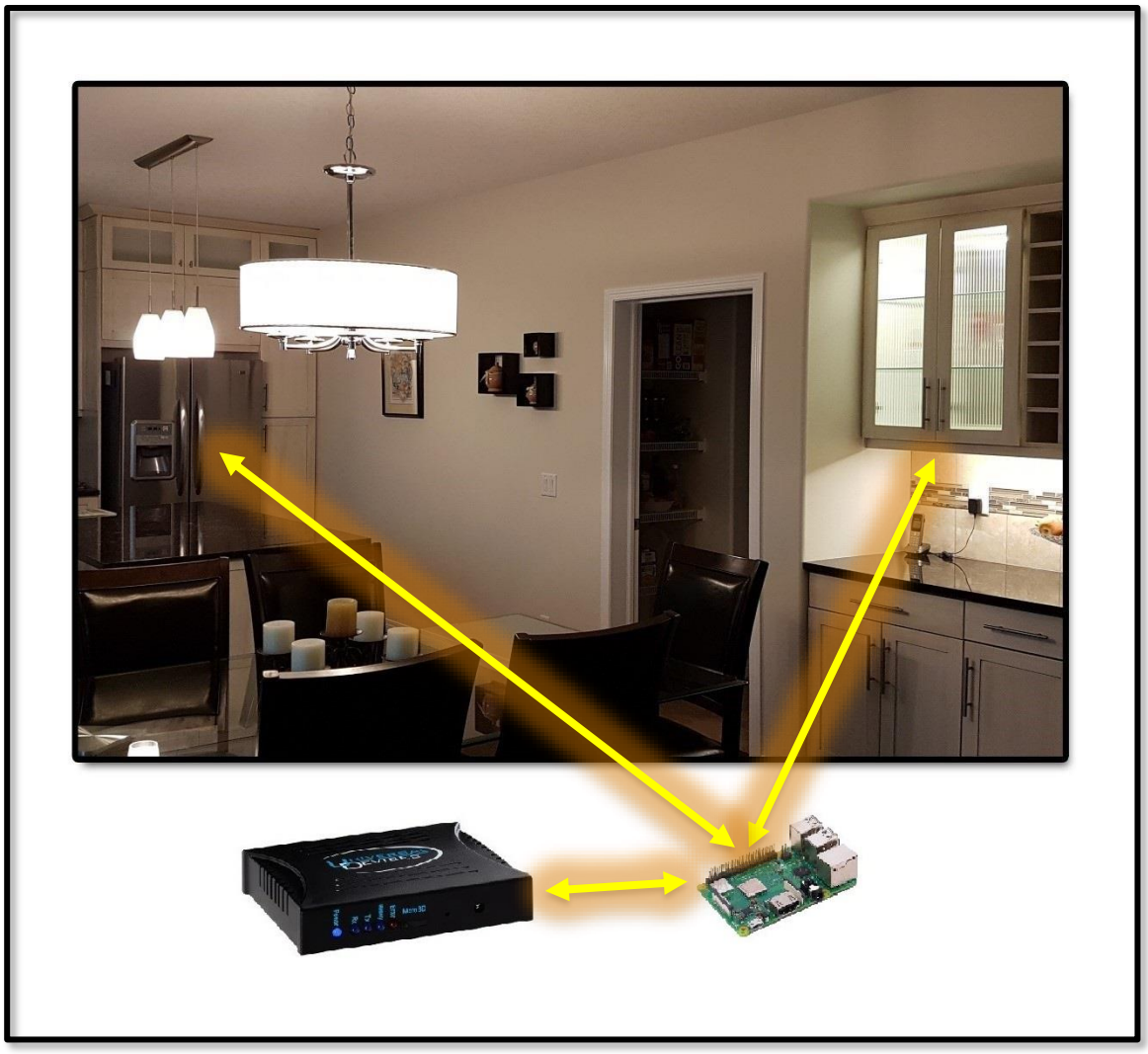


The ISY-994i Developer's Cookbook

Table of Contents

1 Disclaimer 13

2 Introduction 13

3 Developer's SDK Packages and Examples 15

3.1 Official SDK [5.0.4] 15

3.2 C# Examples 15

3.3 Perl 15

3.4 PHP 15

3.5 Python 15

3.6 Polyglot 15

4 NodeServers 16

4.1 Creating a NodeServer 16

4.1.1 Getting Started 16

4.1.2 Create the NodeServer folder 16

4.1.3 Writing the Code 19

4.1.4 You built your NodeServer, now what? 19

5 Polyglot 27

5.1 Polyglot Quick Install Notes 27

5.2 Installing Raspberry PI 29

5.3 Installing Polyglot 32

5.4 Installing NodeServers 35

5.5 Polyglot Virtual Node Server Framework 41

5.5.1 Usage 41

5.5.1.1 Installation 41

5.5.1.1.1 Pure Python (aka non-compiled source) 41

5.5.1.1.2 COMPILED version (aka all-in-one) 43

5.5.1.1.3 Command line flags 44

5.5.1.1.4 OSX Instructions 44

5.5.1.1.5 Start Polyglot on Boot 44

5.5.1.1.6 Logging locations 45

5.5.1.2 User Interface 46

5.5.1.2.1 Settings 47

5.5.1.2.2 Adding Node Server 47

5.5.1.2.3 Managing Node Servers 48

5.5.1.2.4 Viewing Polyglot Log 49

5.5.2 Node Server Development 49

5.5.2.1 Background 49

5.5.2.2 File Structure 49

5.5.2.3 Server Metadata 50

5.5.2.4 Python Development 52

5.5.2.5 Python Polyglot Library 53

5.5.2.5.1 Summary 53
5.5.2.5.2 Custom Node Types 53
5.5.2.6 Polyglot API Implimentation 58
5.5.2.7 Node Server Classes 64
5.5.2.8 Helper Functions 73
5.5.2.9 MQTT 73
5.5.2.9.1 Usage 73
5.5.2.9.2 MQTT Subsystem Class 75
5.5.3 Python Node Server Example 75
5.5.3.1 Node Type Definition 75
5.5.3.2 Node Server Creation 81
5.5.3.3 Starting the Node Server 85
5.5.3.4 Installing the Node Server 86
5.5.3.5 Custom Node Server Configuration File 86
5.5.4 Polyglot Node Server API 86
5.5.4.1 General Format 87
5.5.4.2 Node Server STDIN - Polyglot to Node Server 87
5.5.4.3 Node Server STDOUT - Node Server to Polyglot 89
5.5.4.4 Node Server STDERR - Node Server to Polyglot 90
5.5.5 Polyglot Methods and Classes 90
5.5.5.1 Module 90
5.5.5.1.1 Config Manager 90
5.5.5.1.2 Core 91
5.5.5.1.3 Nodeserver API 92
5.5.5.1.4 Nodeserver Helpers 112
5.5.5.1.5 Nodeserver Manager 112
5.5.5.1.6 Node Server Manager 115
5.5.5.1.7 Utils 117
5.5.6 Optional Components 117
5.5.6.1 Docker 117
5.5.6.1.1 Linux x86_64 118
5.5.6.1.2 Raspberry Pi 120
5.6 Polyglot Source Code 124
5.7 Node Hints Documentation 124
6 Java 126
6.1 ISY Java Web Services Tutorial 126
6.1.1 How to program/consume UDI web services in Java 126
6.1.2 Online Documentation 127
6.1.3 Requirements 128
6.1.4 Process 128
6.1.4.1 Basic setup 128
6.1.5 WSDL Import 133
6.1.5.1 Changes to the wsdl file 133
6.1.5.2 Import the updated udiws30_patched.wsdl file 136
6.1.5.3 Import the ELK WSDL file 138
6.1.6 Write your Java Application 138
6.1.6.1 Create the Service And Port 139
6.1.6.2 HTTP Basic Authentication 139
6.1.6.3 Call Web Services 139
6.1.7 Sample Java App 141

6.1.8 Debug SOAP Messages With TCPMon 143
6.2 Java REST Services Requester Example 145
6.2.1 Background 145
6.2.2 Scope of this REST Client 146
6.2.3 Java Source Code 146
6.2.3.1 Extending this class to wrap around the ISY REST services 147
6.2.3.2 HTTP Authentication 147
6.2.4 Dependencies - Apache Commons Codec 149
6.2.4.1 Using Android Base64 Instead 150
6.2.5 Base Class - RestRequester.java 150
6.2.6 ISY-994 Extension Class - ISYRestRequester.java 159
6.2.7 isyRest.jar - source and classes 164
7 Rest 165
7.1 ISY REST Interface 165
7.1.1 Notes 165
7.1.2 Return Values / Codes 165
7.1.3 Configuration 165
7.1.4 Nodes 169
7.1.5 Properties 174
7.1.6 Commands 174
7.1.7 Status 176
7.1.8 Query 177
7.1.9 X10 177
7.1.10 Programs 178
7.1.11 Logs 180
7.1.12 Variables 181
7.1.13 Subscriptions (Web Sockets) 182
7.1.14 Modules 186
7.1.14.1 Electricity 186
7.1.14.2 Climate 187
7.1.14.3 Networking 189
7.1.14.4 Misc 191
7.1.14.5 Zigbee 192
7.1.14.6 Z-Wave 193
7.1.14.7 Gas 196
7.1.15 Batch Commands 196
7.1.16 REST Interface for GreenEye Monitor 200
7.1.17 REST Interface for EM3 201
7.1.18 Portal Integration 205
7.1.19 ELK Integration 206
7.1.19.1 Area Commands 206
7.1.19.2 Zone Commands 208
7.1.19.3 General Commands 209
7.1.19.4 System Commands 210
7.1.19.5 Keypad Commands 210
7.1.19.6 Output Commands 211
7.1.19.7 Audio Commands 211
7.1.19.8 Voice Announcement Commands 212
7.1.19.9 Thermostat Commands 212
7.1.20 OpenADR (VEN) 213
7.1.21 Billing 215

8 ISY Developer's Manual 216

8.1 Introduction 216

8.2 Basic Concepts 217

8.2.1 Control 217
8.2.2 Action 217
8.2.3 Node 218
8.2.4 Group/Scene 219
8.2.5 Putting it Together 219
8.2.6 ISY Messages and Web Services 219

8.3 Discovering ISY and its Resources 220

8.3.1 Discovering ISY Using UPnP Search 220
8.3.2 Listening for ISY Advertisements on the Network 220
8.3.3 Capturing ISY Resources 222
8.3.4 ISY Configuration Resource 223
8.3.4.1 Modules (Features) 231
8.3.5 ISY Nodes Configuration Resource 232
8.3.5.1 Types of Nodes/Parents 233
8.3.5.2 Node (<node>) 233
8.3.5.3 Group/Scene (<group>) 237
8.3.5.4 Folder (<folder>) 238

8.4 Communicating with ISY 238

8.5 Events 242

8.5.1 Device Status (control = Device Property) 242
8.5.2 Heartbeat (control = "_0") 242
8.5.3 Trigger Events (control = "_1") 243
8.5.4 Driver Specific Events (control = "_2") 245
8.5.5 Node Changed/Updated (control = "_3") 245
8.5.6 System Configuration Updated (control = "_4") 249
8.5.7 System Status Updated (control = "_5") 249
8.5.8 Internet Access Status (control = "_6") 249
8.5.9 Progress Report (control = "_7") 250
8.5.10 Security System Event (control = "_8") 250
8.5.11 System Alert Event (control = "_9") 251
8.5.12 OpenADR and Flex Your Power Events (control = "_10") 251
8.5.13 Climate Events (control = "_11") 252
8.5.14 AMI/SEP Events (control = "_12") 265
8.5.15 External Energy Monitoring Events (control = "_13") 265
8.5.16 UPB Linker Events (control = "_14") 266
8.5.17 UPB Device Adder State (control = "_15") 267
8.5.18 UPB Device Status Events (control = "_16") 267
8.5.19 5.17 Gas Meter Events (control = "_17") 267
8.5.20 5.18 Zigbee Events (control = "_18") 267
8.5.21 ELK Events (control = "_19") 268
8.5.22 Device Linker Events (control = "_20") 268
8.5.23 Z-Wave Events (control = "_21") 268
8.5.24 Billing Events (control = "_22") 268
8.5.25 Portal Events (control = "_23") 268

8.6 REST Interface 269

8.6.1 Batch Commands 269

8.6.2 Configuration 269
8.6.3 Nodes 270
8.6.4 X10 270
8.6.5 Properties 271
8.6.6 Status 271
8.6.7 Query 272
8.6.8 Programs 272
8.6.9 Modules 273
8.6.10 Security 273
8.6.11 Energy Management AMI/Smart Grid/SEP 273
8.6.12 Gas 273
8.6.13 Logs 274
8.6.14 Variables 274
8.6.15 Zigbee 275
8.6.16 ELK 276
8.6.17 Z-Wave 277
8.6.18 Subscription using Web Sockets 277

8.7 Programs 277
8.7.1 Program IDs 278
8.7.2 Key 278
8.7.3 Details 279
8.7.4 Examples 280

8.8 Logs 281
8.8.1 System Log (/rest/log) 282
8.8.2 Error Log (/rest/log/error) 282
8.8.3 Converting NTP Formatted Time 283
8.8.4 Log/Error Types 285

8.9 Web Sockets and Subscriptions 286

8.10 Appendix A – INSTEON Device Categories/Subcategories 286
8.10.1 A1. Device Categories 286
8.10.2 A2. Device Sub-Categories 286

8.11 Appendix B – UPB Device Types 287
8.11.1 B1. PCS Device Types 287

9 ISY Node Server Developer's Manual 289

9.1 Introduction 289

9.2 What is a Node? 289

9.3 Node Server Configuration on ISY 289
9.3.1 Files 289
9.3.2 Network Connection 290
9.3.2.1 From Isy to Node Server 290
9.3.2.2 From Node Server to Isy 290
9.3.2.3 Responses 291
9.3.3 Serial Connection 291

9.4 Required API support in Node Server 291
9.4.1 General 291
9.4.1.1 Request IDs 292

9.4.1.2 Node Addresses 292
9.4.2 Install 293
9.4.3 Query node 293
9.4.4 Get Node Status Values 293
9.4.5 Add All Nodes 294
9.4.6 Reports from ISY 294
9.4.7 Run a command 295
9.5 REST support in ISY 296
9.5.1 Reporting status updates 296
9.5.2 Reporting a command 296
9.5.3 Node Management 297
9.5.4 Reporting ISY Request status 298
9.6 Natural Language Support (NLS) 299
9.6.1 General 299
9.6.2 Naming Convention Terminology 300
9.6.3 Device Name 300
9.6.4 Icons 301
9.6.5 Status Names 301
9.6.6 Command Names 301
9.6.7 Command Parameter Names 302
9.6.8 Other Names 302
9.6.9 Name mapped Values (Index, Percent) 302
9.6.10 Formatting in Programs 303
9.6.11 Commands 303
9.6.11.1 Command Formatting Examples 304
9.6.12 Status Conditions 305
9.6.13 Control Conditions 305
9.7 Appendix 306
9.7.1 Editors 306
9.7.2 Encoded Editor ID 307
9.7.3 Node Definitions 308
9.7.4 Icons 310
9.7.5 Driver Controls 311
9.7.6 Units of Measure 313
10 ISY ELK Integration Developer's Manual 321
10.1 Introduction 321
10.2 Getting Started 321
10.3 ELK Events (control = "_19") 322
10.3.1 Topology Changed (action = "1") 322
10.3.2 Area Event (action = "2") 323
10.3.3 Zone Event (action = "3") 323
10.3.4 Keypad Event (action = "4") 324
10.3.5 Output Event (action = "5") 324
10.3.6 System Event (action = "6") 325
10.3.7 Thermostat Event (action = "7") 325
10.4 REST Interface 326
10.4.1 Area Commands 326
10.4.2 Zone Commands 328

10.4.3 General Commands 329
10.4.3.1 System Commands 329
10.4.4 Keypad Commands 330
10.4.5 Keypad Commands 331
10.4.6 Audio Commands 332
10.4.7 Voice Announcement Commands 332
10.4.8 Thermostat Commands 333

11 ISY Z-Wave Integration Developer's Manual 334

11.1 Introduction 334

11.2 Getting Started 334

11.3 Z-Wave Control Events 335

11.4 Z-Wave Events (control = "_21") 336
11.4.1 System Status Events (action = "1.3") 336
11.4.2 Discovery - Inactive (action = "2.1") 336
11.4.3 Discovery - Inclusion (action = "2.2") 336
11.4.4 Discovery - Exclusion (action = "2.3") 337
11.4.5 Discovery - Primary Replication (action = "2.4") 337
11.4.6 Discovery - Learn Mode (action = "2.5") 337
11.4.7 General Status (action = "3.x.y") 337
11.4.8 General Error (action = "4.x.y") 338

11.5 Common REST Interface 338

11.6 REST Interface 339
11.6.1 Adding and Removing devices 339
11.6.2 General Commands 340
11.6.3 Node Information 342
11.6.4 Device Configuration Commands 344
11.6.5 Node Properties 344
11.6.6 Schedule and Security Properties 345

11.7 Appendix 347
11.7.1 Driver Details 347
11.7.2 Units of Measure 347
11.7.3 Node Types 348
11.7.4 Model Types 349

12 ISY OpenADR (VEN) Developer's Manual 350

12.1 Introduction 350

12.2 Putting it Together 350
12.2.1 Retrieve Configuration 351
12.2.2 Save Configuration 351
12.2.3 Retrieve Reporting Configuration (2.0b) 352
12.2.4 Save Reporting Configuration (2.0b) 352
12.2.5 Save Opt Schedules (2.0b) 352
12.2.6 Get notified of OpenADR Events 353
12.2.7 Retrieving OpenADR 2.0 Full Payload 354
12.2.8 Opting In and Out of Events (2.0a/2.0b) 355
12.2.9 Clearing All Events 355
12.2.10 Registration Service URLs (2.0b) 355

12.2.11 Report Service URLs (2.0b) 355
12.2.12 Opt Service URLs (2.0b) 356
12.2.13 Push URLs 356

13 ISY Portal Integration Developer's Manual 357

13.1 Introduction 357

13.2 Definitions and Portal URLs 357
13.2.1 Dispatcher URL 357
13.2.2 Proxy URL 358
13.2.3 Service/Account Request URL 359
13.2.4 2.4 Active Service/Account URL 360
13.2.5 Account/Service Approval URL 361
13.2.6 Account/Service Revocation URL 361
13.2.7 Subscription URL 362
13.2.8 Authentication and Authorization 362
13.2.9 Subscription 363

13.3 Process Flow 363
13.3.1 Putting It Together 364

13.4 REST Interface 365
13.4.1 /rest/whoami 365
13.4.2 /rest/portal/status 367
13.4.3 /rest/portal-server/requests/<ISY's UUID> 367
13.4.4 /rest/portal-server/active/<ISY's UUID> 367
13.4.5 /rest/portal-server/approve/<Account ID> 367
13.4.6 /rest/portal-server/revoke/<Account ID> 368
13.4.7 /rest/portal/approve/<Account ID>/<ISY's UUID>/Key 368
13.4.8 /rest/portal-server/revoke/<Account ID>/ISY's UUID/Key 368
13.4.9 Events 369
13.4.9.1 Portal Events (control = "_23") 369

14 ISY Energy Management System Developer's Manual 370

14.1 Introduction 370

14.2 DRLC (Demand Response Load Control) 370

14.3 Message 371

14.4 Price 371

14.5 Meter 371

14.6 Common Messages/Signals 371

14.7 Getting Started 372

14.8 SEP Event Lifecycle 373

14.9 Lifecycle of DR Events 375
14.9.1 Device Class 376
14.9.2 Duty Cycle 376
14.9.3 Opting In/Out 377
14.9.4 Invalidating a DR Event 377
14.9.5 Revert to the State Prior to DR Event 377

14.10 Configuration 377
14.11 Base Electricity Configuration 378
14.12 Message Configuration 378
14.13 Price Configurations 378
14.14 DRLC Configurations 378
14.15 Meter Configurations 378
14.15.1 ISY SEP Events (control = "_12") 378
14.15.2 Network Status Changed (action = "1") 378
14.15.3 Time Status Changed (action = "2") 378
14.15.4 New Message (action = "3") 379
14.15.5 Scheduled Message (action = "31") 379
14.15.6 Message Stopped (action = "4") 379
14.15.7 New Price (action = "5") 379
14.15.8 Scheduled Price (action = "51") 379
14.15.9 Price Stopped (action = "6") 379
14.15.10 New DR Event (action = "7") 379
14.15.11 Scheduled DR Event (action = "71") 380
14.15.12 DR Event Stopped (action = "8") 380
14.15.13 Metering Event (action = "9") 380
14.15.14 Metering Format Event (action = "10") 380
14.15.15 Fast Poll Mode (action = "110") 380
14.15.16 Normal Poll Mode (action = "111") 380
14.16 ISY Billing Events (control = "_22") 381
14.16.1 Cost/Usage Changed Event (action = "1") 381
14.17 REST Interface 381
14.17.1 Message 381
14.17.2 Price 382
14.17.3 DRLC 382
14.17.4 Zigbee SEP (Metering) 383
14.17.5 Billing 384
14.18 Logs 385
14.18.1 Message 386
14.18.2 Price 386
14.18.3 DRLC 387
14.18.4 Meter 387
15 ISY 994 Z Series Energy Monitoring Developer's Manual 388
15.1 Introduction 388
15.2 Getting Started 389
15.2.1 Configuring ISY 389
15.2.2 Configuring ECM1240 389
15.2.3 Configuring GreenEye Monitor 390
15.2.4 Configuring UDI EM3 390
15.3 3. Nodes, Properties and Events 390
15.3.1 ECM 1240/GreenEye Monitor Nodes 391
15.3.2 UDI EM3 Nodes 392

15.3.3 Events and Properties 393
15.3.4 Raw ECM140 Packet (control = _13 action = "7") 394
15.4 REST Interface 394
15.4.1 Zigbee Network 394
15.4.2 REST Interface for ECM 394
15.4.3 REST Interface for GreenEye Monitor 396
15.4.4 4.4 REST Interface for EM3 398
16 ISY 994 Z Series RCS Thermostat Support Scheduling Addendum 400
16.1 Introduction 400
16.2 Getting Started 400
16.2.1 Configuring ISY 401
16.2.2 Configuring RCS Zigbee Thermostat 401
16.3 Getting/Querying Schedules 402
16.4 4. Setting Schedules 404
16.5 5. Events 406
17 Soap / Web Services (WSDK) 407
17.1 Soap Error Codes 407
17.2 WSDL Core Web Services Documentation 408
17.2.1 AddDDNSHost 408
17.2.2 AddFolder 409
17.2.3 AddGroup 410
17.2.4 AddNode 411
17.2.5 ClearLastError 412
17.2.6 GetCurrentSystemStatus 413
17.2.7 GetDDNSHost 414
17.2.8 GetDebugLevel 415
17.2.9 GetISYConfig 416
17.2.10 GetLastError 420
17.2.11 GetNodesConfig 421
17.2.12 GetSMTPConfig 426
17.2.13 GetSceneProfiles 427
17.2.14 GetStartupTime 428
17.2.15 GetSystemDateTime 429
17.2.16 GetSystemOptions 430
17.2.17 GetSystemStatus 431
17.2.18 GetVariable 432
17.2.19 GetVariables 433
17.2.20 InternetAccess 434
17.2.21 IsDDNSHostAvail 435
17.2.22 IsSubscribed 436
17.2.23 MoveNode 437
17.2.24 Query 438
17.2.25 Reboot 439
17.2.26 RemoveDDNSHost 440
17.2.27 RemoveFolder 441
17.2.28 RemoveFromGroup 442

17.2.29 RemoveGroup 443
17.2.30 RemoveModem 444
17.2.31 RemoveNode 445
17.2.32 RenameFolder 446
17.2.33 RenameGroup 447
17.2.34 RenameNetwork 448
17.2.35 RenameNode 449
17.2.36 ReplaceDevice 450
17.2.37 ReplaceModem 451
17.2.38 RestoreDevice 452
17.2.39 RestoreDevices 453
17.2.40 SecuritySystemAction 454
17.2.41 SendHeartbeat 455
17.2.42 SendTestEmail 456
17.2.43 SetBatchMode 457
17.2.44 SetBatteryDeviceWriteMode 458
17.2.45 SetDebugLevel 459
17.2.46 SetLinkingMode 460
17.2.47 SetNTPOptions 461
17.2.48 SetNodeEnabled 462
17.2.49 SetNodePowerInfo 463
17.2.50 SetNotificationsOptions 464
17.2.51 SetParent 465
17.2.52 SetProgramOptions 466
17.2.53 SetSMTPConfig 467
17.2.54 SetSceneProfile 468
17.2.55 SetSystemDateTime 469
17.2.56 SetUserCredentials 470
17.2.57 SetVariable 471
17.2.58 StartNodesDiscovery 472
17.2.59 StopNodesDiscovery 473
17.2.60 Subscribe 474
17.2.61 SynchWithNTS 475
17.2.62 UDIService 476
17.2.63 Unsubscribe 477
17.2.64 WriteDeviceUpdates 478

17.3 WSDL ELK Core Web Services Documentation 479

17.3.1 ArmArea 479
17.3.2 AudioCmd 480
17.3.3 BypassArea 480
17.3.4 BypassZoneToggle 481
17.3.5 DisarmArea 481
17.3.6 DisplayTextOnAreaKeypads 482
17.3.7 GetAllStatus 482
17.3.8 GetAreaStatus 483
17.3.9 GetConfig 483
17.3.10 GetConnected 484
17.3.11 GetEnabled 484
17.3.12 GetKeypadStatus 485
17.3.13 GetOutputStatus 485
17.3.14 GetSystemStatus 486
17.3.15 GetThermostatStatus 486

17.3.16 GetTopology 487
17.3.17 GetZoneStatus 487
17.3.18 KeypadPressFuncKey 488
17.3.19 QueryAll 488
17.3.20 QueryAllZoneStatus 489
17.3.21 QueryAreaArmStatus 489
17.3.22 QueryKeypadTemperature 490
17.3.23 QueryOutputs 490
17.3.24 QueryThermostat 491
17.3.25 QueryZoneStatus 491
17.3.26 QueryZoneTemperature 492
17.3.27 QueryZoneVoltage 492
17.3.28 RefreshTopology 493
17.3.29 SetOutputOff 493
17.3.30 SetOutputOn 494
17.3.31 SpeakPhrase 494
17.3.32 SpeakWord 495
17.3.33 ThermostatCmd 495
17.3.34 TriggerZone 496
17.3.35 UnbypassArea 496

17.4 WSDL SEP/SmartGrid Core Web Services Documentation 497
17.4.1 SEPCancelAllDREvents 497
17.4.2 SEPCancelAllMessageEvents 498
17.4.3 SEPCancelAllPriceEvents 498
17.4.4 SEPConfirmMessage 499
17.4.5 SEPDROpt 499
17.4.6 SEPStartDREvent 500
17.4.7 SEPStartMessage 500
17.4.8 SEPStartPrice 501
17.4.9 SEPStopDREvent 501
17.4.10 SEPStopMessage 502
17.4.11 SEPStopPrice 502

17.5 WSDL Z-Wave Core Web Services Documentation 503
17.5.1 ExcludeDevice 503
17.5.2 FactoryResetDongle 504
17.5.3 IncludeDevice 504
17.5.4 ReplicateSendPrimary 505
17.5.5 SetActiveAntenna 505
17.5.6 StartLearnMode 506
17.5.7 StopIncludeExclude 506
17.5.8 Sync 507
17.5.9 SyncFull 507

18 Table of Figures 508

19 Bibliography 509

1 Disclaimer

The information in this book is for educational purposes only. Neither the author nor the publisher is responsible or liable for any direct, indirect, consequential, or incidental loss, damage, injury, or other issues that may result or arise from the use, misuse, or abuse of the information provided in this book.

Publication Date: 9/26/2018

2 Introduction

I have been interested in home automation for several years. I started my journey looking into the X10 line of products, and now have moved into the INSTEON line.

There are basic controllers available such as the INSTEON and X10 hubs. These hubs are simple devices that let you control devices such as light bulbs, wall switches, outlets, and thermostats.

However, they did not offer me the degree of control I desired to automate my home. I would like to control devices programmatically, as well as other devices which are not INSTEON, X10, or other similar device, such as serial I/O devices.

Universal Devices produces an extensive line of ISY Series home automation controllers, which have these capabilities.

There is a lot of good information available on working with the ISY994 series of devices. You can find this information online in the Universal Devices Wiki[1], the Universal Devices Forum[2], and in the Comprehensive User Guide[3]. There are also many program samples, created by talented programmers available.

[1] (Universal Devices)
[2] (Universal Devices)
[3] (Universal Devices)

These sources provide a lot of great information. However, I have found that sometimes it is very difficult to find the information needed. Dependent on the search terms you enter, and where you look, you may, or may not find the information required.

That is why I have created this manual. I have tried to consolidate a lot of the information from these sources, into a single source. I have tried to give credit for this work to the appropriate authors/sources. Please see the footnotes and the Bibliography. I have also incorporated a lot of additional information which I have learned in setting up my system.

While I do have this book copyrighted, the freely available content is not. This content is listed in the bibliography. What I do consider copyrighted to myself is any of my personal work, which I have included, as well as the organization, structure, and formatting of this document.

If you are interested in using the ISY and the built-in programming capabilities then refer to the **ISY994i Home Automation Cookbook**. However, if you desire to extend the capabilities of the ISY by creating logic outside of the ISY, specifically using Polyglot, etc., then this book is for you.

The instructions contained in this document should apply to firmware versions 5.x and above, in the ISY. When the instructions do not apply to all versions of the ISY, the minimum firmware version will be noted.

3 Developer's SDK Packages and Examples

Universal Devices offers a full Java SDK, Web Services (WSDK), and REST interface for developers who wish to create their own applications or integrate the ISY with an existing application or a portal. All of UDI's developer's documentation and libraries are completely free to use and distribute.

3.1 Official SDK [5.0.4]

Full Web Services/REST SDK, Java SDK, and Documentation with Open Source UDAjax

3.2 C# Examples

- ISY Test Example
- ISY Event Viewer Example
- Control/Event Viewer

3.3 Perl

- ISY Perl Library By evilpete
- ISY Utils By Jeffrey Honig

3.4 PHP

- Simple ISY PHP Library By MrWolf

3.5 Python

- ISY Python Library By evilpete
- PyISY: https://pypi.org/project/PyISY/
- Python 3 API Module: https://github.com/Einstein42/udi-polyglot-interface

3.6 Polyglot

- Polyglot source code is available at: https://github.com/UniversalDevicesInc/Polyglot

Since there might be WSDK changes and modifications in each firmware release, it's very important to keep up with firmware releases.

4 NodeServers

4.1 Creating a NodeServer[4]

Polyglot itself is a middleware application that integrates 3rd party devices directly into the ISY. It itself doesn't not speak to these 3rd party devices directly. That is accomplished by plugins otherwise known as NodeServers. Each NodeServer can do any number of things and have many individual nodes. For this example we are going to use the LiFX NodeServer as an example. In this guide we will walk through creating the LiFX NodeServer from the beginning of development all the way through making it available in the NodeServer store on Polyglot. It is important to note that with Polyglot version 2 the MQTT protocol is used as the IPC(inter-process communication). This allows for us to have NodeServers that are NOT local to Polyglot itself. While this is very powerful, most of you will still typically run the NodeServers locally to Polyglot. If you install a NodeServer via the Store, then it will be run locally.

4.1.1 Getting Started

Currently in the works is an API for several different languages, however currently the Python3 API is the only fully functional module. That being said, I will outline the IPC communication standards here that you are welcome to replicate in any language as needed.

Python 3.5+ was chosen specifically to mirror the async nature of this type of application. Python 3.5 comes with similar asynchronous co-routines that Polyglot itself has, and it also avoids the silly Python2 endless loops that can kill CPU's.

4.1.2 Create the NodeServer folder

Any folder under the ~/.polyglot/nodeservers/ folder is scanned for the files 'server.json' and 'profile.zip'. These two files are required for Polyglot to recognize it as a NodeServer.

server.json

The server.json is the description and data for your NodeServer. The important fields are: name, executable and optionally install.

[4] (James Milne)

```
{
   "name": "LiFX",**
   "docs": "https://www.lifx.com",
   "type": "python3",
   "executable": "lifx-poly.py",**
   "install": "install.sh",**
   "description": "Add LIFX control to the ISY994.",
   "notice": "\"LIFX\" is trademarked., see http://www.lifx.com for more information. This Node Server is neither developed by nor endorsed by LIFX.",
   "credits": [
      {
         "title": "polylifx: A Python library for LIFX",
         "author": "James Milne (Einstein.42)",
         "version": "0.1.6",
         "date": "November 8, 2017",
         "source": "https://github.com/Einstein42/lifx-nodeserver",
         "license": "https://github.com/Einstein42/lifx-nodeserver/blob/master/LICENSE"
      }
   ]
}
```

When you distribute your NodeServer make sure that the executable and install files are bash run-able (aka shebang'd) and that they are chmod 755 (or +x). This allows Polyglot to run them correctly. To test this just run your install script. ./install.sh should work.

profile.zip

The profile.zip file is the node definitions that ISY expects for any node added to it. Here is the documentation from UDI: **8 ISY Developer's Manual**. You must have the nls, editors, and nodedefs created using that format.

install script

The install script is located in the main directory and named in the server.json properties. This script should be +x or chmod 755 when distributed. Polyglot will automatically run this script after the NodeServer is installed via the Store. If the NodeServer is installed manually it must be run manually before being added to Polyglot. For the LiFX NodeServer example, we need to have several modules installed in Python for the NodeServer to function properly.

So our install.sh looks like this. I'm telling python 3's pip program to install all the modules in the requirements.txt file at the user level (not globally). Keep in mind this script does NOT run as root.

```
#!/usr/bin/env bash

pip3 install -r requirements.txt --user
```

The requirements.txt that the install script references is created by doing **pip freeze > requirements.txt** in your project folder directory. These should typically be developed in a virtual environment (venv) to make things nice and clean. These are the python requirements for the LiFX NodeServer.

```
bitstring==3.1.5

click==6.7

lifxlan>=1.1.5

paho-mqtt>=1.3.0

polyinterface>=1.0.5

python-dotenv>=0.6.4
```

executable

The executable property in the server.json is run via bash in a separate process directly via Polyglot (on local NodeServers) after the install process has completed and the profile.zip has been installed. The shebang (first line of a bash script) is what is called the interpreter for the program. In this case we are using Python 3. So the first line of the executable file 'lifx-poly.py' is

```
#!/usr/bin/env python3
```

This is one of the ways Polyglot allows for language agnosticism in it's NodeServers.

4.1.3 Writing the Code

Jump over to the polyinterface README (https://github.com/Einstein42/udi-polyglot-interface/blob/master/README.md) to see how to use the Python 3 API.

4.1.4 You built your NodeServer, now what?

Now you have your NodeServer built, everything works fine. Time to add it to Polyglot!

Open Polyglot:

Figure 1: Open Polyglot

Login to Polyglot:

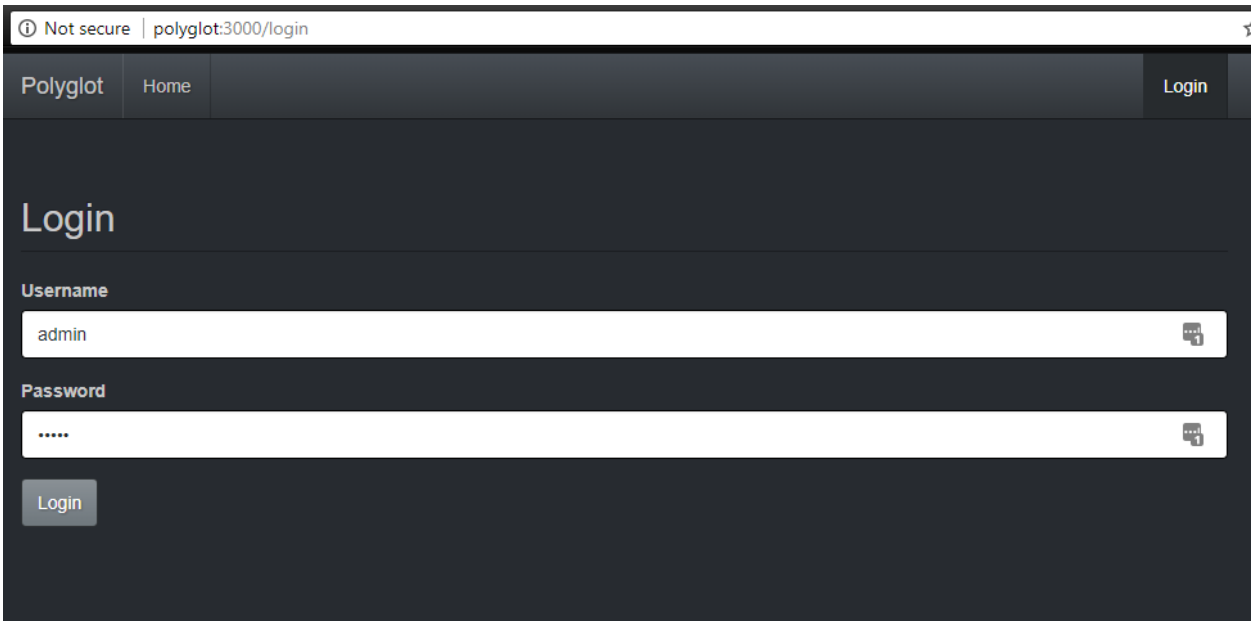


Figure 2: Login to Polyglot

Click on Add NodeServer. Choose type Local, and then you will see a drop-down list of all installed NodeServers found by Polyglot. This refreshes every time the page is loaded. Then choose a Profile Number to install it into. You will only see available Profile Numbers. If the ISY already has a slot filled, it will not be available.

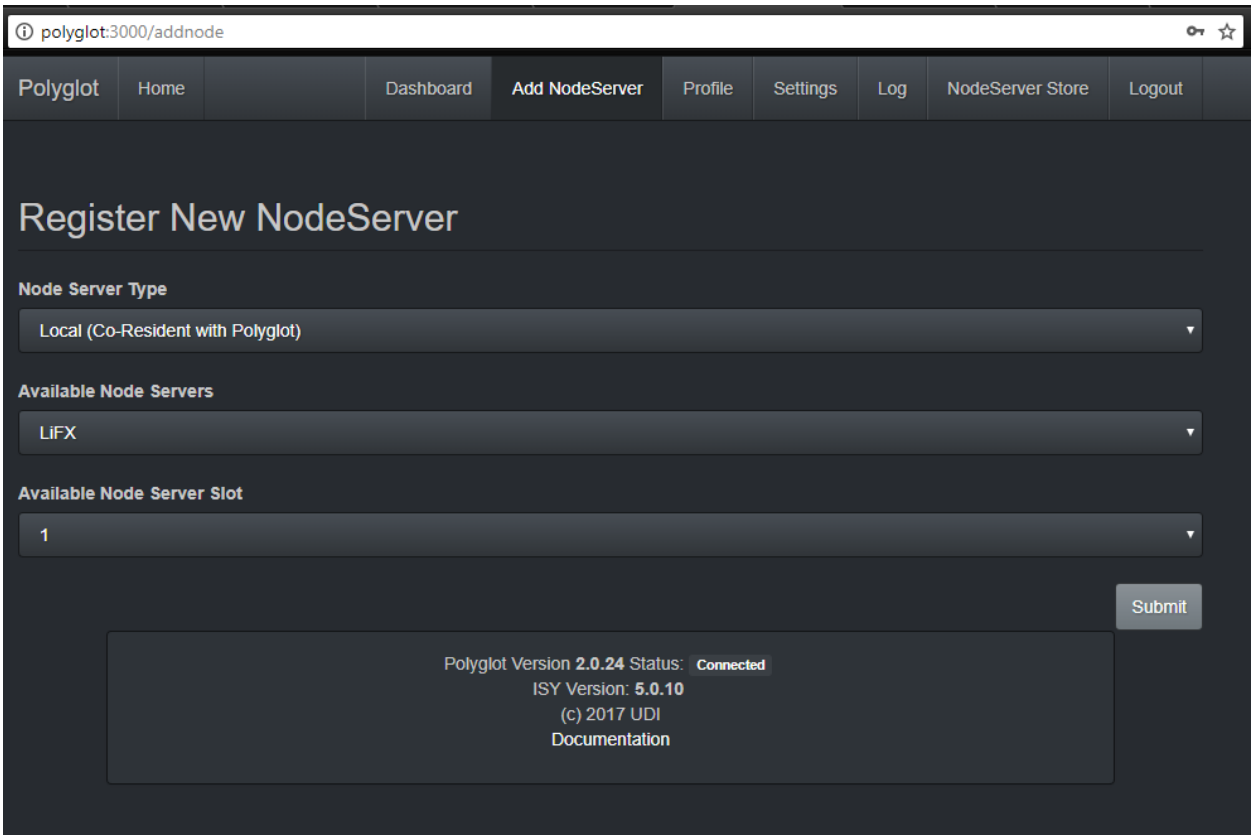


Figure 3: Register New NodeServer

Confirm that the ISY will reboot. This step automatically uploads the profile.zip from the NodeServer folder to the ISY and reboots it to take effect. No manual steps required.

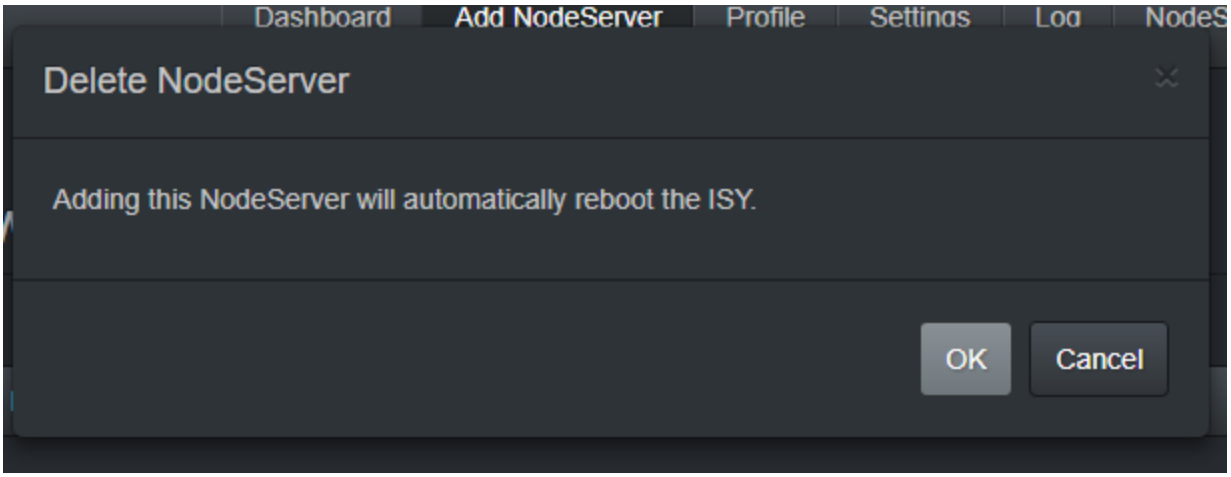


Figure 4: Delete NodeServer

Wait 60 seconds for the ISY to reboot. Then Polyglot automatically starts the NodeServer via the executable property in the server.json

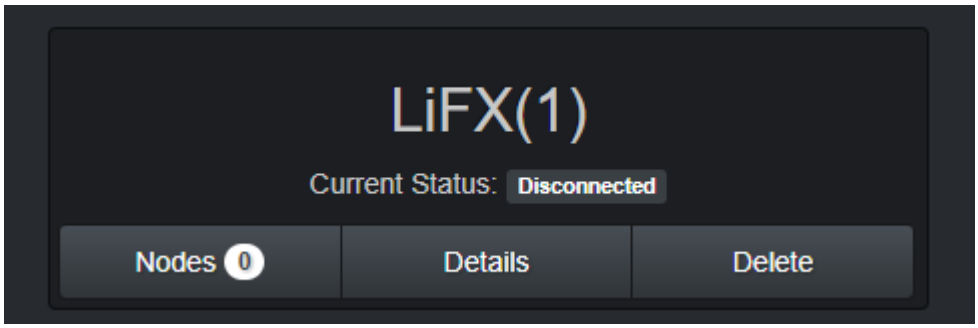


Figure 5: Delete NodeServer Disconnected

Watch the log file in real time via the Log button on the Navigation bar.

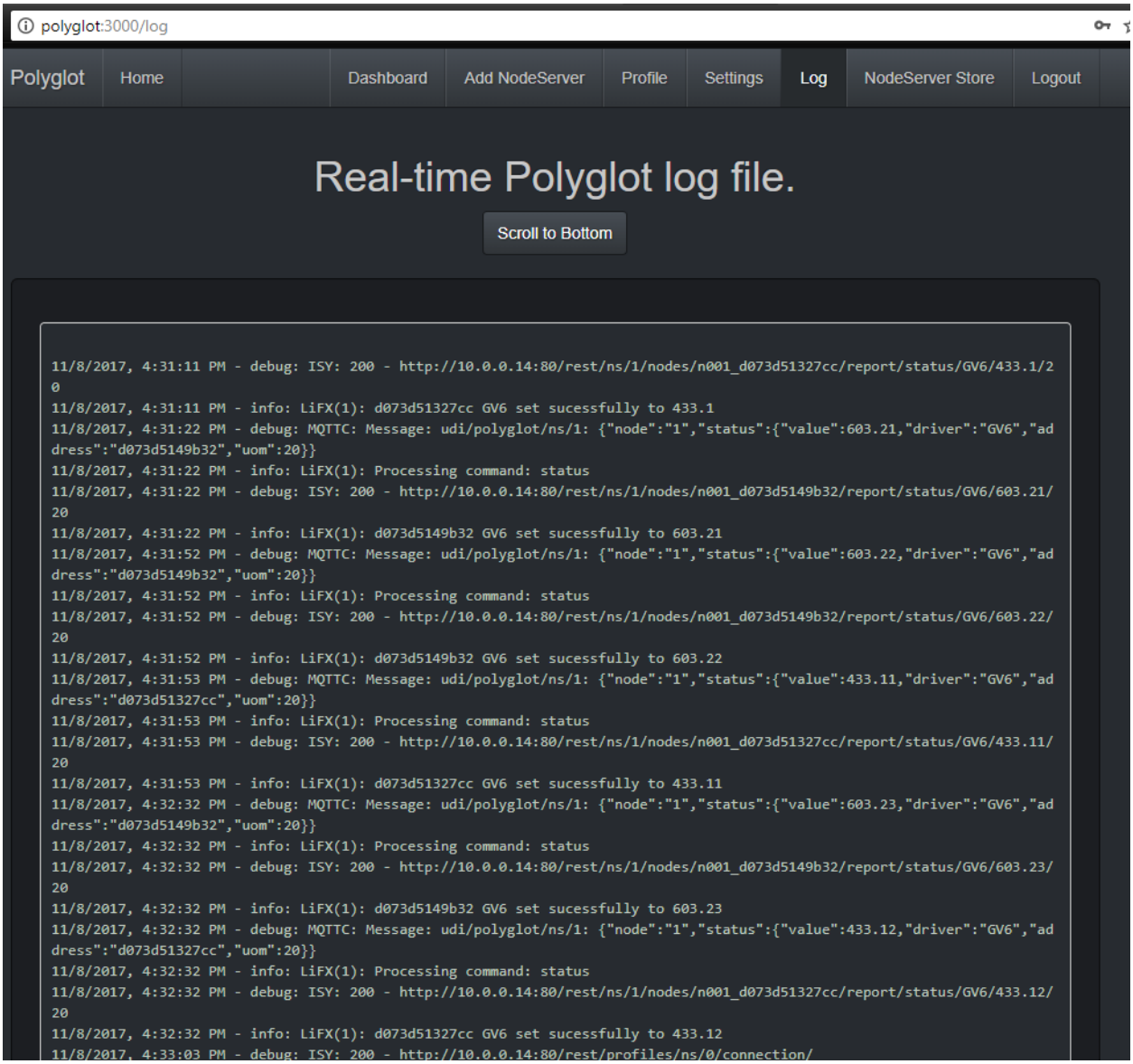


Figure 6: Polyglot Real-time Log File

Once the NodeServer starts. Check the Dashboard again. Notice it now says connected and shows how many sub-nodes it has.

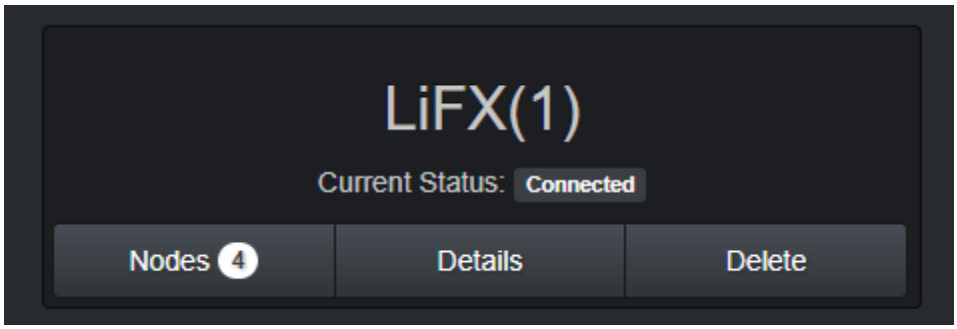


Figure 7: NodeServer Connected

Click on Details then Nodes and see real time information on the nodes themselves.

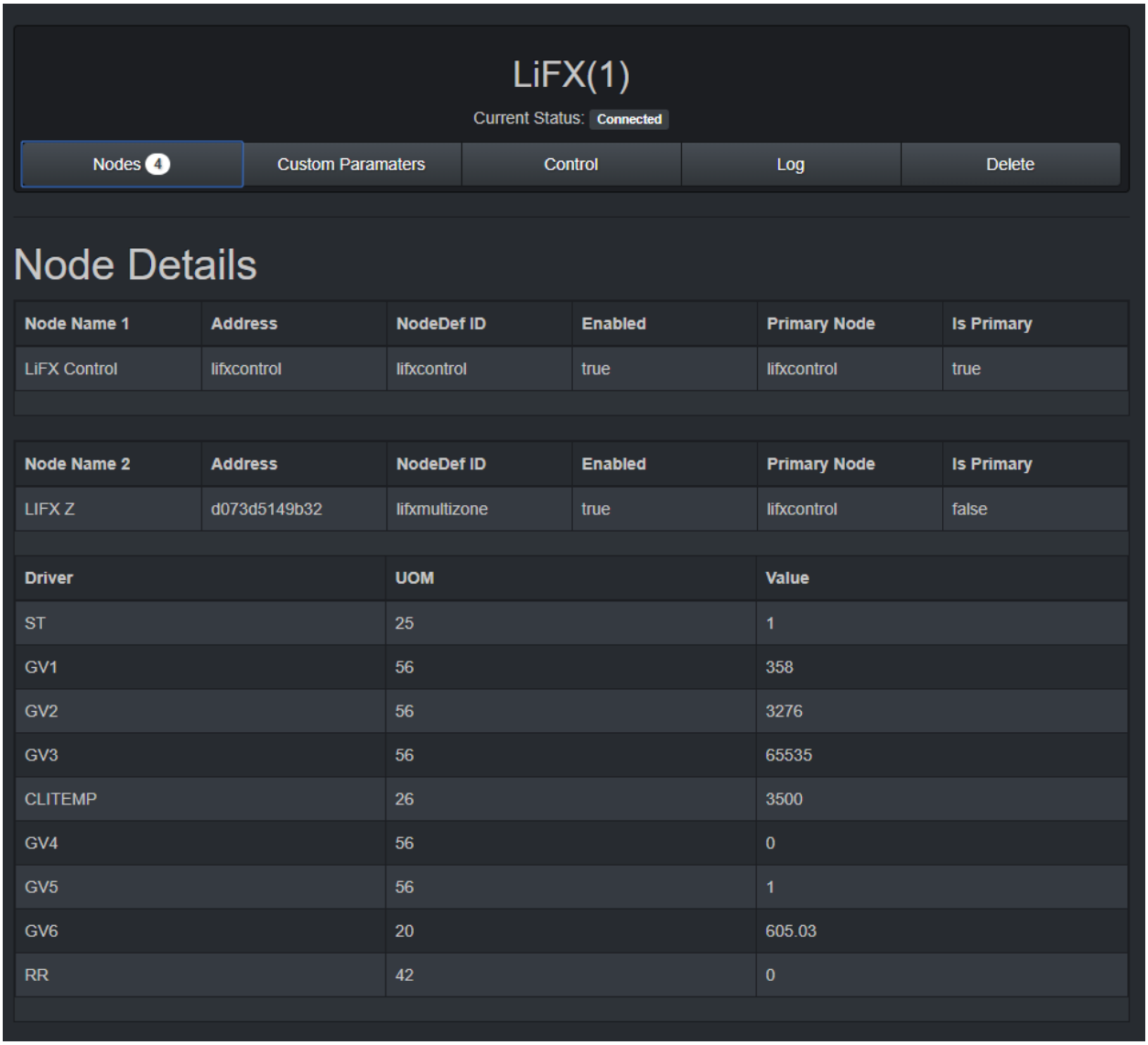


Figure 8: Node Details

Click on Log in the details page to see the real-time log for the NodeServer itself.

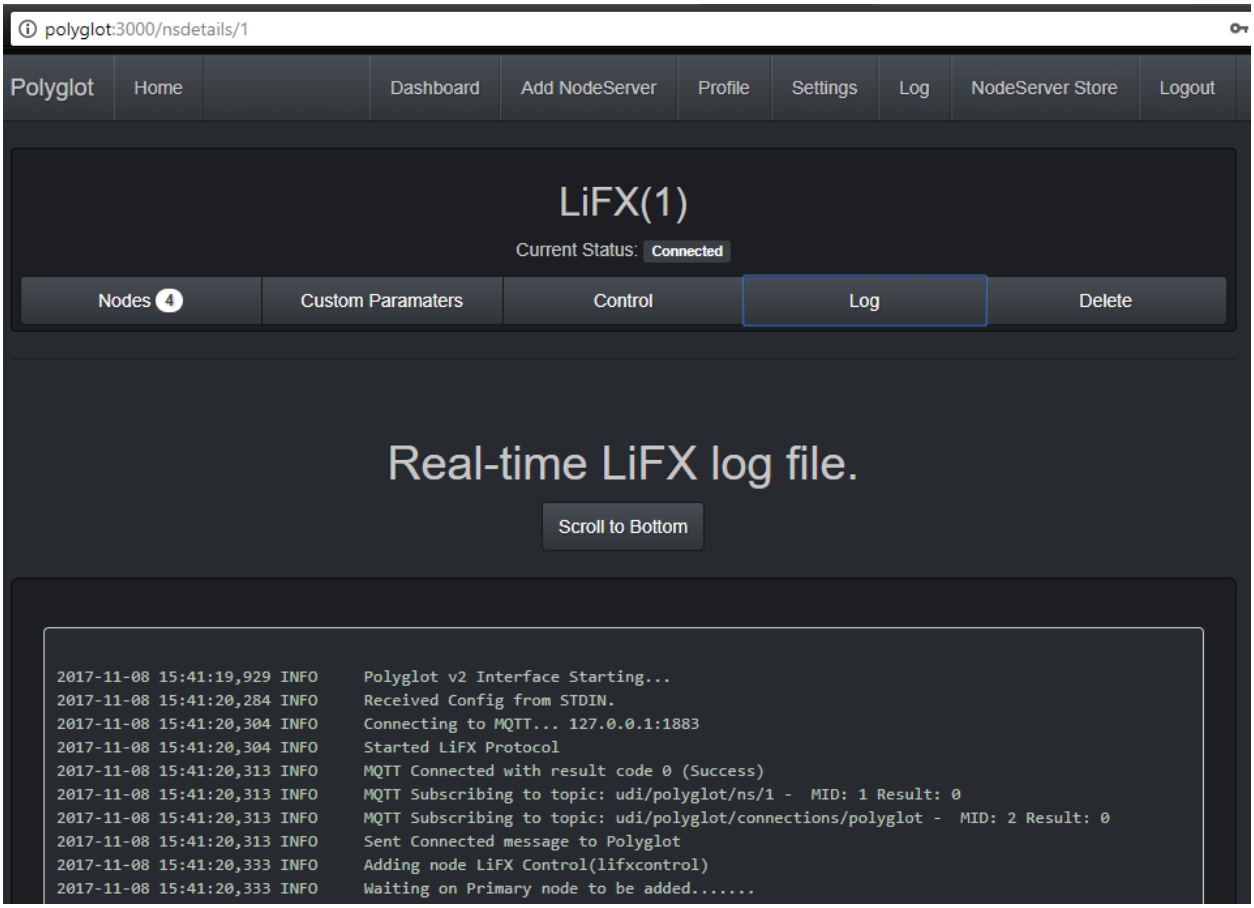


Figure 9: NodeServer Real-time Log File

Away you go. If you are troubleshooting, you can tail the log file in ~/.polyglot/nodeserver//logs/debug.log or watch it live in the Polyglot frontend. You can start, stop or restart, the NodeServer under Details > Control.

5 Polyglot

Polyglot is a Python library which seamlessly integrates with ISY releases 5.0.2+ and thus extend the capabilities of ISY for supporting other devices as nodes. The initial release includes support for Phillips Hue and Kodi.

5.1 Polyglot Quick Install Notes

Use these quick instructions to install Raspberry PI, Polyglot and some NodeServers. If you need additional information, see following sections.

RASBERRY PI INSTALL (note: ignore windows warnings about needing formatting or whatever)

- run etcher (www.etcher.io) and follow instructions pointing to .img file downloaded/unzipped from raspberrypi.org website (raspbian lite) and install image on a microsd card.
- create file called "ssh" in root directory, delete the file extension, just "ssh". This enables ssh session from putty/other telnet app.
- for wifi access create file called: **wpa_supplicant.conf**
- open in a text editor and enter the following to configure wifi into the wpa_supplicant file:

```
country=US

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev

update_config=1

network={

    ssid="MyWiFiNetwork"

    psk="aVeryStrongPassword"

    key_mgmt=WPA-PSK

}
```

- put sd card into pi and boot
- find pi ip address on network
- run ssh session in putty using above ip address and default port

- username: pi, password: raspberry
- change password by enering the following at prompt: **passwd**
- other config can be done by entering: **sudo raspi-config**

POLYGLOT INSTRUCTION:

Instructions and file located at https://github.com/UniversalDevicesInc/polyglot-v2

- run polyglot install script (note, to paste into putty, just right click) (also note, this may take 5 or more minutes):

  wget -qO - https://raw.githubusercontent.com/UniversalDevicesInc/polyglot-v2/master/scripts/install.sh | bash -e

- Enter the following to check if polyglot is running:

  sudo systemctl status polyglot-v2

- To open polyglot admin console enter the following into a web browser of computer on same network:

  https://rasberrypiipaddress:3000

- skip over warnings about security
- Defulat user/pass is admin/admin
- Under Settings tab, enter your ISY username/password
- Check that it now says it is connected at the bottom of page

TO INSTALL NODES:

- Go to Node Servers/Node Store tab and "install" the one(s) you want
- Go to Node Servers/Install Node Server to complete the installation
- Go to ISY admin console and the nodes should be there

5.2 Installing Raspberry PI[5]

The following instructions are based on an excellent video produced by James Milne. For more detailed instructions see **https://www.youtube.com/watch?v=w9I_bip6Vbo**

- The first thing that you will need to do is to purchase a Raspberry PI. You can find this on Amazon.com. You will need to get the Raspberry Pi 3 Model B+. You can buy this with an included SSD memory card. You can also purchase it without this card, but if you did that you would also need to get an SSD card.
- You will need to get the operating system for the Raspberry PI. You can find it on the **https://raspberry.pi.org/downloads/raspbian** web site. You can get both the desktop and the lite versions. For these instructions we will be using the lite version, as we will be running "headless". In other words, we will not need be plugging in a HDMI cable, mouse, or keyboard. We will be running with a "command line" and we won't be using the "desktop". These instructions are also based on a hard-wire network connection, not Wi-fi.

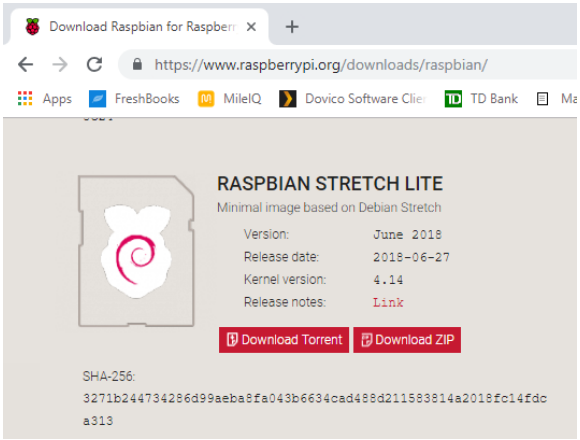


Figure 10: Raspian Stretch Lite Download

- Extract the zip file so that you have the **.img** file to work with.
- Plug the SSD card into your computer. You will see a prompt to format the card. Cancel these dialogs. Since this card will not be using a Windows operating system, it does not require formatting. It will be running a form of Linux. The file systems are not compatible.

[5] (James Milne)

- Download the **Etcher** software program to “flash” an image to the SSD card. You can find this file on the **https://etcher.io** web site. Select the **.img** file you extracted, select the SSD card, and then select “flash”

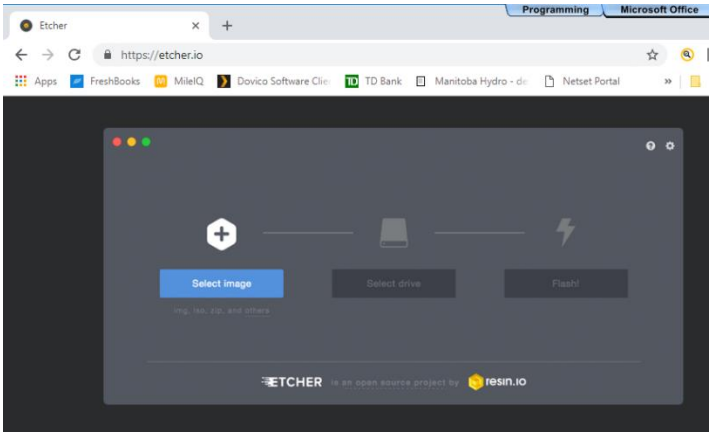


Figure 11: Etcher Program

- Cancel any “format disk” dialogs that may appear.
- You will be presented with the “boot” drive. If you do not see it use the file explorer, find it, and open it.
- You will need to create a “blank” file on this boot drive. Create a text file, call it **SSH**, and remove any extensions. It should not be called **ssh.txt**. Instead it should be called **ssh**. This enables ssh on the Raspberry PI.

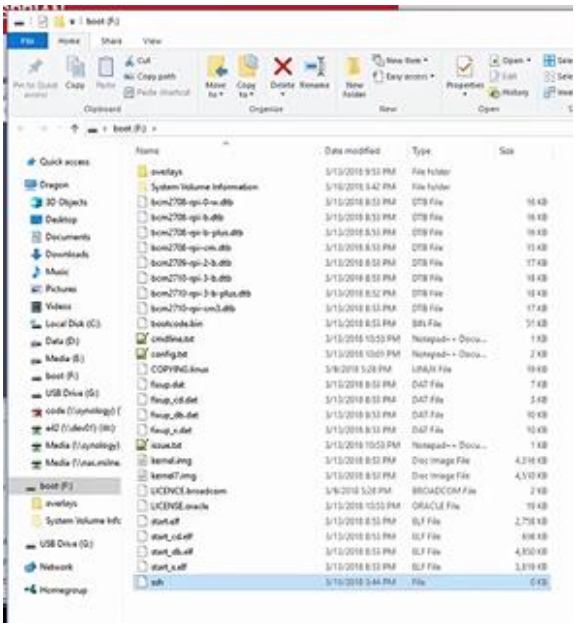


Figure 12: BOOT Drive with SSH File Added

- for wifi access create file called: **wpa_supplicant.conf**
- open in a text editor and enter the following to configure wifi into the wpa_supplicant file:

```
country=US

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev

update_config=1

network={

    ssid="MyWiFiNetwork"

    psk="aVeryStrongPassword"

    key_mgmt=WPA-PSK

}
```

- Eject the SSD card from your computer and put it into your Raspberry PI. You should have the ethernet connected, as well as having the SSD card inserted before connecting power to your Raspberry PI. Power up the Raspberry PI.
- You will need to find the IP address of the Raspberry PI. You can use the your router login to determine the IP address of the Raspberry PI, and then do an “address reservation” in order to keep the address from shifting.
- Connect to your Raspberry Pi using SSH. You can use several different ssh client programs to access using SSH. I will be using **SecureCRT** for these instructions. You can get this file from the **https://www.vandyke.com/prodducts/securecrt/** web site. Enter the host name, which is the IP address of the Raspberry PI. You will need to accept and save the “host” key. Enter the username and password, which initially for PI is: **username: pi**, and **password: raspberry**. Change the password using the **passwd** command.

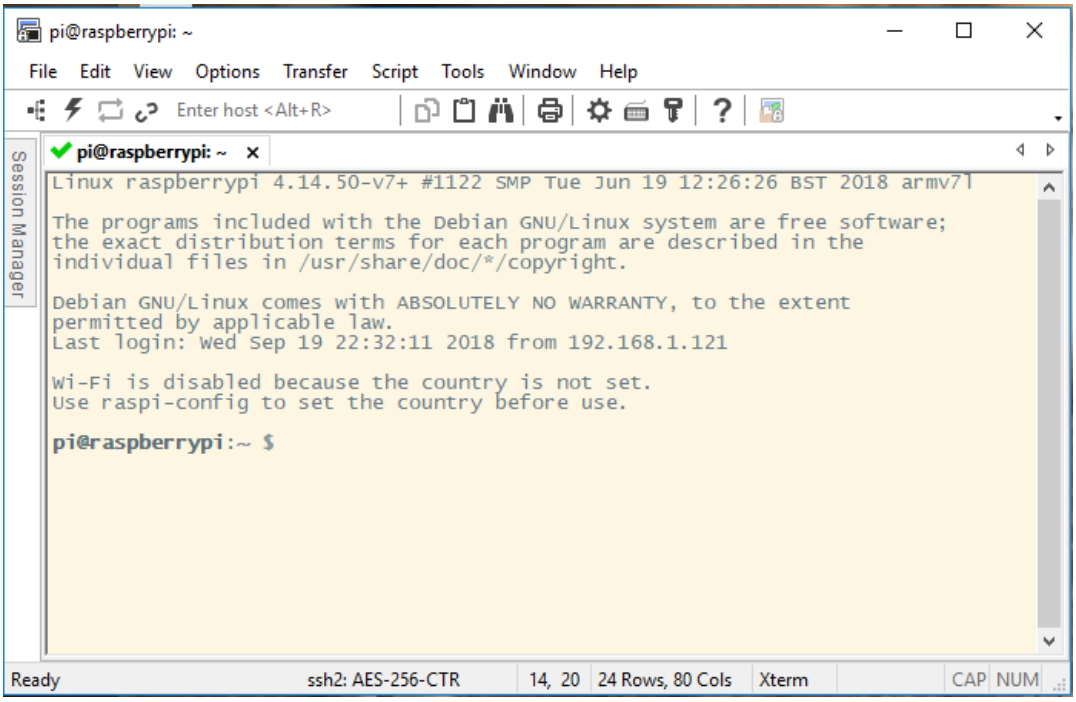


Figure 13: SecureCRT SSH Raspberry PI Session

5.3 Installing Polyglot[6]

The following instructions are based on an excellent video produced by James Milne. For more detailed instruction see **https://www.youtube.com/watch?v=lWqAq4Xf-9c&index=3&t=0s&list=PL418LHgc2F6uttVUW4cSDlpuX4BewQk8Y**

- To install Polyglot start by getting the script at: https://github.com/UniversalDevicesInc/polyglot-v2

```
wget -qO - https://raw.githubusercontent.com/UniversalDevicesInc/polyglot-v2/master/scripts/install.sh | bash -e
```

- Copy this script and paste into your SSH client. This will update the Raspberry PI with all the latest patches and files required, all the pre-requisite files.

[6] (James Milne)

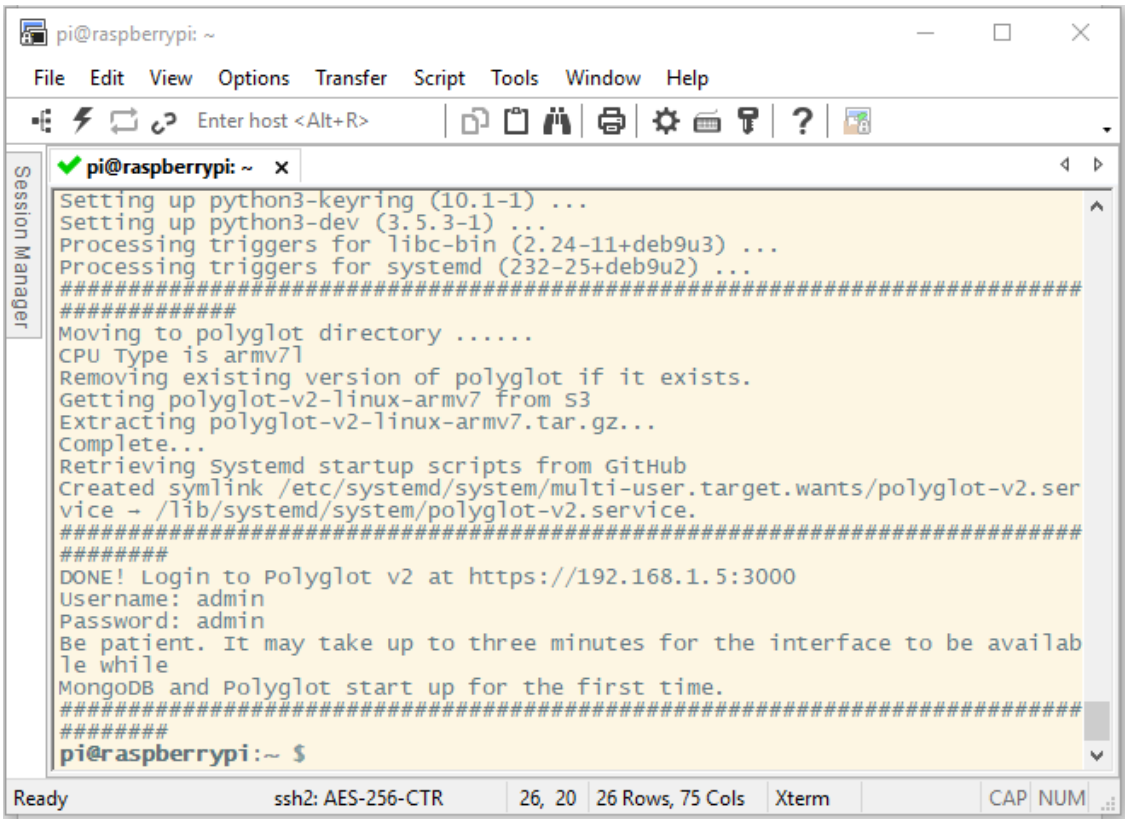


Figure 14: Polyglot Script Completion

- To verify that Polyglot was successfully installed type the following:

  sudo systemctl status polyglog-v2

  You will see the status that Polyglot is active and running, such as follows:

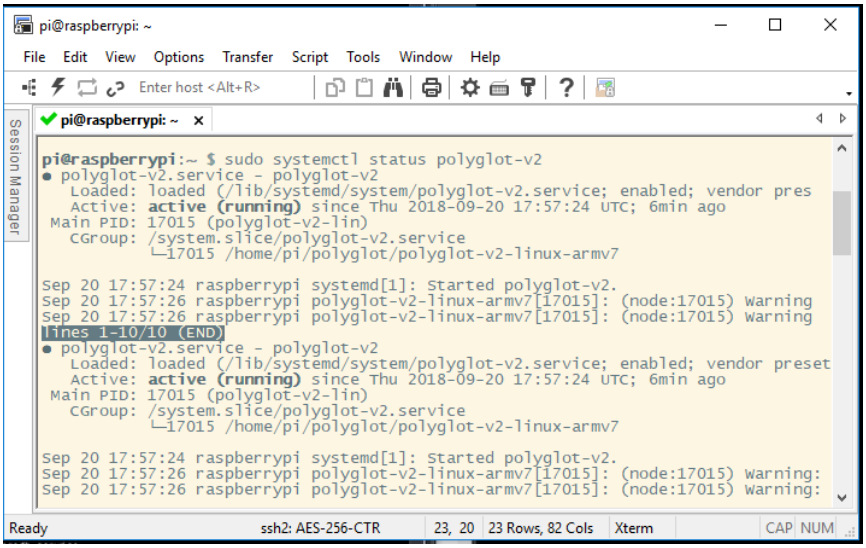


Figure 15: Polyglot Installation Status

- Copy the address Polyglot is using, such as **https://192.168.1.5:3000**
- Open your browser and paste this address into the URL bar. You can ignore any connection not private messages that come up.
- Go to the "ADVANCED" option on the screen
- Select "Proceed to … (unsafe) option.
- You should see the following display:

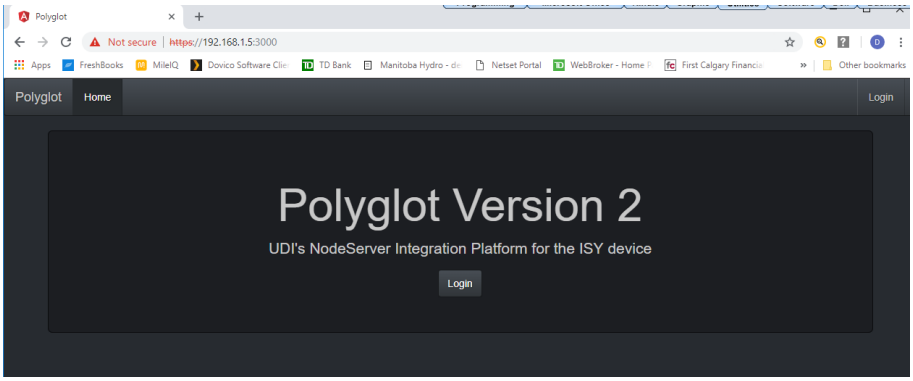


Figure 16: Polyglot Login Screen

5.4 Installing NodeServers[7]

The following instructions are based on an excellent video produced by James Milne. For more detailed instructions see https://www.youtube.com/watch?v=EnlqOl_46A&t=19s

- Login using username: **admin**, and password: **admin**. These are the initial values.
- You will notice that the ISY was automatically discovered. However, you will need to change the username and password to what you use in your ISY. Go to the **Settings** tab. Change these values and save them here. You will see the following screen:

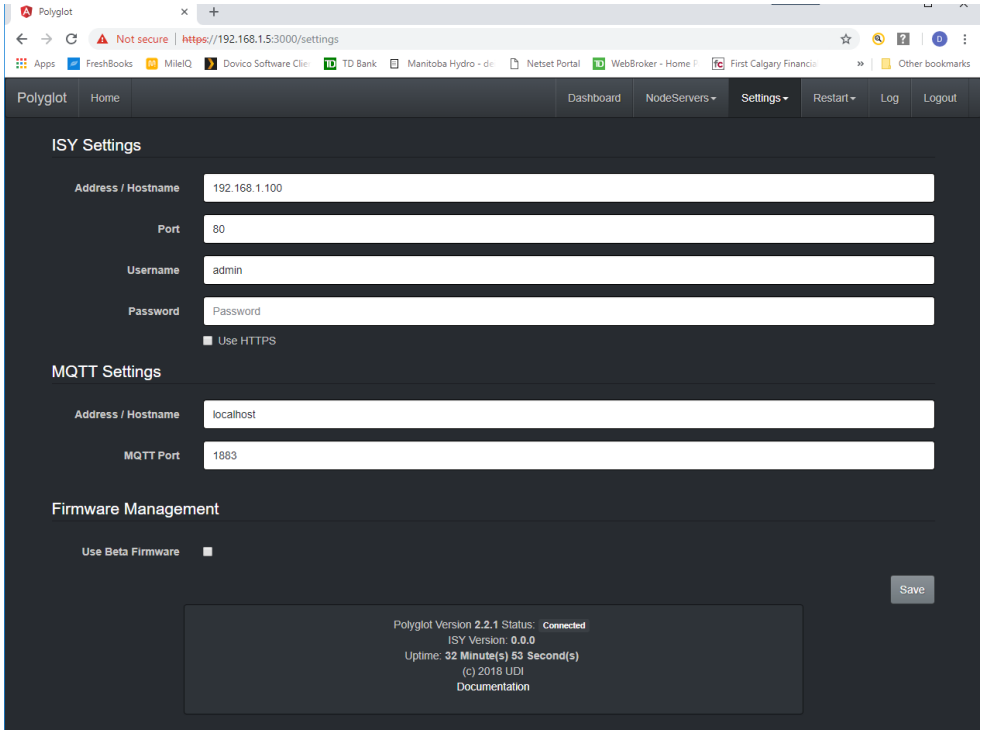


Figure 17: Polyglot Settings Status

- You will see on the bottom of the screen that the ISY was found, and the version number is shown.
- To change the password for Polyglot itself, to the **Settings->Profile** screen and change it t there.

[7] (James Milne)

- If you go to the **Dashboard** screen you will see that you currently do not have any NodeServers installed.

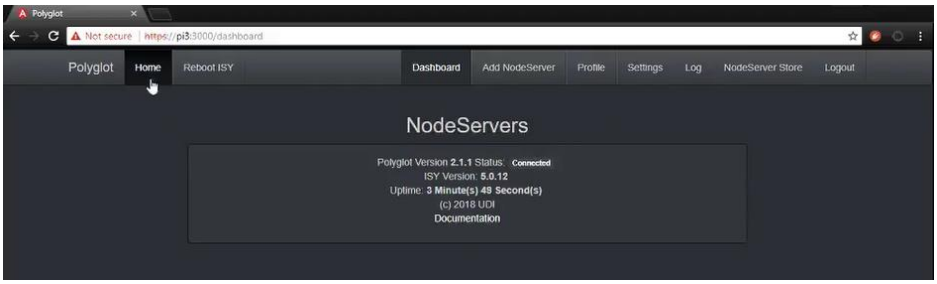


Figure 18: Dashboard Screen – No NodeServers

- Once every couple of minutes, Polyglot will communicate with the ISY to see if there are any unmanaged NodeServers installed there, and if so that information will be brought in, such as NodeLink and ISY Portal. For example this next screen shows that Polyglot found that the ISY Portal was on the ISY and has installed that. It also shows that the ISY Portal NodeServer was installed in slot number 1.

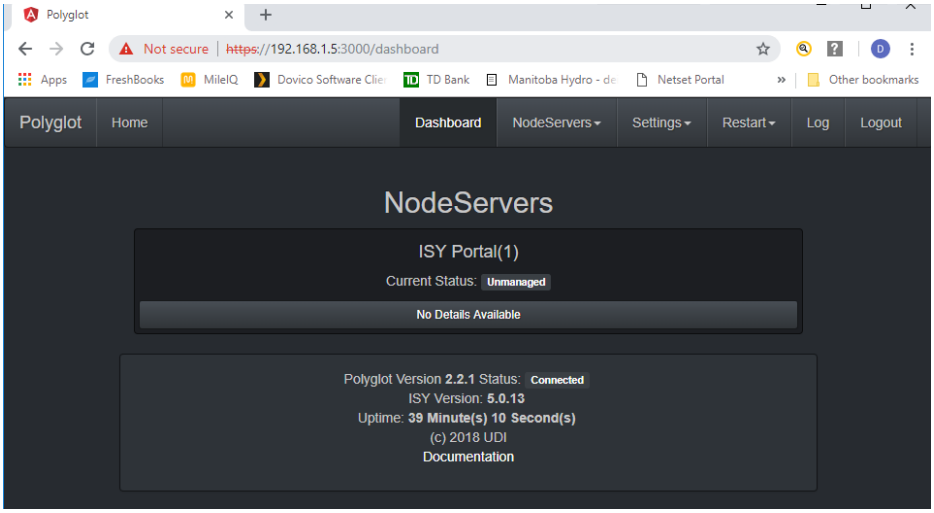


Figure 19: Dashboard Screen – With ISY Portal NodeServer

- To add a NodeServer go to the **NodeServer->NodeServer Store** screen. The example below shows that currently there are thirty-seven (37) NodeServers available for download. This number will change as developers create new ones.

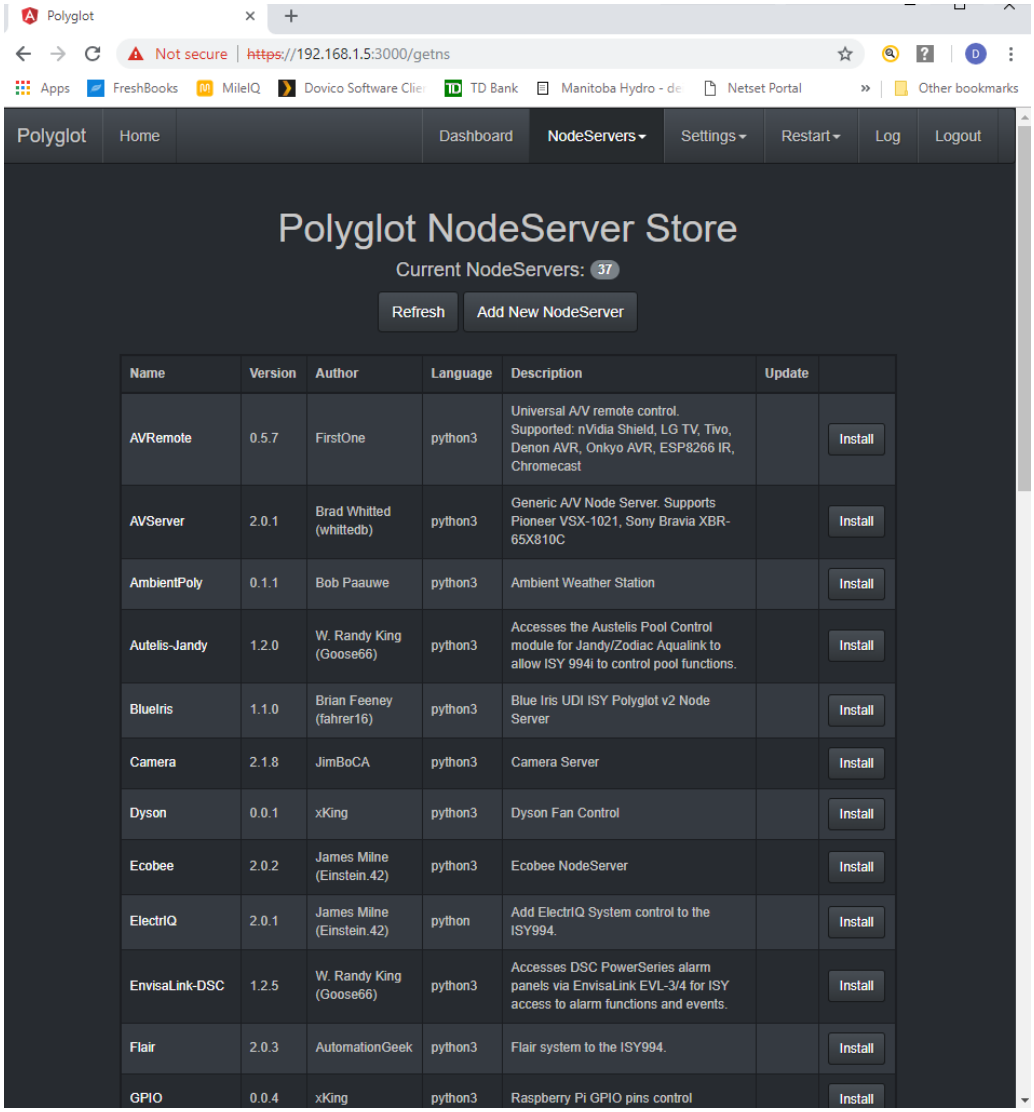


Figure 20: NodeServer Store Screen

- Select the desired NodeServer and press the **Install** button. This will clone the repository from github with the files you need.

- You will then need to add the NodeServer you just downloaded. Go to the **NodeServer->Add NodeServer** screen. You will see the NodeServer you downloaded in the **Available Node Servers** section of the screen, as well as the **Available Node Server Slot**.

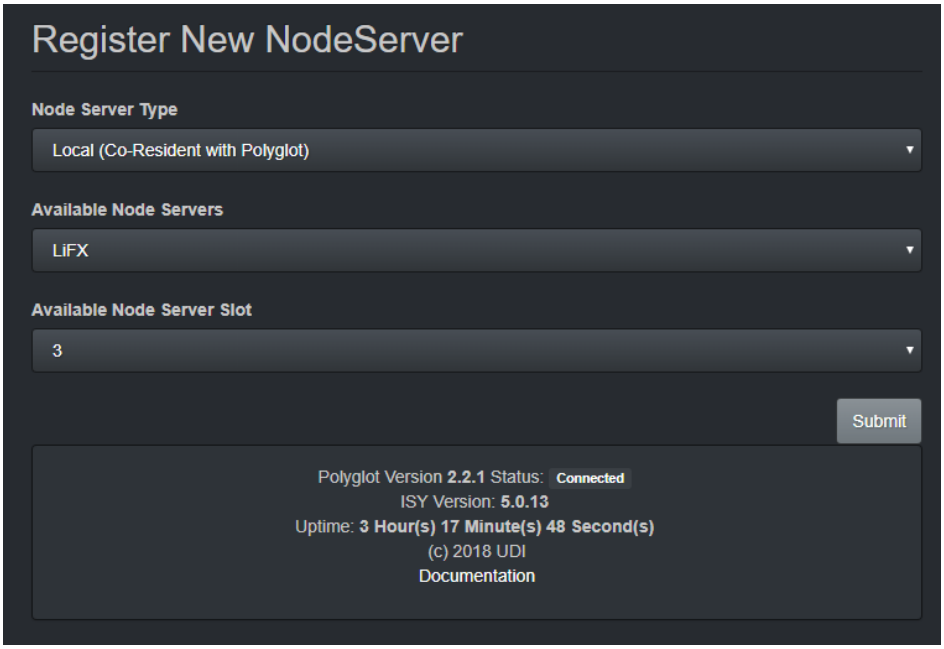


Figure 21: Register New NodeServer Screen

- Press the **Submit** button.

- If you check the **Dashboard** you will see the NodeServer.

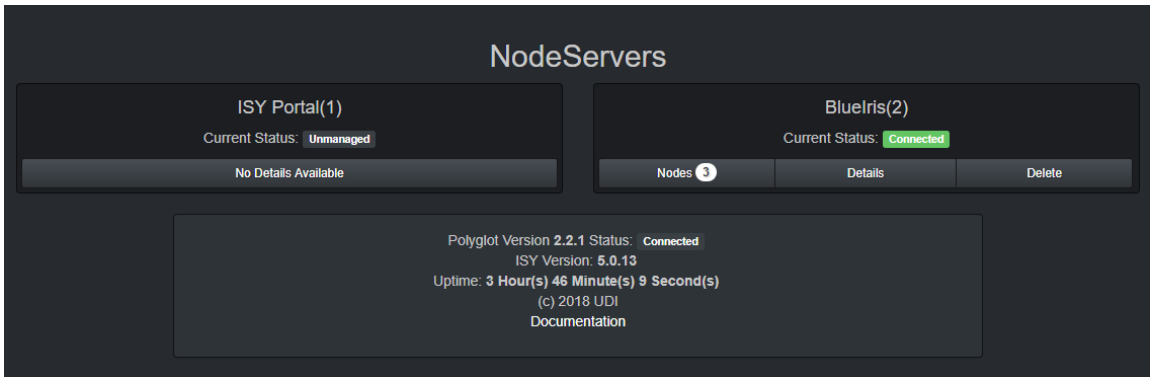


Figure 22: NodeServer Status

- If you check in the ISY you will see the NodeServer listed under the Network.

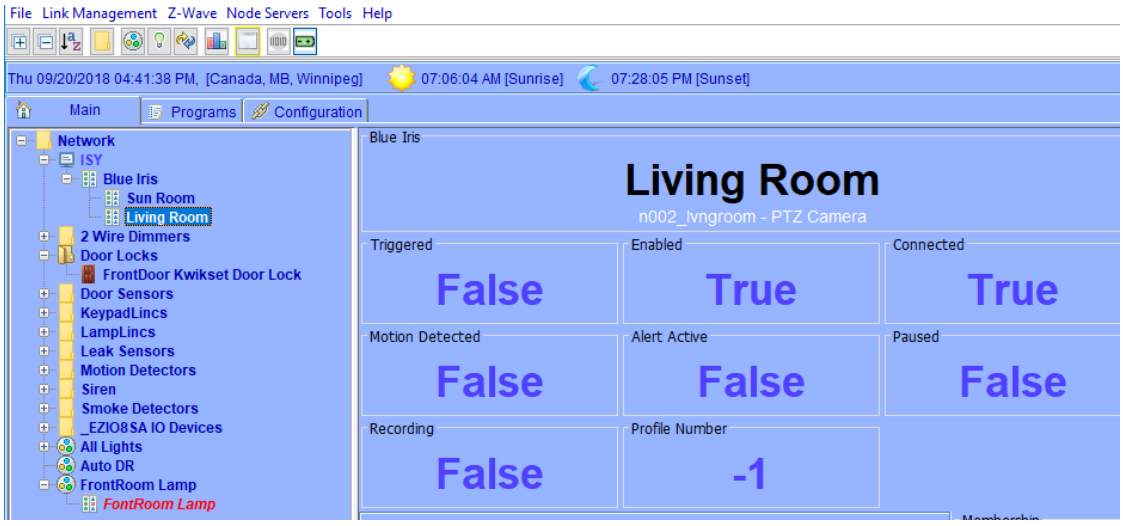
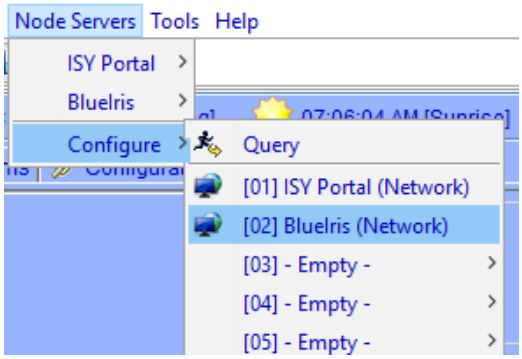


Figure 23: NodeServer Shown in ISY Network Tree

- If you check the **Node Servers** menu will see the NodeServer listed there.



5.5 Polyglot Virtual Node Server Framework[8]

Polyglot Virtual Node Server Framework is an application that makes it easy and quick to both develop and maintain virtual node servers for the ISY-994i home automation controller by Universal Devices Inc. Using virtual node servers, the ISY-994i is able to communicate with and control third-party devices to which the ISY-994i cannot natively connect.

Polyglot is written primarily with Python 2.7 and makes it easy to develop new Virtual Node Servers with Python 2.7. It should, however, by noted, that Virtual Node Servers may by developed using any language. Polyglot is intended to be run on a Raspberry Pi 2 Model B, but could potentially run on any ARM based machine running Linux with Python 2.7. FreeBSD, OSX, and x64 linux binaries are provided as well.

5.5.1 Usage

5.5.1.1Installation

Polyglot is open source software provided under the MIT license. There are two ways to install Polyglot. Either a binary compiled version or the pure python (from source) version.

Version 0.0.6 Released November 30th, 2016

These methods are for linux debian x86 or raspbian on a rpi.

Get the system pre-requisites:

```
apt-get install python-git python-pip python3-pip libjpeg-dev
```

5.5.1.1.1Pure Python (aka non-compiled source)

Clone the repository to a directory under the user you wish to run Polyglot. This will not run as root. eg. /home/pi/ the following line will create a directory called Polyglot so no need to do that.

```
git clone https://github.com/UniversalDevicesInc/Polyglot.git
```

Move yourself into the Polyglot directory

```

```

[8] (Universal Devices)

```
cd Polyglot
```

Install the Python required modules this does require root as we want to install them globally for Python to access.

```
sudo pip install -r requirements.txt
```

Run Polyglot!

```
python -m polyglot -v
```

This will launch Polyglot and create a directory titled config in the current directory. Polyglot will store all of its configuration and its log inside of this directory. You may specify a manual path for this directory using the command line flags.

5.5.1.1.2COMPILED version (aka all-in-one)

We will need to create a home for Polyglot

```
mkdir Polyglot && cd $_
```

We still need the python pre-requisites:

```
sudo pip install -r https://github.com/UniversalDevicesInc/Polyglot/raw/ unstable-release/requirements.txt
```

Download the polyglot binary for your system. One of these:

For ARM (Raspberry Pi's)
https://github.com/UniversalDevicesInc/Polyglot/raw/unstable-release/bin/polyglot.linux.armv7l.pyz

For x86 Linux flavors (Built with Debian sid):
https://github.com/UniversalDevicesInc/Polyglot/raw/unstable-release/bin/polyglot.linux.x86_64.pyz

For MAC (Built on Yosemite)
https://github.com/UniversalDevicesInc/Polyglot/raw/unstable-release/bin/polyglot.osx.x86_64.pyz

Make the file executable. Use the filename you downloaded. Example is the ARM version.

```
chmod 755 polyglot.linux.arm7l.pyz
```

Run Polyglot!

```
./polyglot.linux.arm7l.pyz -v
```

5.5.1.1.3Command line flags

```
-h, --help          show this help message and exit

-c CONFIG_DIR, --config CONFIG_DIR

                    Polyglot configuration directory

-v, --verbose       Enable verbose logging

-vv                 Enable very verbose logging
```

While running in its default mode, Polyglot will log all warnings and errors. Verbose logging will include info messages. Very verbose mode adds debug messages that could be useful when developing a new node server.

5.5.1.1.4OSX Instructions

Install XCODE Developer Tools (enables git) The easiest way to do this is to go to the console and type:

```
git
```

This will automatically launch the XCODE installer.

Once XCODE is installed run:

```
sudo easy_install pip
```

This installs pip 8.1.1 and now we are ready to get our binary or clone the github repository as instructed above.

5.5.1.1.5Start Polyglot on Boot

If you are running the module you already have the polyglot.service file in your Polyglot root folder. If not then get it like so:

```
wget https://github.com/UniversalDevicesInc/Polyglot/raw/unstable-release/polyglot.service
```

Edit the file polyglot.service with your favorite editor. Modify WorkingDirectory to be your root Polyglot directory. eg. /home/pi/Polyglot

```
WorkingDirectory=/home/pi/Polyglot
```

Modify ExecStart to be how you start it. Full path needed. For pure Python(Non-compiled):

```
ExecStart=/usr/bin/python -m polyglot -v
```

For the compiled binary:

```
ExecStart=/home/pi/Polyglot/polyglot.linux-arm7l.pyz -v
```

Change the user to the user account that will run polyglot (NOT ROOT)

```
User=pi
```

Copy polyglot.service to /lib/systemd/system/ You need sudo as /lib/systemd/system is a system directory.

```
sudo cp /home/pi/Polyglot/polyglot.service /lib/systemd/system/
```

Enable systemctl (Make sure polyglot isn't already running):

```
sudo systemctl enable polyglot

sudo systemctl start polyglot
```

5.5.1.1.6Logging locations

Log file is found at config/polyglot.log. To watch the live action:

```
tail -fn 50 /home/pi/Polyglot/config/polyglot.log
```

5.5.1.2User Interface

Once Polyglot is running, the user interface may be accessed by opening your favorite browser and navigating to:

```
http://localhost:8080
```

The default username and password are both ***admin***.

If you are accessing the frontend from another machine, replace localhost with the IP Address or URL of the machine running Polyglot. If you are having trouble accessing the user interFace from a remote machine, check your firewall settings.

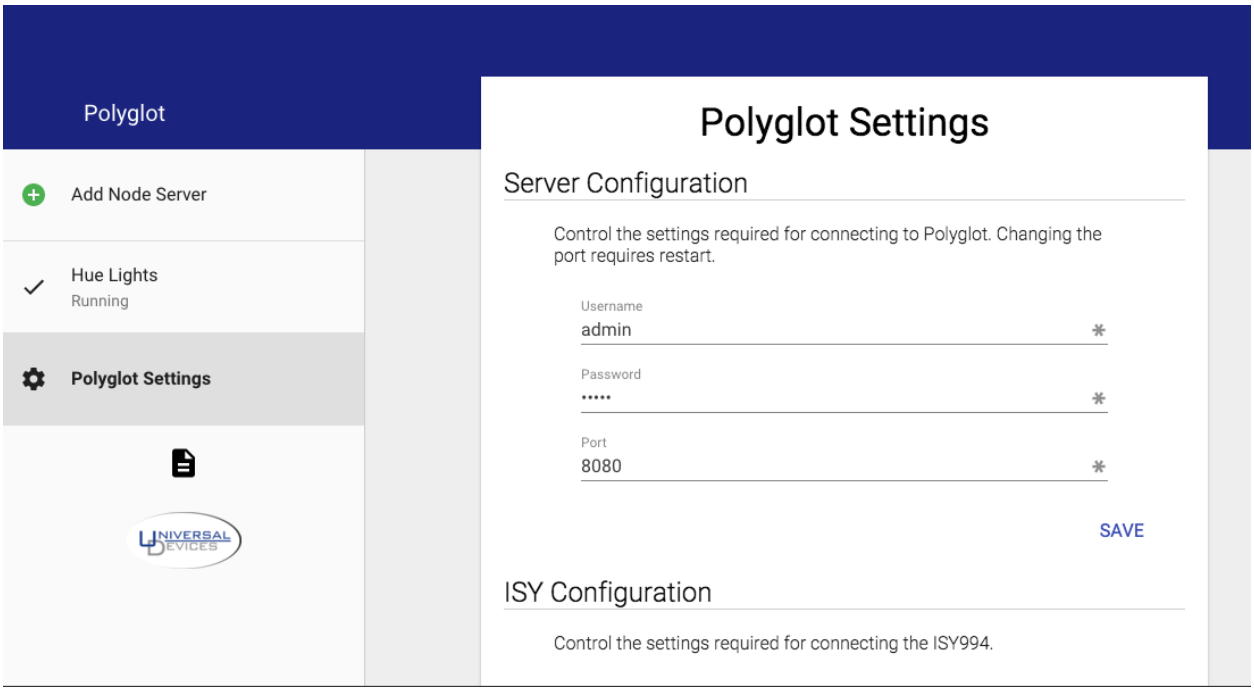


Figure 24: Polyglot Settings

The user interface is designed to be simple and intuitive to use. Pictured above is the settings page. Using the menu bar on the left, new node servers can be added and existing node servers may be monitored. The button on the bottom of the menu will open Polyglot's log in a new browser window.

The user interface is fully compatible with both tablet and mobile devices.

5.5.1.2.1Settings

The settings view allows the user to alter settings for Polyglot's HTTP server as well as Polyglot's connection to the ISY controller. It is recommended that the username and password are changed from the default. If a new different port is desired, it may be set in the Server Configuration block.

It is also necessary to set the username, password, host name, and port required for connecting to the ISY. These may be configured in the ISY Configuration block.

5.5.1.2.2Adding Node Server

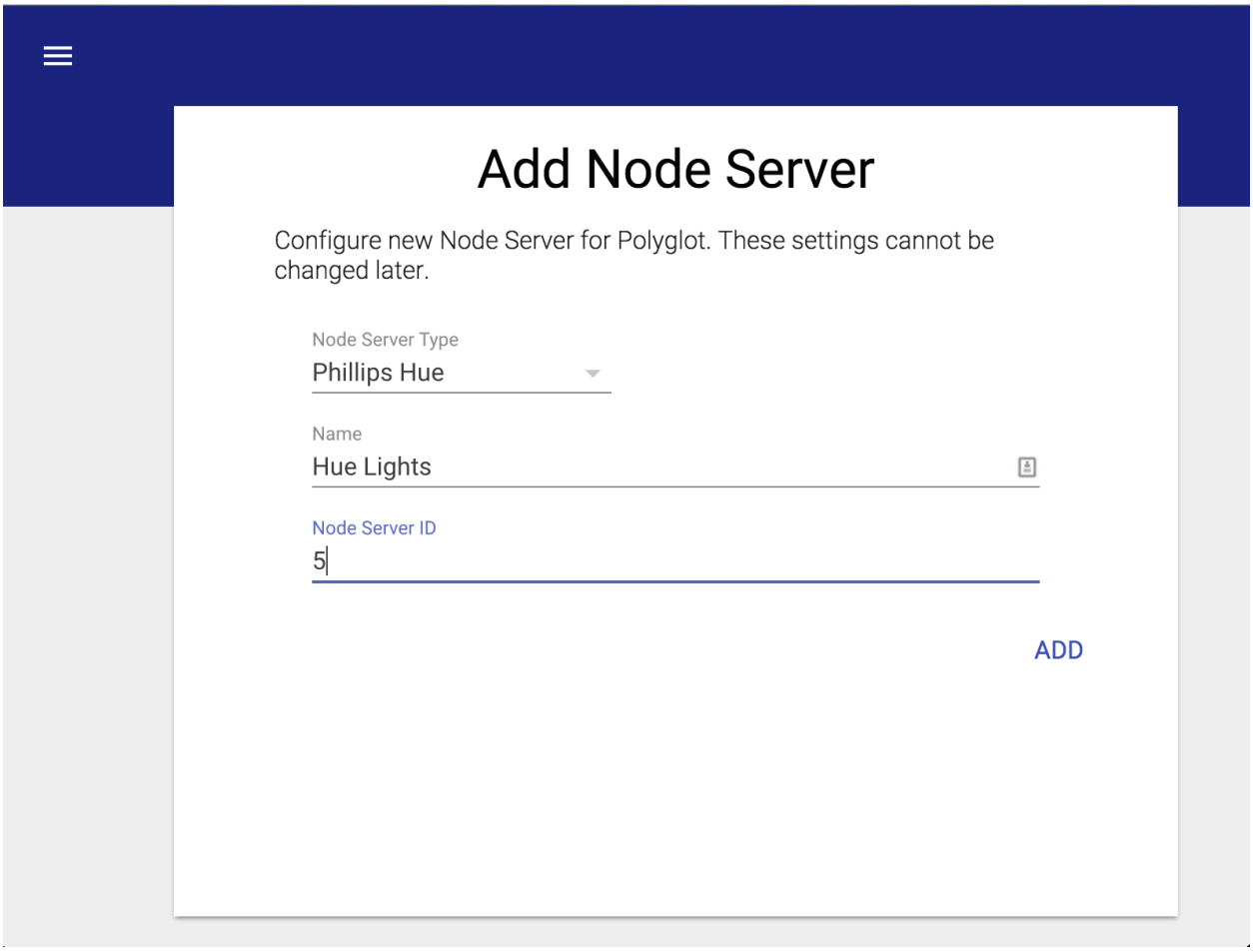


Figure 25: Add Node Server

To add a node server, navigate to the Add Node Server view using the menu. This view is pictured above.

Populate this form with the details for the new node server. Select a type from all installed types using the drop down. Give the node server any name allows for easy recognition. Finally, populate the Node Server ID field with an ID that is available in the ISY. Press ADD when complete.

The node server will now be available in Polyglot. You may navigate to it using the menu. The node server view in Polyglot will show the Node Server ID, Base URL, and allow for the Profile to be downloaded.

In order to access the node server from the ISY, it must be added to the ISY. To do this, inside of the ISY console, navigate to Node Servers then Configure then the Node Server ID that was set while creating the node server. This will open a dialog that accepts all the information from the node server view. Populate this with the Profile Name and Base URL from the node server view. The User ID, Password, Host Name, and Port here must be the values used for connecting to Polyglot. Timeout may be left as 0, and the Isy User should be set to the appropriate user ID that was configured in Polyglot. If you are unsure, use 0.

Click the Upload Profile button and navigate to the zip file obtained from Polyglot's node server view. Once this has been uploaded, click Ok and restart the ISY controller. Once the ISY has fully rebooted, restart the node server in Polyglot using the node server view.

5.5.1.2.3Managing Node Servers

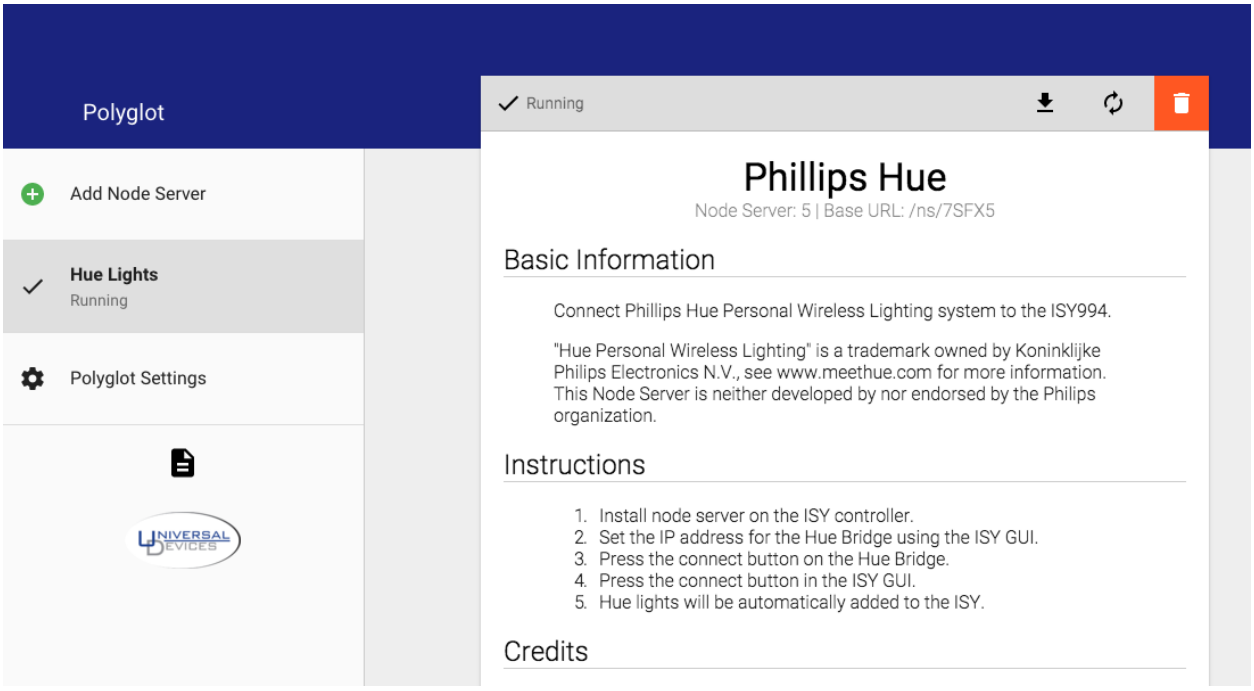


Figure 26: Managing Node Servers – Phillips Hue

Clicking a Node Server in the menu will activate the node server view. In this view, there is a menu bar at the top. This menu bar will indicate is the node server is Running or Stopped. It also provides buttons to download the profile, restart the node server, or delete the node server.

Also, in this view are instructions for using this node server. Different node servers may have their own instructions on how to use them in the ISY. Any open-source, third party libraries that were used for the development of the node server are also credited here.

If the node server were to crash, a red X will appear next to it in the menu and it will be indicated in the menu bar on the top of the node server view. If this happens, it is best to save the log for debugging and then restart the node server using the button in the menu bar.

5.5.1.2.4Viewing Polyglot Log

There is a file icon below all the main menu items. Clicking this icon will open Polyglot's log in a new browser window. This log file is critical for debugging issues with Polyglot.

5.5.2 Node Server Development

5.5.2.1Background

Node servers in Polyglot are nothing more than stand-alone processes that are managed by Polyglot. Polyglot communicates with the node servers by reading the STDOUT and STDERR streams as well as writing to the STDIN stream. STDIN and STDOUT messages are JSON formatted commands that are documented in **5.5.4 Polyglot Node Server API.**

As of Polyglot 0.0.6 MQTT is available as a communication mechanism as well. See **5.5.2.9MQTT.**

5.5.2.2File Structure

Node servers are defined in self-contained folders. The name given to this folder will be the node server ID and must be unique form all other node servers. New node servers can be stored in Polyglot's configuration directory in the folder titled node servers. Inside of this folder, at least the following three files must exist.

- profile.zip is the profile that must be uploaded to the ISY describing the node server. This file is documented in the ISY Node Server API documentation.

- instructions.txt should be a file containing instructions on the use of the node server documented using markdown. The contents of this file will be formatted and displayed on the frontend.
- server.json is the metadata used by Polyglot to identify the node server. This file is documented in the next section.

The rest of the node server's folder should contain the code required to execute the node server and all necessary libraries with the exception of those explicitly included as part of the Polyglot distribution.

Node servers are executed in special directories in the user's configuration directory. Each node server type is assigned its own directory. Any required information may be written to this directory. Keep in mind, that all running node servers of the same type will share the same directory.

5.5.2.3Server Metadata

The server.json file in the node server source directory is a JSON formatted file that informs Polyglot of how the node server is executed as well as other important details about the node server. The file contains a dictionary formatted object with specific fields. A sample server.json is included below. It has been extracted from the Philips Hue node server.

```
{

  "name": "Phillips Hue",

  "docs": "https://www.universal-devices.com/",

  "type": "python",

  "executable": "hue.py",

      "configfile": "config.yaml",

      "interface": "Default",

  "description": "Connect Phillips Hue Personal Wireless Lighting system to the ISY994.",

  "notice": "\"Hue Personal Wireless Lighting\" is a trademark owned by Koninklijke Philips Electronics N.V., see www.meethue.com for more information. This Node Server is neither developed by nor endorsed by the Philips organization.",

  "credits": [
```

```
    {
        "title": "phue: A Python library for Philips Hue",
        "author": "Nathanaël Lécaudé (studioimaginaire)",
        "version": "0.9",
        "date": "May 18, 2015",
        "source":
"https://github.com/studioimaginaire/phue/tree/c48845992b476f4b1de9549ea5b52773
99f56581",
        "license":
"https://raw.githubusercontent.com/studioimaginaire/phue/c48845992b476f4b1de9549
ea5b5277399f56581/LICENSE"
    }
  ]
}
```

Below is a description of the required fields:

- name is the name of the node server type as it will be displayed to the user.
- docs is a link to an appropriate website about the node server. This value is not currently displayed anywhere.
- type is the node server executable type. This instructs Polyglot as to how the node server should be launched. Currently, only python is accepted.
- executable is the file that Polyglot should execute to start the node server process.
- description is a short description of the node server that will be displayed to the user on the frontend.
- notice contains any important notices the user might need to know.
- credits is a list of dictionaries indicating all third party library used in the node server. Some open source projects require that they be credited some where in the project. Others do not. Either way, it is nice to give credit here. When including a third party library in your node server, ensure that it is licensed for commercial use.

In the credits list:

- title is the title of the third party library.
- author is the author of the third party library.
- version is the appropriate versioning tag used to identify the third party library.
- date is the date the third party library was either released or obtained.
- source is a link to the library's source code.
- license is a link to the library's license file. Ensure that this is a static link whose contents cannot be changed. Linking to a specific GitHub commit is handy for this.

It can be a good idea to check the formatting of this file with a JSON linter before attempting to load the node server in Polyglot. If this file cannot be read, for whatever reason, the node server will not appear in the Polyglot frontend and an error will be logged.

5.5.2.4Python Development

A Python 2.7 compatible implementation of the API is provided with Polyglot to assist in Node Server development. It may be easily imported as shown below. In the future, more libraries may be made available and more languages may be supported.

```
from polyglot import nodeserver_api
```

The provided Node Server Library exposes all of the ISY controller's Node Server RESTful API as is. Data received by Polyglot's web server is parsed and directed immediately to the node server process via this library. The library will also send messages back up to Polyglot to be transmitted directly to the ISY. The only exception to this rule is that node ID's will not have the node server ID prefix prepended to them. It will also be expected that the node server will not prepend these prefixes. Polyglot will handle the node ID prefixes on behalf of the node servers.

There also exists, in the Python library, some abstract classes that may be used to ease the development of a new node server. Except in rare cases where it may not be appropriate, it is recommended that these be used.

When Python is used to develop node server, the Polyglot environment is loaded into the Python path. This environment includes the Requests library.

5.5.2.5Python Polyglot Library

5.5.2.5.1Summary

This library consists of four classes and one function to assist with node server development. The classes ***polyglot.nodeserver_api.NodeServer*** and ***polyglot.nodeserver_api.SimpleNodeServer*** and basic structures for creating a node server. The class ***polyglot.nodeserver_api.Node*** is used as an abstract class to crate custom nodes for node servers. The class ***polyglot.nodeserver_api.PolyglotConnector*** is a bottom level implementation of the API used to communicate between Polyglot and your node server. Finally, included in this library is a method decorator, ***polyglot.nodeserver_api.auto_request_report()***, that wraps functions and methods to automatically handle report requests from the ISY.

5.5.2.5.2Custom Node Types

When creating a new node server, each node type that will be controlled by the server must be defined. This abstract class may be used as a skeleton for each node type. When inheriting this class, a new method should be defined for each command that the node can perform. Additionally, the _drivers and _commands attributes should be overwritten to define the drivers and commands relevant to the node.

class polyglot.nodeserver_api.Node(parent, address, name, primary=True, manifest=None)[source]

Abstract class for representing a node in a node server.

Parameters:

- parent (polyglot.nodeserver_api.NodeServer) – The node server that controls the node
- address (str) – The address of the node in the ISY without the node server ID prefix
- name (str) – The name of the node
- primary (polyglot.nodeserver_api.Node or True if this node is the primary.) – The primary node for the device this node belongs to, or True if it's the primary.
- manifest (dict or None) – The node manifest saved by the node server

_drivers = {}

The drivers controlled by this node. This is a dictionary of lists. The key's are the driver names as defined by the ISY documentation. Each list contains at least three values: the initial value, the UOM identifier, and a function that will properly format the value before assignment. The fourth value is optional – if provided, and set to "False", the driver's value will not be recorded in the manifest (this is useful to reduce I/O when there is no benefit to restoring the driver's value on restart).

Insteon Dimmer Example:

```
_drivers = {
   'ST': [0, 51, int],
   'OL': [100, 51, int],
   'RR': [0, 32, int]
}
```

Pulse Time Example:

```
_drivers = {
   'ST': [0, 56, int, False],
}
```

_commands = {}

A dictionary of the commands that the node can perform. The keys of this dictionary are the names of the command. The values are functions that must be defined in the node object that perform the necessary actions and return a boolean indicating the success or failure of the command.

*_report_driver_cb(driver, status_code, **kwargs)[source]*

Private method - updates ISY synchronization flag based on the success/fail of the status update API call to the ISY.

add_node()[source]

Adds node to the ISY

Returns boolean:

Indicates success or failure of node addition

get_driver(driver=None)[source]

Gets a driver's value

Parameters: driver (str or None) – The driver to return the value for

Returns: The current value of the driver

manifest

The node's manifest entry. Indicates the current value of each of the drivers. This is called by the node server to create the full manifest.

Type: dict

node_def_id = ''

The node's definition ID defined in the node server's profile

query()[source]

Abstractly queries the node. This method should generally be overwritten in development.

Returns boolean:

Indicates success or failure of node query

report_driver(driver=None)[source]

Reports a driver's current value to ISY

Parameters: driver (str or None) – The name of the driver to report. If None, all drivers are reported.

Returns boolean:

Indicates success or failure to report driver value

*report_isycmd(isycommand, value=None, uom=None, timeout=None, **kwargs)[source]*

Sends a single command from the node server to the ISY, optionally passing in a value.

No formatting, and little validation, of the isy cmd, value, or uom is done by this simple low-level API. It is up to the caller to ensure correctness.

Parameters:

- isycommand (str) – Name of the ISY command to send (e.g. 'DON')
- value (string, float, int, or None) – (optional) The value to be sent for the command
- uom (int or None) – (optional) - The value's unit of measurement. If provided, overrides the uom defined for this command
- timeout (int or None) – (optional) - the number of seconds before this command expires and is discarded

Returns boolean:

Indicates success or failure to queue for sending (Note: does NOT indicate if actually delivered)

*run_cmd(command, **kwargs)[source]*

Runs one of the node's commands.

Parameters:

- command (str) – The name of the command
- kwargs (dict) – The parameters specified by the ISY in the incoming request. See the ISY Node Server documentation for more information.

Returns boolean:

Indicates success or failure of command

set_driver(driver, value, uom=None, report=True)[source]

Updates the value of one of the node's drivers. This will pass the given value through the driver's formatter before assignment.

Parameters:

- driver (str) – The name of the driver
- value – The new value for the driver
- uom (int or None) – The given values unit of measurement. This should correspond to the UOM IDs used by the ISY. Refer to the ISY documentation for more information.
- report (boolean) – Indicates if the value change should be reported to the ISY. If False, the value is changed silently.

Returns boolean:

Indicates success or failure to set new value

smsg(strng)[source]

Logs/sends a diagnostic/debug, informative, or error message. Individual node servers can override this method if they desire to redirect or filter these messages.

5.5.2.6Polyglot API Implimentation

This class implements the Polyglot API and calls registered functions when the API is invoked. This class is a singleton and will not allow itself to be initiated more than once. This class binds itself to the STDIN stream to accept commands from Polyglot.

To create a connection in your node server to Polyglot, use something similar to the following. This creates the connection, connect to Polyglot, and then waits for the Node Server's configuration to be received. The configuration will be the first command received from Polyglot and will never be sent again after the first transmission.:

```
poly = PolyglotConnector()

poly.connect()

poly.wait_for_config()
```

Then, commands can be sent upstream to Polyglot or to the ISY by using the connector's methods.:

```
poly.send_error('This is an error message. It will be in Polyglot\'s log.')

poly.add_node('node_id_1', 'NODE_DEFINITION', 'node_id_0', 'New Node')

poly.report_status('node_id_1', 'ST', value=55, uom=51)

poly.remove_node('node_id_1')
```

To respond to commands received from Polyglot and the ISY, handlers must be registered for events. The handler's arguments will be the parameters specified in the API for that event. This will look something like the following.:

```
def status_handler(node_address, request_id=None):

   print('Status Event Handler Called')

poly.listen('status', status_handler)
```

Now, when the ISY requests a status update from Polyglot, this function will be called. Handlers will not be called in the node server's main thread.

class polyglot.nodeserver_api.PolyglotConnector[source]

Polyglot API implementation. Connects to Polyglot and handles node server IO.

Raises:RuntimeError

add_node(node_address, node_def_id, primary, name, timeout=None, seq=None)[source]

Adds a node to the ISY. To make this node the primary, set primary to the same value as node_address.

Parameters:

- node_address (str) – The full address of the node (e.g. ‘dimmer_1’)
- node_def_id (str) – The id of the node definition to use for this node
- primary (str) – The primary node for the device this node belongs to
- name (str) – The name of the node
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

change_node(node_address, node_def_id, timeout=None, seq=None)[source]

Changes the node definition to use for an existing node. An example of this is may be to change a thermostat node from Fahrenheit to Celsius.

Parameters:

- node_address (str) – The full address of the node (e.g. ‘dimmer_1’)
- node_def_id (str) – The id of the node definition to use for this node
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

commands = ['config', 'install', 'query', 'status', 'add_all', 'added', 'removed', 'renamed', 'enabled', 'disabled', 'cmd', 'ping', 'exit', 'params', 'result', 'statistics']

Commands that may be invoked by Polyglot

connect()[source]

Connects to Polyglot if not currently connected

connected

Indicates if the object is connected to Polyglot. Can be set to control connection with Polyglot.

Type: boolean

disconnect()[source]

Disconnects from Polyglot. Blocks the thread until IO stream is clear

*exit(*args, **kwargs)[source]*

Tells Polyglot that this Node Server is done.

*get_params(**kwargs)[source]*

Get the params from nodeserver and makes them available to the nodeserver api

*install(*args, **kwargs)[source]*

Abstract method to install the node server in the ISY. This has not been implemented yet and running it will raise an error.

Raises:NotImplementedError

listen(event, handler)[source]

Register an event handler. Returns True on success. Event names are defined in commands. Handlers must be callable.

Parameters:

- event (str) – Then event name to listen for.
- handler (callable) – The callable event handler.

logger = None

logger is initialized after the node server wait_for_config completes by the setup_log method and the log file is located in the node servers sandbox. Once wait_for_config is complete, you can call poly.logger.info('This variable is set to %s', variable)

*pong(*args, **kwargs)[source]*

Sends pong reply to Polyglot's ping request. This verifies that the communication between the Node Server and Polyglot is functioning.

remove_node(node_address, timeout=None, seq=None)[source]

Removes a node from the ISY. A node cannot be removed if it is the primary node for at least one other node.

Parameters:

- node_address (str) – The full address of the node (e.g. 'dimmer_1')
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

*report_command(node_address, command, value=None, uom=None, timeout=None, seq=None, **kwargs)[source]*

Sends a command to the ISY that may be used in programs and/or scenes. A common use of this is a physical switch that somebody turns on or off. Each time the switch is used, a command should be reported to the ISY. These are used for scenes and control conditions in ISY programs.

Parameters:

- node_address (str) – The full address of the node (e.g. 'switch_1)
- command (str) – The command to perform (e.g. 'DON', 'CLISPH', etc.)
- value (str, int, or float) – Optional unnamed value the command used
- uom (int or str) – Optional units of measurement of value
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired
- <pN>.<uomN> (optional) – Nth Parameter name (e.g. 'level') . Unit of measure of the Nth parameter (e.g. 'seconds', 'uom58')

report_request_status(request_id, success, timeout=None, seq=None)[source]

When the ISY sends a request to the node server, the request may contain a 'requestId' field. This indicates to the node server that when the request is completed, it must send a fail or success report for that request. This allows the ISY to in effect, have the node server synchronously perform tasks. This message must be sent after all other messages related to the task have been sent.

For example, if the ISY sends a request to query a node, all the results of the query must be sent to the ISY before a fail/success report is sent.

Parameters:

- request_id (str) – The request ID the ISY supplied on a request to the node server.
- success (bool) – Indicates if the request was sucessful
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

report_status(node_address, driver_control, value, uom, timeout=None, seq=None)[source]

Updates the ISY with the current value of a driver control (e.g. the current temperature, light level, etc.)

Parameters:

- node_address (str) – The full address of the node (e.g. 'dimmer_1')
- driver_control (str) – The name of the status value (e.g. 'ST', 'CLIHUM', etc.)
- value (str, float, or int) – The numeric status value (e.g. '80.5')
- uom (int or str) – Unit of measure of the status value
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

*request_stats(*args, **kwargs)[source]*

Sends a command to Polyglot to request a statistics message.

restcall(api, timeout=None, seq=None)[source]

Calls a RESTful api on the ISY. The api is the portion of the url after the “https://isy/rest/” prefix.

Parameters:

- api (str) – The url for the api to call
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

send_config(config_data)[source]

Update the configuration in Polyglot.

Parameters: config_data (dict) – Dictionary of updated configuration

Raises:ValueError

send_error(err_str)[source]

Enqueue an error to be sent back to Polyglot.

Parameters: err_str (str) – Error text to be sent to Polyglot log

smsg(str)[source]

Logs/sends a diagnostic/debug, informative, or error message. Individual node servers can override this method if they desire to redirect or filter these messages.

uptime

The number of sections the connection with Polyglot has been alive

Type: float

wait_for_config()[source]

Blocks the thread until the configuration is received

write_nodeserver_config(default_flow_style=False, indent=4)[source]

Writes any changes to the nodeserver custom configuration file. self.nodeserver_config should be a dictionary. Refrain from calling this in a poll of any kind. Typically you won't even have to write this unless you are storing custom data you want retrieved on the next run. Saved automatically during normal Polyglot shutdown. Returns True for success, False for failure.

Parameters:

- default_flow_style (boolean) – YAML's default flow style formatting. Default False
- indent (int) – Override the default indent spaces for YAML. Default 4

5.5.2.7Node Server Classes

class polyglot.nodeserver_api.NodeServer(poly, shortpoll=1, longpoll=30)[source]

It is generally desirable to not be required to bind to each event. For this reason, the NodeServer abstract class is available. This class should be abstracted. It binds appropriate handlers to the API events and contains a suitable run loop. It should serve as a basic structure for any node server.

Parameters:

- poly (polyglot.nodeserver_api.PolyglotConnector) – The connected Polyglot connection
- optional shortpoll (int) – The seconds between poll events
- optional longpoll (int) – The second between longpoll events

*add_node(node_address, node_def_id, node_primary_addr, node_name, callback=None, timeout=None, **kwargs)[source]*

Add this node to the polyglot

Returns bool: True on success

long_poll()[source]

Called every longpoll seconds for less important polling.

on_add_all(request_id=None)[source]

Received add all command from ISY

Parameters: optional request_id (str) – Status request id

Returns bool: True on success

on_added(node_address, node_def_id, primary_node_address, name)[source]

Received node added report from ISY

Parameters:

- node_address (str) – The address of the node to act on
- node_def_id (str) – The node definition id
- primary_node_address (str) – The node server's primary node address
- name (str) – The node's friendly name
- optional request_id (str) – Status request id

Returns bool: True on success

*on_cmd(node_address, command, value=None, uom=None, request_id=None, **kwargs)[source]*

Received run command from ISY

Parameters:

- node_address (str) – The address of the node to act on
- command (str) – The command to run
- value (optional) – The value of the command's unnamed parameter
- uom (optional) – The units of measurement for the unnamed parameter
- optional request_id (str) – Status request id
- <pN>.<uomN> (optional) – The value of parameter pN with units uomN

Returns bool: True on success

*on_config(**data)[source]*

Received configuration data from Polyglot

Parameters: data (dict) – Configuration data

Returns bool: True on success

on_disabled(node_address)[source]

Received node disabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_enabled(node_address)[source]

Received node enabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

*on_exit(*args, **kwargs)[source]*

Polyglot has triggered a clean shutdown. Generally, this method does not need to be orwritten.

Returns bool: True on success

on_install(profile_number)[source]

Received install command from ISY

Parameters: profile_number (int) – Noder Server's profile number

Returns bool: True on success

on_query(node_address, request_id=None)[source]

Received query command from ISY

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

on_removed(node_address)[source]

Received node removed report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_renamed(node_address, name)[source]

Received node renamed report from ISY

Parameters:

- node_address (str) – The address of the node to act on
- name (str) – The node's friendly name

Returns bool: True on success

*on_result(seq, status_code, elapsed, text, retries, reason_code, **kwargs)[source]*

Handles a result message, which contains the result from a REST API call to the ISY. The result message is uniquely identified by the seq id, and will always contain at least the numeric status.

*on_statistics(**kwargs)[source]*

Handles a statistics message, which contains various statistics on the operation of the Polyglot server and the network communications.

on_status(node_address, request_id=None)[source]

Received status command from ISY

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

poll()[source]

Called every shortpoll seconds to allow for updating nodes.

poly = None

The Polyglot Connection

Type: polyglot.nodeserver_api.PolyglotConnector

*register_result_cb(func, **kwargs)[source]*

Registers a callback function to handle a result. Returns the unique sequence ID to be passed to the function whose result is to be handled by the registered callback.

*report_status(node_address, driver_control, value, uom, callback=None, timeout=None, **kwargs)[source]*

Report a node status to the ISY

Returns bool: True on success

*restcall(api, callback=None, timeout=None, **kwargs)[source]*

Sends an asynchronous REST API call to the ISY. Returns the unique seq id (that can be used to match up the result later on after the REST call completes).

run()[source]

Run the Node Server. Exit when triggered. Generally, this method should not be overwritten.

setup()[source]

Setup the node server. All node servers must override this method and call it thru super. Currently it only sets up the reference for the logger.

smsg(strng)[source]

Logs/sends a diagnostic/debug, informative, or error message. Individual node servers can override this method if they desire to redirect or filter these messages.

tock()[source]

Called every few seconds for internal housekeeping.

class polyglot.nodeserver_api.SimpleNodeServer(poly, shortpoll=1, longpoll=30)[source]

Simple Node Server with basic functionality built-in. This class inherits from **polyglot.nodeserver_api.NodeServer** and is the best starting point when developing a new node server. This class implements the idea of manifests which are dictionaries that contain the relevant information about all of the nodes. The manifest gets sent to Polyglot to be saved as part of the configuration. This allows the node server to automatically recall its last known values when it is restarted.

*add_node(*args, **kwargs)[source]*

Add node to the Polyglot and the nodes dictionary.

Parameters: node (polyglot.nodeserver_api.Node) – The node to add

Returns boolean:

Indicates success or failure of node addition

exist_node(address)[source]

Check if a node exists by its address.

Parameters: address (str) – The node address

Returns bool: True if the node exists

get_node(address)[source]

Get a node by its address.

Parameters: address (str) – The node address

Returns polyglot.nodeserver_api.Node:

If found, otherwise False

nodes = OrderedDict()

Nodes registered with this node server. All nodes are automatically added by the add_node method. The keys are the node IDs while the values are instances of **polyglot.nodeserver_api.Node**. Classes inheriting can access this directly, but the prefered method is by using get_node or exist_node methods.

*on_add_all(*args, **kwargs)[source]*

Adds all nodes to the ISY. Also sends requests reponses when necessary.

Parameters: optional request_id (str) – Status request id

Returns bool: True on success

on_added(node_address, node_def_id, primary_node_address, name)[source]

Internally indicates that the specified node has been added to the ISY.

Parameters:

- node_address (str) – The address of the node to act on
- node_def_id (str) – The node definition id
- primary_node_address (str) – The node server's primary node address
- name (str) – The node's friendly name
- optional request_id (str) – Status request id

Returns bool: True on success

*on_cmd(*args, **kwargs)[source]*

Runs the specified command on the specified node. Also sends requests responses when necessary.

Parameters:

- node_address (str) – The address of the node to act on
- command (str) – The command to run
- value (optional) – The value of the command's unnamed parameter
- uom (optional) – The units of measurement for the unnamed parameter
- optional request_id (str) – Status request id
- <pN>.<uomN> (optional) – The value of parameter pN with units uomN

Returns bool: True on success

on_disabled(node_address)[source]

Received node disabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_enabled(node_address)[source]

Received node enabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

*on_exit(*args, **kwargs)[source]*

Triggers a clean shut down of the node server by saving the manifest, clearing the IO, and stopping.

Returns bool: True on success

*on_query(*args, **kwargs)[source]*

Queries each node and reports all control values to the ISY. Also responds to report requests if necessary.

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

on_removed(node_address)[source]

Internally indicates that a node has been removed from the ISY.

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_renamed(node_address, name)[source]

Changes the node name internally to match the ISY.

Parameters:

- node_address (str) – The address of the node to act on
- name (str) – The node's friendly name

Returns bool: True on success

*on_status(*args, **kwargs)[source]*

Reports the requested node's control values to the ISY without forcing a query. Also sends requests reponses when necessary.

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

update_config(replace_manifest=False)[source]

Updates the configuration with new node manifests and sends the configuration to Polyglot to be saved.

Parameters: replace_manifest (boolean) – replace or merge existing manifest

5.5.2.8Helper Functions

polyglot.nodeserver_api.auto_request_report(fun)[source]

Python decorator to automate request reporting. Decorated functions must return a boolean value indicating their success or failure. It the argument request_id is passed to the decorated function, a response will be sent to the ISY. This decorator is implemented in the SimpleNodeServer.

5.5.2.9MQTT

MQTT communication mechanism has been developed for communications between Polyglot and the nodeservers. This becomes useful so that you can connected many different nodes to your nodeserver without having the distributed code running directly on the same platform Polyglot resides on. For example, I have a system of nodes that all communicates over MQTT to keep updated, and I don't want to wrap that entire system into Polyglot. I can develop a simple nodeserver that runs and listens to your existing MQTT to allow communications without having to completely redevelop the system as a polyglot nodeserver. This lays the framework of allowing for completely distributed nodeservers.

5.5.2.9.1Usage

To use the MQTT Subsystem for communications you must include the following into your server.json of your nodeserver:

```
"interface": "mqtt",

"mqtt_server": "192.168.1.20",

"mqtt_port": "1883",
```

When Polyglot reads the server.json file and see's the MQTT interface command it enables the MQTT subsystem for that particular nodeserver. It then sends these params along with

the normal params and config to the nodeserver over STDIN, which allows for dynamic passing of the mqtt_server and mqtt_port information to the nodeserver. This is done to avoid having to set the connection information on both server.json AND your nodeserver. If the “interface” setting is missing or anything other than “mqtt” case insensitive, then the standard STDIN/STDOUT mechansims will be used, regardless of if mqtt_server and mqtt_port are provided.

When the Polyglot nodeserver manager enables the MQTT interface, it automatically connects to the MQTT broker and subscribes to the topic:

```
udi/polyglot/<nodeserver name>/poly
```

This topic is where your nodeserver will publish any commands destined for Polyglot for example the pong keepalive messages. These messages are formatted in JSON exactly like the exisiting STDOUT messages.

When creating a nodeserver that uses MQTT, you should listen on both STDIN and MQTT and respond based on which mechanism was used. MQTT is a network resource, therefore inherently it is possible to have network disruption or connection issues. When using MQTT in your nodeserver you will subscribe to the topic:

```
udi/polyglot/<nodeserver name>/node
```

There is a retianed message that you will get upon subscription to the topic above reflecting the current connection state of the Polyglot side. The json messages are listed below.:

```
{"connected": {}}

or

{"disconnected": {}}
```

This state is how you should respond to Polyglot. Using STDOUT if disconnected or MQTT if connected. A “Last Will and Testament” message is configured on the Polyglot side to always make sure the state is accurate even on catastrophic failure. The nodeserver should be configured with this same feature. An example is provided in the Node Server example section of the documentation.

5.5.2.9.2MQTT Subsystem Class

class polyglot.nodeserver_manager.mqttSubsystem(parent)[source]

mqttSubsystem class instantiated if interface is mqtt in server.json

Parameters: parent (polyglot.nodeserver_manager.NodeServer) – The NodeServer object that called this function

start()[source]

The client start method. Starts the thread for the MQTT Client and publishes the connected message.

stop()[source]

The client stop method. If the client is currently connected stop the thread and disconnect. Publish the disconnected message if clean shutdown.

5.5.3 Python Node Server Example

The following is a brief example of some implemented node servers written in Python. The examples included are pulled from the Philips Hue Node Server and may not be current with the actual code used in that node server and is redacted a bit for clarity, but will serve as a solid jumping off point for defining the process by which a new node server can be developed.

5.5.3.1Node Type Definition

Some may find it easiest to start by developing all the types of nodes that the node server may be controlling. As these are being defined in code, it may be best to also define them in the file that will eventually make up the profile.zip file. Documentation for profile files is available in the ISY Virtual Node Server API documentation.

Below is the definition for a Hue color changing light.

```
from converters import RGB_2_xy, color_xy, color_names

from functools import partial

from polyglot.nodeserver_api import Node
```

```
def myint(value):
    """ round and convert to int """
    return int(round(float(value)))


def myfloat(value, prec=4):
    """ round and return float """
    return round(float(value), prec)


class HueColorLight(Node):
    """ Node representing Hue Color Light """


    def __init__(self, parent, address, name, lamp_id, manifest=None):
        super(HueColorLight, self).__init__(parent, address, name, manifest)
        self.lamp_id = int(lamp_id)


    def query(self):
        """ command called by ISY to query the node. """
        updates = self.parent.query_node(self.address)
        if updates:
            self.set_driver('GV1', updates[0], report=False)
            self.set_driver('GV2', updates[1], report=False)
            self.set_driver('ST', updates[2], report=False)
```

```
        self.report_driver()
        return True
    else:
        return False


def _set_brightness(self, value=None, **kwargs):
    """ set node brightness """
    # pylint: disable=unused-argument
    if value is not None:
        value = int(value / 100. * 255)
        if value > 0:
            command = {'on': True, 'bri': value}
        else:
            command = {'on': False}
    else:
        command = {'on': True}
    return self._send_command(command)


def _on(self, **kwargs):
    """ turn light on """
    status = kwargs.get("value")
    return self._set_brightness(value=status)
```

```
def _off(self, **kwargs):
    """ turn light off """
    # pylint: disable=unused-argument
    return self._set_brightness(value=0)


def _set_color_rgb(self, **kwargs):
    """ set light RGB color """
    color_r = kwargs.get('R.uom56', 0)
    color_g = kwargs.get('G.uom56', 0)
    color_b = kwargs.get('B.uom56', 0)
    (color_x, color_y) = RGB_2_xy(color_r, color_g, color_b)
    command = {'xy': [color_x, color_y], 'on': True}
    return self._send_command(command)


def _set_color_xy(self, **kwargs):
    """ set light XY color """
    color_x = kwargs.get('X.uom56', 0)
    color_y = kwargs.get('Y.uom56', 0)
    command = {'xy': [color_x, color_y], 'on': True}
    return self._send_command(command)
```

```
def _set_color(self, value=None, **_):
    """ set color from index """
    ind = int(value) - 1

    if ind >= len(color_names):
        return False

    cname = color_names[int(value) - 1]
    color = color_xy(cname)
    return self._set_color_xy(
        **{'X.uom56': color[0], 'Y.uom56': color[1]})

def _send_command(self, command):
    """ generic method to send command to hue hub """
    responses = self.parent.hub.set_light(self.lamp_id, command)
    return all(
        [list(resp.keys())[0] == 'success' for resp in responses[0]])

_drivers = {'GV1': [0, 56, myfloat], 'GV2': [0, 56, myfloat],
            'ST': [0, 51, myint]}
""" Driver Details:
GV1: Color X
```

```
GV2: Color Y

ST: Status / Brightness

"""

_commands = {'DON': _on, 'DOF': _off,

        'SET_COLOR_RGB': _set_color_rgb,

        'SET_COLOR_XY': _set_color_xy,

        'SET_COLOR': _set_color}

node_def_id = 'COLOR_LIGHT'
```

As can be seen here, one method is defined for each of the commands that the node may run. The query method from the Node ABC is also overwritten to provide the desired functionality. An additional method called _send_command is also created. This is not called by the ISY directly, but is a helper used to send information to the Hue device. This method calls a method from a third party library that connects to the Hue lighting system.

Additionally, the _drivers, _command, and node_def_id properties are overwritten. This must be done by every node class as it instructs the node server classes on how to interact with this node. Custom formatters myint and myfloat are used to format the control values.

This process must be repeated for each type of node that is desired.

5.5.3.2Node Server Creation

Once all the nodes are defined, the node server class can be created.

from polyglot.nodeserver_api import SimpleNodeServer, PolyglotConnector

```
# ... additional imports are redacted for clarity
class HueNodeServer(SimpleNodeServer):
    """ Phillips Hue Node Server """
    hub = None
    def setup(self):
        """ Initial node setup. """
        super(SimpleNodeServer, self).setup()
        # define nodes for settings
        manifest = self.config.get('manifest', {})
        HubSettings(self, 'hub', 'Hue Hub', manifest)
        self.connect()
        self.update_config()

    def connect(self):
        """ Connect to Phillips Hue Hub """
        # get hub settings
        hub = self.get_node('hub')
        ip_addr = '{}.{}.{}.{}'.format(
            hub.get_driver('GV1')[0], hub.get_driver('GV2')[0],
```

```
        hub.get_driver('GV3')[0], hub.get_driver('GV4')[0])


    # ... Connects to the hub and validate connection. Redacted for clarity.


def poll(self):
    """ Poll Hue for new lights/existing lights' statuses """


    # ... Connects to Hue Hub and gets current values for lights,
    #     stores in dictionary called lights. Redacted for clarity.


    for lamp_id, data in lights.items():
        address = id_2_addr(data['uniqueid'])
        name = data['name']


        lnode = self.get_node(address)
        if not lnode:
            # Add the light to the Node Server if it doesn't already
            # exist. Sets the primary to the 'hub' Node.
            # This automatically adds the light to the ISY.
            lnode = HueColorLight(self, address,
                                  name, lamp_id,
                                   self.get_node('hub'), manifest)
```

```
        (color_x, color_y) = [round(val, 4)
                              for val in data['state']['xy']]
        brightness = round(data['state']['bri'] / 255. * 100., 4)
        brightness = brightness if data['state']['on'] else 0
        lnode.set_driver('GV1', color_x)
        lnode.set_driver('GV2', color_y)
        lnode.set_driver('ST', brightness)

    return True

def query_node(self, lkp_address):
    """ find specific node in api. """

    # ... Polls Hue Hub for current specified light values, and updates
    #     Node object with new values. Works very similarly to poll
    #     above. Redacted for clarity.

def _get_api(self):
    """ get hue hub api data. """

    # ... Uses third party library to get updated Hue Hub information.
```

```
    #     Redacted for clarity.



def long_poll(self):

    """ Save configuration every 30 seconds. """

    self.update_config()

    # In this example, the configuration is autoatically saved every

    # 30 seconds. Make sure your node server saves its configuration

    # at some point.
```

This example class contains four methods that are not part of the abstract class. They are setup, connect, query_node, and _get_api. These functions will probably not appear in all node servers and are very specific to this one.

However, the setup method is a good way to handle any node server setup that must be done that is specific to your node server. In this example, the primary node, the Hue Hub, is created and a connection is attempted.

This class also stores an object called hub as an attribute. This object is an instance of a class from the third-party library used. This object is the actual connection to the Hue Hub. It may be best to follow a similar method when creating node servers so that the code that handles the connection is differentiated from the code that organizes the nodes.

The poll and long_poll methods from the abstract class are used in this example. The Hue Hub sends no event stream, so it must be polled for updates. This is done in the poll method. The long_poll method is utilized to ensure the configuration data is saved consistently. These methods do not need to be manually called anywhere as they are automatically invoked from the run loop every (approximately) 1 second and 30 seconds respectively.

5.5.3.3Starting the Node Server

Finally, your program must be able to initialize itself and begin running the node server. In Python, it will very nearly look like this.

```
def main():
    """ setup connection, node server, and nodes """
    poly = PolyglotConnector()
    nserver = HueNodeServer(poly)
    poly.connect()  # begin listening for Polyglot commands
    poly.wait_for_config()  # This is best practice to not start until
                    # Polyglot has begun communicating. This way,
                    # Polyglot will not miss messages sent from
                    # the node server.
    nserver.setup()  # setup method is specific to this example
    nserver.run()  # begin node server run loop


if __name__ == "__main__":
    main()
```

5.5.3.4Installing the Node Server

Once all of this has been coded and all the appropriate files (documented in the last section) have been created, the node server directory can be placed in the configuration directory in a subfolder called node_servers. Polyglot should then be restarted to trigger the discovery of new node server types. If there is an issue with your node server, it will appear in the log.

5.5.3.5Custom Node Server Configuration File

You may specify a custom configuration file in the server.json file as such:

```
"configfile": "customfile.yaml"
```

This should be placed in the top level of configuration, for example right after “executable”. If no “configfile” is specified, Polyglot will look for “config.yaml” in the root node_server folder(the same location as the server.json). If either file is found, then the contents will be loaded into a dictionary for consumption. The poly.nodeserver_config variable holds this dictionary.

Your node server may modify this dictionary as necessary and use the function

```
write_nodeserver_config():
```

This method has two parameters that are optional. The defaults are shown here:

```
default_flow_style = False

indent = 4
```

The default_flow_style is the formatting of the YAML file, look at the PyYAML documentation for specifics. The indent parameter is the number of spaces indented for each subline in the file. The default for our method is 4 because Python...

This method checks for any differences in the running configuration and the existing file, and refrains from writing if they are identical. This method is also automatically called upon a normal shutdown of Polyglot. If Polyglot shuts down abnormally, it will not record any changes that you made if you did not call the write_nodeserver_config() method.

5.5.4 Polyglot Node Server API

Documented here is the JSON API used for communication between Polyglot and the Node Server processes. This API will never be referenced directly by either by an end user and

will rarely be referenced by a developer. It is documented here for continuity. Nearly each command and its arguments maps to a command and arguments specified in the ISY Virtual Node Server API documentation. The only exceptions are the additions of some commands necessary for Polyglot's operation.

5.5.4.1General Format

In general, each API message is formatted as such:

```
{COMMAND: {ARG_NAME_1: ARG_VALUE_1, ..., ARG_NAME_N: ARG_VALUE_N}}
```

All of the arguments are named. Each message ends with a new line and will contain no new lines. Each message will contain only one command. Never will multiple command be sent in the same message.

5.5.4.2Node Server STDIN - Polyglot to Node Server

The following messages may be sent from Polyglot to the Node Server to trigger an action inside of the Node Server.

{'config': {... arbitrary data saved by the node server ...}}

This command is the first one sent to the node server and is only sent once. The arguments dictionary will be of an arbitrary structure and will match what the Node Server had last saved.

{'install': {'profile_number': ...}}

Instructs the node server to install itself with the specified profile_number.

{"params": {"profile": 8, "pgver": "0.0.4", "name": "nodeservername", "pgapiver": "1", "sandbox": "/home/Polyglot/config/nodeservername", "configfile": "config.yaml", "interface": "mqtt", "path": "/home/Polyglot/config/node_servers/nodeservername", "isyver": "5.0.4", "mqtt_server": "pi3", "mqtt_port": "1883"}}

Params passed back from Polyglot to the node server with info about the node server.

{'query': {'node_address': ..., 'request_id': ...}}

Instructs the node server to query a node. request_id is optional.

{'status': {'node_address': ..., 'request_id': ...}}

Requests the node server to send current node status to the ISY. request_id is optional.

{'add_all': {'request_id': ...}}

Requests that the node server add all its nodes to the ISY. request_id is optional.

{'added': {'node_address': ..., 'node_def_id': ..., 'primary_node_address': ..., 'name': ...}}

Indicates that the node has been added to the ISY.

{'removed': {'node_address': ...}}

Indicates that the node has been removed from the ISY.

{'renamed': {'node_address': ..., 'name': ...}}

Indicates that the node has been renamed in the ISY.

{'enabled': {'node_address': ...}}

Indicates that the node has been enabled in the ISY.

{'disabled': {'node_address': ...}}

Indicates that the node has been disabled in the ISY.

{'cmd': {'node_address': ..., 'command': ..., *'value': ...., *'uom': ..., *'<pn>.<uomn>': ..., *'request_id': ...}}

Instructs the node server to run the specified command on the specified node. value and uom are optional and described the unnamed parameter. They will always appear together. <pn>.<uomn> will be repeated as necessary to described the unnamed parameters. They are also optional. request_id is optional.

{'ping': {}}

This is a command from Polyglot requesting a Pong response. This is handled in the PolyglotConnector class.

{'exit': {}}

This command is Polyglot instructing the node server to cleanly shut down.

5.5.4.3Node Server STDOUT - Node Server to Polyglot

The following messages are accepted by Polyglot from the Node Server and will typically instruct Polyglot to send a response upstream to the ISY.

{'config': {... arbitrary data saved by the node server ...}}

Sends configuration data to Polyglot to be saved. This data will be sent back to the Node Server, exactly as it has been sent to Polyglot, the next time the Node Server is started.

{'install': {}}

Install the node server on the ISY. This has not been implemented yet.

{'status': {'node_address': ..., 'driver_control': ..., 'value': ..., 'uom': ...}}

Reports a node's driver status.

{'command': {'node_address': ..., 'command', ..., 'value': ...., 'uom': ..., '<pn>.<uomn>': ...}}

Reports that a command has been run on a node. value and uom are optional and described the unnamed parameter. They will always appear together. <pn>.<uomn> will be repeated as necessary to described the unnamed parameters. They are also optional.

{'add': {'node_address': ..., 'node_def_id': ..., 'primary': ..., 'name': ...}}

Adds a node to the ISY.

{'change': {'node_address': ..., 'node_def_id': ...}}

Changes the node's definition in the ISY.

{'remove': {'node_address': ...}}

Instructs the ISY to remove a node.

{'request': {'request_id': ..., 'result': ...}}

Replies to the ISY indicating that a request has been finished either successfully or unsuccessfully. The result parameter must be a boolean indicating this.

{'pong': {}}

The proper response to a Ping command. Must be recieved within 30 seconds of a Ping command or Polyglot assumes the Node Server has stalled and kills it. This is handled automatically in the PolyglotConnector class.

{'exit': {}}

Indicates to Polyglot that the node server has exited and is now closing. This is the last message sent from a node server. All messages following this will be ignored. It is not guaranteed that the node server process will continue to run after this command is sent.

5.5.4.4Node Server STDERR - Node Server to Polyglot

STDERR messages have no structured formatting, they are free flowing text. Anything received by Polyglot through this stream will not be processed and will be immediately logged as an error. Do not send personal information in error messages as they will always be logged regardless of the log verbosity.

5.5.5 Polyglot Methods and Classes

5.5.5.1Module

These are the classes and methods from the Polyglot module.

5.5.5.1.1Config Manager

The configuration management module for Polyglot

class polyglot.config_manager.ConfigManager(config_dir)[source]

Configuration Manager class

Parameters: config_dir – The configuration directory to use

decode(encoded)[source]

Decode passwords and return decoded

encode()[source]

Encode passwords and return an encoded copy

*make_path(*args)[source]*

make a path to a file in the config directory

nodeserver_sandbox(url_base)[source]

make and return a sandbox directory for a node server.

read()[source]

Reads configuration file

*update(*args, **kwargs)[source]*

Update the configuration with values in dictionary

write()[source]

Writes configuration file

5.5.5.1.2Core

The primary functionality for the Polyglot application

class polyglot.core.Polyglot(config_dir)[source]

Core class

Parameters: config_dir – Directory where configuration is stored

Variables: config – Dictionary of current config

get_log()[source]

Read and return the log file contents.

run()[source]

Run the Polyglot server

setup()[source]

Setup Polyglot to resume the last known state

*stop(*args)[source]*

Stops the Polyglot server

update_config()[source]

Signal Polyglot to fetch updated configuration.

5.5.5.1.3Nodeserver API

This library consists of four classes and one function to assist with node server development. The classes **polyglot.nodeserver_api.NodeServer** and **polyglot.nodeserver_api.SimpleNodeServer** and basic structures for creating a node server. The class **polyglot.nodeserver_api.Node** is used as an abstract class to crate custom nodes for node servers. The class **polyglot.nodeserver_api.PolyglotConnector** is a bottom level implementation of the API used to communicate between Polyglot and your node server. Finally, included in this library is a method decorator, **polyglot.nodeserver_api.auto_request_report()**, that wraps functions and methods to automatically handle report requests from the ISY.

class polyglot.nodeserver_api.Node(parent, address, name, primary=True, manifest=None)[source]

Abstract class for representing a node in a node server.

Parameters:

- parent (polyglot.nodeserver_api.NodeServer) – The node server that controls the node
- address (str) – The address of the node in the ISY without the node server ID prefix
- name (str) – The name of the node
- primary (polyglot.nodeserver_api.Node or True if this node is the primary.) – The primary node for the device this node belongs to, or True if it's the primary.
- manifest (dict or None) – The node manifest saved by the node server

_drivers = {}

The drivers controlled by this node. This is a dictionary of lists. The key's are the driver names as defined by the ISY documentation. Each list contains at least three values: the initial value, the UOM identifier, and a function that will properly format the value before assignment. The fourth value is optional – if provided, and set to "False", the driver's value will not be recorded in the manifest (this is useful to reduce I/O when there is no benefit to restoring the driver's value on restart).

Insteon Dimmer Example:

```
_drivers = {

   'ST': [0, 51, int],

   'OL': [100, 51, int],

   'RR': [0, 32, int]

}
```

Pulse Time Example:

```
_drivers = {

   'ST': [0, 56, int, False],

}
```

_commands = {}

A dictionary of the commands that the node can perform. The keys of this dictionary are the names of the command. The values are functions that must be defined in the node object that perform the necessary actions and return a boolean indicating the success or failure of the command.

add_node()[source]

Adds node to the ISY

Returns boolean:

Indicates success or failure of node addition

get_driver(driver=None)[source]

Gets a driver's value

Parameters: driver (str or None) – The driver to return the value for

Returns: The current value of the driver

manifest

The node's manifest entry. Indicates the current value of each of the drivers. This is called by the node server to create the full manifest.

Type: dict

node_def_id = ''

The node's definition ID defined in the node server's profile

query()[source]

Abstractly queries the node. This method should generally be overwritten in development.

Returns boolean:

Indicates success or failure of node query

report_driver(driver=None)[source]

Reports a driver's current value to ISY

Parameters: driver (str or None) – The name of the driver to report. If None, all drivers are reported.

Returns boolean:

Indicates success or failure to report driver value

*report_isycmd(isycommand, value=None, uom=None, timeout=None, **kwargs)[source]*

Sends a single command from the node server to the ISY, optionally passing in a value.

No formatting, and little validation, of the isy cmd, value, or uom is done by this simple low-level API. It is up to the caller to ensure correctness.

Parameters:

- isycommand (str) – Name of the ISY command to send (e.g. 'DON')
- value (string, float, int, or None) – (optional) The value to be sent for the command
- uom (int or None) – (optional) - The value's unit of measurement. If provided, overrides the uom defined for this command
- timeout (int or None) – (optional) - the number of seconds before this command expires and is discarded

Returns boolean:

Indicates success or failure to queue for sending (Note: does NOT indicate if actually delivered)

*run_cmd(command, **kwargs)[source]*

Runs one of the node's commands.

Parameters:

- command (str) – The name of the command
- kwargs (dict) – The parameters specified by the ISY in the incoming request. See the ISY Node Server documentation for more information.

Returns boolean:

Indicates success or failure of command

set_driver(driver, value, uom=None, report=True)[source]

Updates the value of one of the node's drivers. This will pass the given value through the driver's formatter before assignment.

Parameters:

- driver (str) – The name of the driver
- value – The new value for the driver
- uom (int or None) – The given values unit of measurement. This should correspond to the UOM IDs used by the ISY. Refer to the ISY documentation for more information.
- report (boolean) – Indicates if the value change should be reported to the ISY. If False, the value is changed silently.

Returns boolean:

Indicates success or failure to set new value

smsg(strng)[source]

Logs/sends a diagnostic/debug, informative, or error message. Individual node servers can override this method if they desire to redirect or filter these messages.

class polyglot.nodeserver_api.NodeServer(poly, shortpoll=1, longpoll=30)[source]

It is generally desirable to not be required to bind to each event. For this reason, the NodeServer abstract class is available. This class should be abstracted. It binds appropriate handlers to the API events and contains a suitable run loop. It should serve as a basic structure for any node server.

Parameters:

- poly (polyglot.nodeserver_api.PolyglotConnector) – The connected Polyglot connection
- optional shortpoll (int) – The seconds between poll events
- optional longpoll (int) – The second between longpoll events

*add_node(node_address, node_def_id, node_primary_addr, node_name, callback=None, timeout=None, **kwargs)[source]*

Add this node to the polyglot

Returns bool: True on success

long_poll()[source]

Called every longpoll seconds for less important polling.

on_add_all(request_id=None)[source]

Received add all command from ISY

Parameters: optional request_id (str) – Status request id

Returns bool: True on success

on_added(node_address, node_def_id, primary_node_address, name)[source]

Received node added report from ISY

Parameters:

- node_address (str) – The address of the node to act on
- node_def_id (str) – The node definition id
- primary_node_address (str) – The node server's primary node address
- name (str) – The node's friendly name
- optional request_id (str) – Status request id

Returns bool: True on success

*on_cmd(node_address, command, value=None, uom=None, request_id=None, **kwargs)[source]*

Received run command from ISY

Parameters:

- node_address (str) – The address of the node to act on
- command (str) – The command to run
- value (optional) – The value of the command's unnamed parameter
- uom (optional) – The units of measurement for the unnamed parameter
- optional request_id (str) – Status request id
- <pN>.<uomN> (optional) – The value of parameter pN with units uomN

Returns bool: True on success

*on_config(**data)[source]*

Received configuration data from Polyglot

Parameters: data (dict) – Configuration data

Returns bool: True on success

on_disabled(node_address)[source]

Received node disabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_enabled(node_address)[source]

Received node enabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

*on_exit(*args, **kwargs)[source]*

Polyglot has triggered a clean shutdown. Generally, this method does not need to be overwritten.

Returns bool: True on success

on_install(profile_number)[source]

Received install command from ISY

Parameters: profile_number (int) – Noder Server's profile number

Returns bool: True on success

on_query(node_address, request_id=None)[source]

Received query command from ISY

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

on_removed(node_address)[source]

Received node removed report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_renamed(node_address, name)[source]

Received node renamed report from ISY

Parameters:

- node_address (str) – The address of the node to act on
- name (str) – The node's friendly name

Returns bool: True on success

*on_result(seq, status_code, elapsed, text, retries, reason_code, **kwargs)[source]*

Handles a result message, which contains the result from a REST API call to the ISY. The result message is uniquely identified by the seq id, and will always contain at least the numeric status.

*on_statistics(**kwargs)[source]*

Handles a statistics message, which contains various statistics on the operation of the Polyglot server and the network communications.

on_status(node_address, request_id=None)[source]

Received status command from ISY

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

poll()[source]

Called every shortpoll seconds to allow for updating nodes.

poly = None

The Polyglot Connection

Type: polyglot.nodeserver_api.PolyglotConnector

*register_result_cb(func, **kwargs)[source]*

Registers a callback function to handle a result. Returns the unique sequence ID to be passed to the function whose result is to be handled by the registered callback.

*report_status(node_address, driver_control, value, uom, callback=None, timeout=None, **kwargs)[source]*

Report a node status to the ISY

Returns bool: True on success

*restcall(api, callback=None, timeout=None, **kwargs)[source]*

Sends an asynchronous REST API call to the ISY. Returns the unique seq id (that can be used to match up the result later on after the REST call completes).

run()[source]

Run the Node Server. Exit when triggered. Generally, this method should not be overwritten.

setup()[source]

Setup the node server. All node servers must override this method and call it thru super. Currently it only sets up the reference for the logger.

smsg(strng)[source]

Logs/sends a diagnostic/debug, informative, or error message. Individual node servers can override this method if they desire to redirect or filter these messages.

tock()[source]

Called every few seconds for internal housekeeping.

class polyglot.nodeserver_api.PolyglotConnector[source]

Polyglot API implementation. Connects to Polyglot and handles node server IO.

Raises:RuntimeError

add_node(node_address, node_def_id, primary, name, timeout=None, seq=None)[source]

Adds a node to the ISY. To make this node the primary, set primary to the same value as node_address.

Parameters:

- node_address (str) – The full address of the node (e.g. 'dimmer_1')
- node_def_id (str) – The id of the node definition to use for this node
- primary (str) – The primary node for the device this node belongs to
- name (str) – The name of the node
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

change_node(node_address, node_def_id, timeout=None, seq=None)[source]

Changes the node definition to use for an existing node. An example of this is may be to change a thermostat node from Fahrenheit to Celsius.

Parameters:

- node_address (str) – The full address of the node (e.g. ‘dimmer_1’)
- node_def_id (str) – The id of the node definition to use for this node
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

commands = ['config', 'install', 'query', 'status', 'add_all', 'added', 'removed', 'renamed', 'enabled', 'disabled', 'cmd', 'ping', 'exit', 'params', 'result', 'statistics']

Commands that may be invoked by Polyglot

connect()[source]

Connects to Polyglot if not currently connected

connected

Indicates if the object is connected to Polyglot. Can be set to control connection with Polyglot.

Type: boolean

disconnect()[source]

Disconnects from Polyglot. Blocks the thread until IO stream is clear

*exit(*args, **kwargs)[source]*

Tells Polyglot that this Node Server is done.

*get_params(**kwargs)[source]*

Get the params from nodeserver and makes them available to the nodeserver api

*install(*args, **kwargs)[source]*

Abstract method to install the node server in the ISY. This has not been implemented yet and running it will raise an error.

Raises:NotImplementedError

listen(event, handler)[source]

Register an event handler. Returns True on success. Event names are defined in commands. Handlers must be callable.

Parameters:

- event (str) – Then event name to listen for.
- handler (callable) – The callable event handler.

logger = None

logger is initialized after the node server wait_for_config completes by the setup_log method and the log file is located in the node servers sandbox. Once wait_for_config is complete, you can call poly.logger.info('This variable is set to %s', variable)

*pong(*args, **kwargs)[source]*

Sends pong reply to Polyglot's ping request. This verifies that the communication between the Node Server and Polyglot is functioning.

remove_node(node_address, timeout=None, seq=None)[source]

Removes a node from the ISY. A node cannot be removed if it is the primary node for at least one other node.

Parameters:

node_address (str) – The full address of the node (e.g. 'dimmer_1')

timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY

seq (str or int) – (optional) set to unique id if result callback desired

*report_command(node_address, command, value=None, uom=None, timeout=None, seq=None, **kwargs)[source]*

Sends a command to the ISY that may be used in programs and/or scenes. A common use of this is a physical switch that somebody turns on or off. Each time the switch is used, a command should be reported to the ISY. These are used for scenes and control conditions in ISY programs.

Parameters:

- node_address (str) – The full address of the node (e.g. 'switch_1)
- command (str) – The command to perform (e.g. 'DON', 'CLISPH', etc.)
- value (str, int, or float) – Optional unnamed value the command used
- uom (int or str) – Optional units of measurement of value
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired
- <pN>.<uomN> (optional) – Nth Parameter name (e.g. 'level') . Unit of measure of the Nth parameter (e.g. 'seconds', 'uom58')

report_request_status(request_id, success, timeout=None, seq=None)[source]

When the ISY sends a request to the node server, the request may contain a 'requestId' field. This indicates to the node server that when the request is completed, it must send a fail or success report for that request. This allows the ISY to in effect, have the node server synchronously perform tasks. This message must be sent after all other messages related to the task have been sent.

For example, if the ISY sends a request to query a node, all the results of the query must be sent to the ISY before a fail/success report is sent.

Parameters:

- request_id (str) – The request ID the ISY supplied on a request to the node server.
- success (bool) – Indicates if the request was sucessful
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

report_status(node_address, driver_control, value, uom, timeout=None, seq=None)[source]

Updates the ISY with the current value of a driver control (e.g. the current temperature, light level, etc.)

Parameters:

- node_address (str) – The full address of the node (e.g. 'dimmer_1')
- driver_control (str) – The name of the status value (e.g. 'ST', 'CLIHUM', etc.)
- value (str, float, or int) – The numeric status value (e.g. '80.5')
- uom (int or str) – Unit of measure of the status value
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

*request_stats(*args, **kwargs)[source]*

Sends a command to Polyglot to request a statistics message.

restcall(api, timeout=None, seq=None)[source]

Calls a RESTful api on the ISY. The api is the portion of the url after the "https://isy/rest/" prefix.

Parameters:

- api (str) – The url for the api to call
- timeout (str, float, or int) – (optional) timeout (seconds) for REST call to ISY
- seq (str or int) – (optional) set to unique id if result callback desired

send_config(config_data)[source]

Update the configuration in Polyglot.

Parameters: config_data (dict) – Dictionary of updated configuration

Raises:ValueError

send_error(err_str)[source]

Enqueue an error to be sent back to Polyglot.

Parameters: err_str (str) – Error text to be sent to Polyglot log

smsg(str)[source]

Logs/sends a diagnostic/debug, informative, or error message. Individual node servers can override this method if they desire to redirect or filter these messages.

uptime

The number of sections the connection with Polyglot has been alive

Type: float

wait_for_config()[source]

Blocks the thread until the configuration is received

write_nodeserver_config(default_flow_style=False, indent=4)[source]

Writes any changes to the nodeserver custom configuration file. self.nodeserver_config should be a dictionary. Refrain from calling this in a poll of any kind. Typically, you won't even have to write this unless you are storing custom data you want retrieved on the next run. Saved automatically during normal Polyglot shutdown. Returns True for success, False for failure.

Parameters:

- default_flow_style (boolean) – YAML's default flow style formatting. Default False
- indent (int) – Override the default indent spaces for YAML. Default 4

class polyglot.nodeserver_api.SimpleNodeServer(poly, shortpoll=1, longpoll=30)[source]

Simple Node Server with basic functionality built-in. This class inherits from **polyglot.nodeserver_api.NodeServer** and is the best starting point when developing a

new node server. This class implements the idea of manifests which are dictionaries that contain the relevant information about all of the nodes. The manifest gets sent to Polyglot to be saved as part of the configuration. This allows the node server to automatically recall its last known values when it is restarted.

*add_node(*args, **kwargs)[source]*

Add node to the Polyglot and the nodes dictionary.

Parameters: node (polyglot.nodeserver_api.Node) – The node to add

Returns boolean:

Indicates success or failure of node addition

exist_node(address)[source]

Check if a node exists by its address.

Parameters: address (str) – The node address

Returns bool: True if the node exists

get_node(address)[source]

Get a node by its address.

Parameters: address (str) – The node address

Returns polyglot.nodeserver_api.Node:

If found, otherwise False

nodes = OrderedDict()

Nodes registered with this node server. All nodes are automatically added by the add_node method. The keys are the node IDs while the values are instances of **polyglot.nodeserver_api.Node**. Classes inheriting can access this directly, but the prefered method is by using get_node or exist_node methods.

*on_add_all(*args, **kwargs)[source]*

Adds all nodes to the ISY. Also sends requests reponses when necessary.

Parameters: optional request_id (str) – Status request id

Returns bool: True on success

on_added(node_address, node_def_id, primary_node_address, name)[source]

Internally indicates that the specified node has been added to the ISY.

Parameters:

- node_address (str) – The address of the node to act on
- node_def_id (str) – The node definition id
- primary_node_address (str) – The node server's primary node address
- name (str) – The node's friendly name
- optional request_id (str) – Status request id

Returns bool: True on success

*on_cmd(*args, **kwargs)[source]*

Runs the specified command on the specified node. Also sends requests responses when necessary.

Parameters:

- node_address (str) – The address of the node to act on
- command (str) – The command to run
- value (optional) – The value of the command's unnamed parameter
- uom (optional) – The units of measurement for the unnamed parameter
- optional request_id (str) – Status request id
- <pN>.<uomN> (optional) – The value of parameter pN with units uomN

Returns bool: True on success

on_disabled(node_address)[source]

Received node disabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_enabled(node_address)[source]

Received node enabled report from ISY

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

*on_exit(*args, **kwargs)[source]*

Triggers a clean shut down of the node server by saving the manifest, clearing the IO, and stopping.

Returns bool: True on success

*on_query(*args, **kwargs)[source]*

Queries each node and reports all control values to the ISY. Also responds to report requests if necessary.

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

on_removed(node_address)[source]

Internally indicates that a node has been removed from the ISY.

Parameters: node_address (str) – The address of the node to act on

Returns bool: True on success

on_renamed(node_address, name)[source]

Changes the node name internally to match the ISY.

Parameters:

- node_address (str) – The address of the node to act on
- name (str) – The node's friendly name

Returns bool: True on success

*on_status(*args, **kwargs)[source]*

Reports the requested node's control values to the ISY without forcing a query. Also sends requests reponses when necessary.

Parameters:

- node_address (str) – The address of the node to act on
- optional request_id (str) – Status request id

Returns bool: True on success

update_config(replace_manifest=False)[source]

Updates the configuration with new node manifests and sends the configuration to Polyglot to be saved.

Parameters: replace_manifest (boolean) – replace or merge existing manifest

polyglot.nodeserver_api.auto_request_report(fun)[source]

Python decorator to automate request reporting. Decorated functions must return a boolean value indicating their success or failure. It the argument request_id is passed to the decorated function, a response will be sent to the ISY. This decorator is implemented in the SimpleNodeServer.

5.5.5.1.4Nodeserver Helpers

Helper functions for managing Node Servers.

polyglot.nodeserver_helpers.available_servers()[source]

list all available elements

polyglot.nodeserver_helpers.dirs_in(path, include_path=True)[source]

Returns a list of directories in a path.

Parameters: path – The path to search.

polyglot.nodeserver_helpers.get_path(platform)[source]

Find Node Server Platform path

polyglot.nodeserver_helpers.ignore_dir(path, dir_name)[source]

Deterermine if a directory should be ignored.

5.5.5.1.5Nodeserver Manager

The element management module for Polyglot

class polyglot.nodeserver_manager.NodeServer(pglot, ns_platform, profile_number, nstype, nsexe, nsname, config, sandbox, configfile=None, interface=None, mqtt_server=None, mqtt_port=None)[source]

Node Server Class

alive

Indicates if the Node Server is running.

definition

Return the server definition from server.json

kill()[source]

Kill the node server process.

profile

Return the profile.zip data.

responding

Indicates if the Node Server is responding.

restart()[source]

restart the nodeserver

send_addall(request_id=None)[source]

Send add all request to Node Server.

send_added(node_address, node_def_id, primary_node_address, name)[source]

Send node added confirmation to Node Server.

*send_cmd(node_address, command, value=None, uom=None, request_id=None, **kwargs)[source]*

Send run command signal to Node Server.

send_config()[source]

Send configuration to Node Server.

send_disabled(node_address)[source]

Send node disabled confirmation to Node Server.

send_enabled(node_address)[source]

Send node enabled confirmation to Node Server.

send_exit()[source]

Send exit command to the Node Server.

send_install(profile_number=None)[source]

Send install command to Node Server.

send_params()[source]

Send parameters to Node Server.

send_ping()[source]

Send Ping request to the Node Server.

send_query(node_address, request_id=None)[source]

Send query command to Node Server.

send_removed(node_address)[source]

Send node removed confirmation to Node Server.

send_renamed(node_address, name)[source]

Send node renamed confirmation to Node Server.

send_status(node_address, request_id=None)[source]

Send status request to Node Server.

start()[source]

start the node server

class polyglot.nodeserver_manager.NodeServerManager(pglot)[source]

5.5.5.1.6Node Server Manager

Parameters:

pglot – The parent Polyglot object

Variables:

- pglot – The parent Polyglot object
- servers – Dictionary of active Node Servers

config

Node Server configuration block.

delete(base_url)[source]

Remove a server from Polyglot.

load()[source]

Initial load of the active Node Servers

start_server(ns_platform, profile_number, nsname=None, base=None, config=None)[source]

starts a node server

unload()[source]

Unload all node servers

class polyglot.nodeserver_manager.mqttSubsystem(parent)[source]

mqttSubsystem class instantiated if interface is mqtt in server.json

Parameters: parent (polyglot.nodeserver_manager.NodeServer) – The NodeServer object that called this function

start()[source]

The client start method. Starts the thread for the MQTT Client and publishes the connected message.

stop()[source]

The client stop method. If the client is currently connected stop the thread and disconnect. Publish the disconnected message if clean shutdown.

polyglot.nodeserver_manager.random_string(length)[source]

Generate a random string of uppercase, lowercase, and digits

5.5.5.1.7Utils

Generic utilities used by Polyglot.

class polyglot.utils.AsyncFileReader(fd, handler)[source]

Helper class to implement asynchronous reading of a file in a separate thread. Pushes read lines on a queue to be consumed in another thread.

Source: http://stefaanlippens.net/python-asynchronous-subprocess-pipe-reading

run()[source]

The body of the thread: read lines and put them on the queue.

class polyglot.utils.LockQueue(*args, **kwargs)[source]

Python queue with a locking utility

*put(*args, **kwargs)[source]*

Put item into queue

*put_nowait(*args, **kwargs)[source]*

Put item into queue without waiting

5.5.6 Optional Components

5.5.6.1Docker

These are the instructions for Docker configuration provided by UDI forum and Polyglot contributor i814u2.

5.5.6.1.1Linux x86_64

Create a file named “Dockerfile” (case-sensitive) and place these commands in there:

```
ARG binfile=polyglot.linux.x86_64.pyz

ARG user=polyglot

ARG group=polyglot

ARG uid=1000

ARG gid=1000



RUN addgroup -g ${gid} ${group} \

   && adduser -h /home/${user} -s /bin/sh -G ${group} -D -u ${uid} ${user}



WORKDIR /home/${user}

COPY custom.txt .



RUN pip install --upgrade pip \

&& pip install -r
https://raw.githubusercontent.com/UniversalDevicesInc/Polyglot/unstable-
release/requirements.txt \

&& while read line; do $line; done < custom.txt



RUN wget https://github.com/UniversalDevicesInc/Polyglot/raw/unstable-
release/bin/${binfile}-P Polyglot \

&& chown -R ${user}:${group} /home/${user} \
```

```
&& chmod 755 /home/${user}/Polyglot/${binfile}

USER ${user}

WORKDIR /home/${user}/Polyglot

ENTRYPOINT ["./polyglot.linux.x86_64.pyz","-v"]
```

Create a file named “custom.txt” and add whatever python modules and node servers you’d like installed. Here is the example I have for adding Sonos and Nest:

```
pip install soco

pip install python-nest

git clone https://github.com/Einstein42/sonos-polyglot Polyglot/config/node_servers/sonos-polyglot

git clone https://github.com/Einstein42/nest-polyglot Polyglot/config/node_servers/nest-polyglot
```

Create a folder to store your config:

```
mkdir -p ~/docker/polyglot
```

(If you already have polyglot up and running, backup your config folder and then just point this to your existing config folder instead of the one above. This should work, but always make a backup.)

The image is built by running this command (from the folder where you Dockerfile and custom.txt files are located):

```
docker build -t polyglot .
```

After that completes, you should be able to run a command like this:

```
docker run -d --name=polyglot -v ~/docker/polyglot:/home/polyglot/Polyglot/config --net=host -t polyglot-docker
```

The polyglot page should be available via your docker host machine’s IP and port 8080. If that needs to change, allow it to start, then stop the container (docker stop polyglot), edit the configuration.json in the config folder on your docker host. Then start it back up (docker start polyglot). I chose to use the non-compiled variation so that it would run

across more systems. Another item I'd like to improve is only using the compiled version in order to save space.

Ready to use image for linux x86_64

Run this command, assuming you already have docker setup and running for your user on your linux machine.

```
mkdir -p ~/docker/polyglot && docker run -d --name=polyglot -v
~/docker/polyglot:/home/polyglot/Polyglot/config --net=host -t i814u2/udi-polyglot
```

Then re-pull the image in order to update it, and re-create the container (this will also start the container, of course):

5.5.6.1.2Raspberry Pi

Use NOOBS to install Raspbian on your Raspberry Pi (see here: https://www.raspberrypi.org/documentation/installation/noobs.md)

Once the Raspberry Pi is booted, open the terminal and enter the following command:

```
curl -sSL https://get.docker.com | sh && sudo usermod -aG docker pi
```

This may take a while, depending on the speed of the machine and SD card.

Once the above command is complete, logout and log back in, or just reboot if you prefer.

Run this command to download and start the docker container (using the pre-setup image in my docker hub repository):

```
mkdir ~/docker/polyglot && docker run -d --name=polyglot -v
~/docker/polyglot:/home/polyglot/Polyglot/config --net=host -t i814u2/rpi-udi-polyglot
```

Image contains the default Kodi and Hue nodes and also the Sonos and Nest nodes. Second command updated to include the creation of the local config directory to ensure proper permissions.

If you've followed my instructions so far, and want to update, here is what you can do: (This assumes you followed the instructions and have your config files pointed to a local directory. Always make a backup of your config first, just in case):

```
docker stop polyglot

docker rm polyglot

docker pull i814u2/rpi-udi-polyglot

docker run -d --name=polyglot -v ~/docker/polyglot:/home/polyglot/Polyglot/config --net=host -t i814u2/rpi-udi-polyglot
```

(4 separate commands, one per line, each starts with the word 'docker')

Again, the last line assumes you followed the copy/paste style info in my previous posts for a RaspberryPi setup. Adjust, as needed, for your own directory structure.

Quick note: this will take a while on initial startup. It's upgrading packages when the container is started (in order to avoid the need to re-pull the image each time).

More detail on how the image is setup:

Follow the normal guides to install Raspbian on your RPi, then run this command to install Docker:

```
curl -sSL https://get.docker.com | sh
```

When that is complete, it will mention adding your user to the docker group in order to run without sudo. That is accomplished with this command:

```
sudo usermod -aG docker pi
```

That assumes you're using the "pi" user, and not your own. You'll need to log out, then log back on in order for that change to take effect.

Here is the Dockerfile info for a Raspberry Pi:

```
FROM fnphat/rpi-alpine-python:2.7



ARG binfile=polyglot.linux.armv7l.pyz

ARG user=polyglot

ARG group=polyglot

ARG uid=1000

ARG gid=1000



RUN addgroup -g ${gid} ${group} \

    && adduser -h /home/${user} -s /bin/sh -G ${group} -D -u ${uid} ${user}



WORKDIR /home/${user}

COPY custom.txt .



RUN apk add --update \

    build-base python-dev linux-headers git ca-certificates wget openssl-dev \

    && rm -rf /var/cache/apk/*



RUN pip install --upgrade pip \

  && pip install -r
https://raw.githubusercontent.com/UniversalDevicesInc/Polyglot/unstable-
release/requirements.txt \
```

```
   && while read line; do $line; done < custom.txt



RUN wget https://github.com/UniversalDevicesInc/Polyglot/raw/unstable-
release/bin/${binfile} -P Polyglot \

   && chown -R ${user}:${group} /home/${user} \

   && chmod 755 /home/${user}/Polyglot/${binfile}



USER ${user}

WORKDIR /home/${user}/Polyglot

ENTRYPOINT ["./polyglot.linux.armv7l.pyz", "-v"]
```

Use NOOBS to install Raspbian on your Raspberry Pi (see here: https://www.raspberrypi.org/documentation/installation/noobs.md)

Once the Raspberry Pi is booted, open the terminal and enter the following command:

```
curl -sSL https://get.docker.com | sh && sudo usermod -aG docker pi
```

That will take a while, depending on the speed of the machine and SD card. 3. Once the above command is complete, logout and log back in, or just reboot if you prefer. 4. Run this command to download and start the docker container (using the pre-setup image in my docker hub repository):

```
mkdir -p ~/docker/polyglot && docker run -d --name=polyglot -v
~/docker/polyglot:/home/polyglot/Polyglot/config --net=host -t i814u2/rpi-udi-polyglot
```

That image contains the default Kodi and Hue nodes and also the Sonos and Nest nodes. Re-pulling the images and restarting new containers is easily done by running these commands:

```
docker stop polyglot

docker rm polyglot
```

Then re-pull the image in order to update it, and re-create the container (this will also start the container, of course):

```
docker pull i814u2/rpi-udi-polyglot && docker run -d --name=polyglot -v
~/docker/polyglot:/home/polyglot/Polyglot/config --net=host -t i814u2/rpi-udi-polyglot
```

5.6 Polyglot Source Code

- Polyglot source code is available at: https://github.com/UniversalDevicesInc/Polyglot

5.7 Node Hints Documentation[9]

Hint specification version 0.2

A hint an optional way for a node server to provide additional information describing the nodes that it creates. A hint is composed of 4 bytes of information. This document describes a common specification for these 4 bytes so that both node servers and external applications have an understanding of the properties [drivers in Polyglot terms] associated with the nodes. The ISY itself ignores this information.

```
<hint> = <byte1> . <byte2> . <byte3> . <byte4>
```

<byte1> [0 - 0xff]

Represents the device category. A category is a high-level generic description. See list below for currently defined categories. Each category will have a specific set of mandatory drivers.

<byte2> [0 - 0xff]

Represents more specific type of device within the category. This allows for differentiation between devices of the same category. For example, to differentiate between a fire alarm and a water leak alarm.

<byte3> [0 - 0xff]

Represents device/model specific identification.

[9] (Universal Devices)

<byte4> [0 - 0xff]

Is currently undefined and can be used for node server specific information. The values/format of this byte are not described in this specification.

Categories

- 0x00 - Reserved
- 0x01 - Dimmer device
- 0x02 - Relay device
- 0x03 - Network device
- 0x04 - Irrigation device
- 0x05 - Climate control device
- 0x06 - Pool control device
- 0x07 - Sensor device

Examples

0x00000000 [0.0.0.0] Default, non-compliant.

x005010000 [5.1.0.0] This is a thermostat that has basic heat/cool setpoints.

0x02000000 [2.0.0.0] This is a relay device with two states; open/close or on/off; It has a single status value and two commands.

0x07030000 [7.3.0.0] This is a specific type (TBD) of sensor.

6 Java

6.1 ISY Java Web Services Tutorial[10]

6.1.1 How to program/consume UDI web services in Java

This will show you how to use Java API for XML - WebServices (JAX-WS) to parse the UDI web services (and UDI Elk web services) wsdl files and turn them into Java object classes.

JAX-WS has an import tool (wsimport) that reads a W3C compliant wsdl file and turns its "services" into Java objects so all you have to do is call the appropriate Java classes.

For example; once the processing is complete, you can use a Java class like the following to access the ISY:

```
package ca.bc.webarts.tools.isy;

/*

* These are all generated from the JAX-WS wsimport tool

*/

import ca.bc.webarts.tools.isy.webservices.DateTime;

import ca.bc.webarts.tools.isy.webservices.Empty;

import ca.bc.webarts.tools.isy.webservices.UDIServices;

import ca.bc.webarts.tools.isy.webservices.UDIServicesPortType;

import ca.bc.webarts.tools.isy.webservices.Variable;

import java.util.List;

import java.util.Map;

import javax.xml.ws.BindingProvider;

public class IsyWSClient
```

[10] (Universal Devices)

```
{
 private String isyUsername_ = "admin";
 private String isyPassword_ = "admin";
 private UDIServices service_ = new UDIServices();
 private UDIServicesPortType port_ = service_.getUDIServicesPort();
  public static void main(String[] args)
  {
    IsyWSClient instance = new IsyWSClient();
     Map<String, Object> reqCtx = ((BindingProvider) instance.port_).getRequestContext();
     reqCtx.put(BindingProvider.USERNAME_PROPERTY, instance.isyUsername_);
     reqCtx.put(BindingProvider.PASSWORD_PROPERTY, instance.isyPassword_);
     DateTime dt = instance.port_.getSystemDateTime(new Empty());
     System.out.println("NTP time=" + dt.getNTP());
  }
}
```

6.1.2 Online Documentation

Before I start, here are some references to some good background information and tools, specific to processing Web Services in Java.

- Oracle's Java WebServices Tutorial: https://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/1.6/tutorial/doc/index.html
- Oracle's example WebServices code: https://community.oracle.com/welcome
- TCPmon[1]- a TCP monitor (written in Java) to monitor ALL TCP packets (ie http and soap): https://code.google.com/archive/p/tcpmon/
- UDI Online WSDK Forum: https://forum.universal-devices.com/forum/44-wsdk/

6.1.3 Requirements

- UDI Webservices Development kit (UDI WSDK)
  - Use the same version as firmware in your device
- NetBeans: https://netbeans.org/
  - Includes JAX-WS (https://github.com/javaee/metro-jax-ws), Ant (http://ant.apache.org/) and all the other libraries and tools to do the processing

or

- JAX-WS or J2EE SDK (http://www.oracle.com/technetwork/java/javaee/downloads/index.html)
  - JAX-WS Documentation (https://docs.oracle.com/javase/7/docs/technotes/guides/xml/jax-ws/index.html)
  - JAX-WS implementation and tools (https://github.com/javaee/metro-jax-ws)
  - To build a client, you don't need the full J2EE SDK (http://www.oracle.com/technetwork/java/javaee/downloads/index.html), just JAX-WS (https://github.com/javaee/metro-jax-ws) to do the WSDL import processing and API.

I used NetBeans and will document the process using NetBeans.

6.1.4 Process

6.1.4.1Basic setup

Download and install NetBeans. http://www.netbeans.org

Create a new 'Java Application' Project...

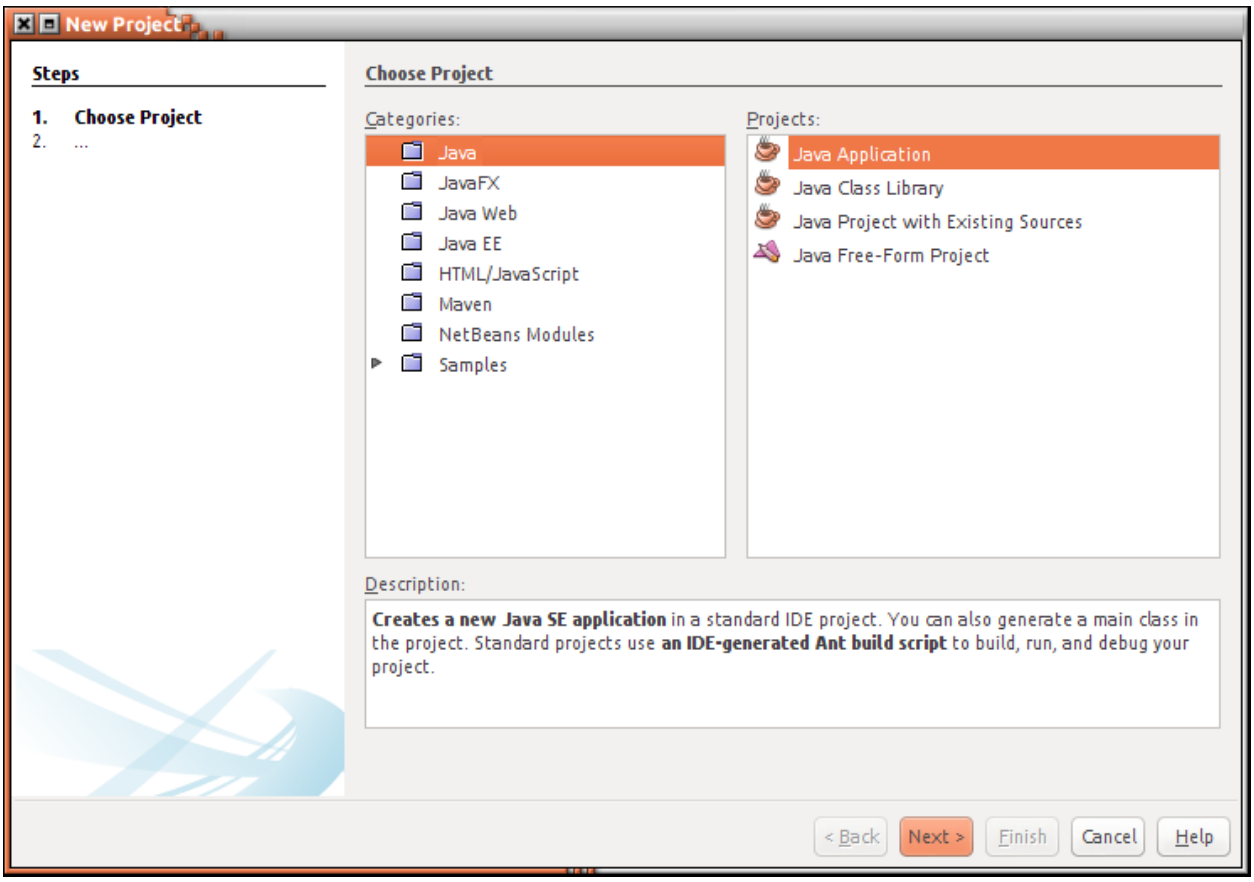


Figure 27: New Java App

Name Your Project and NEW Class...

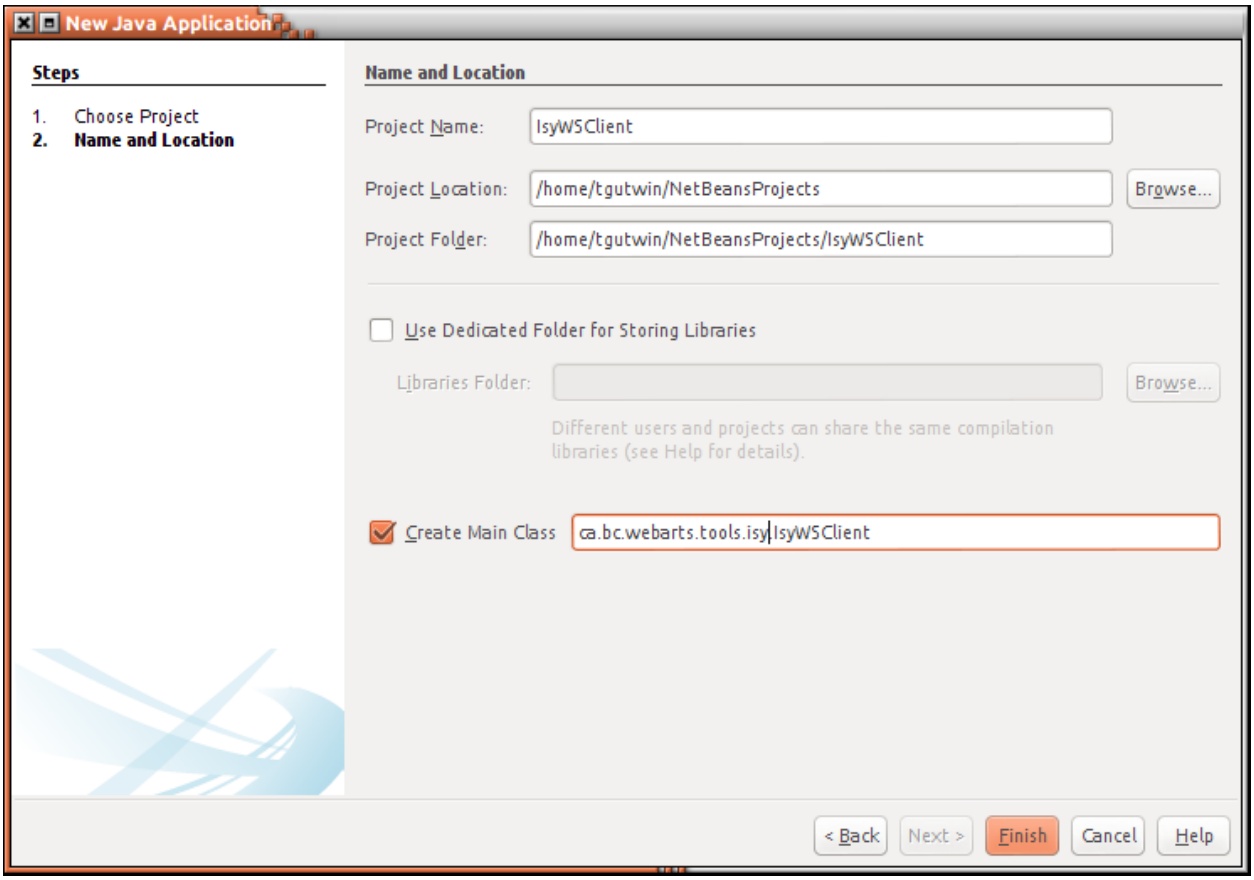


Figure 28: Name Your Project

Right-click on your new project in the projects listed along the left (to bring up the context menu) and click Properties then select Libraries to Add Library JAX-WS 1.1 ...

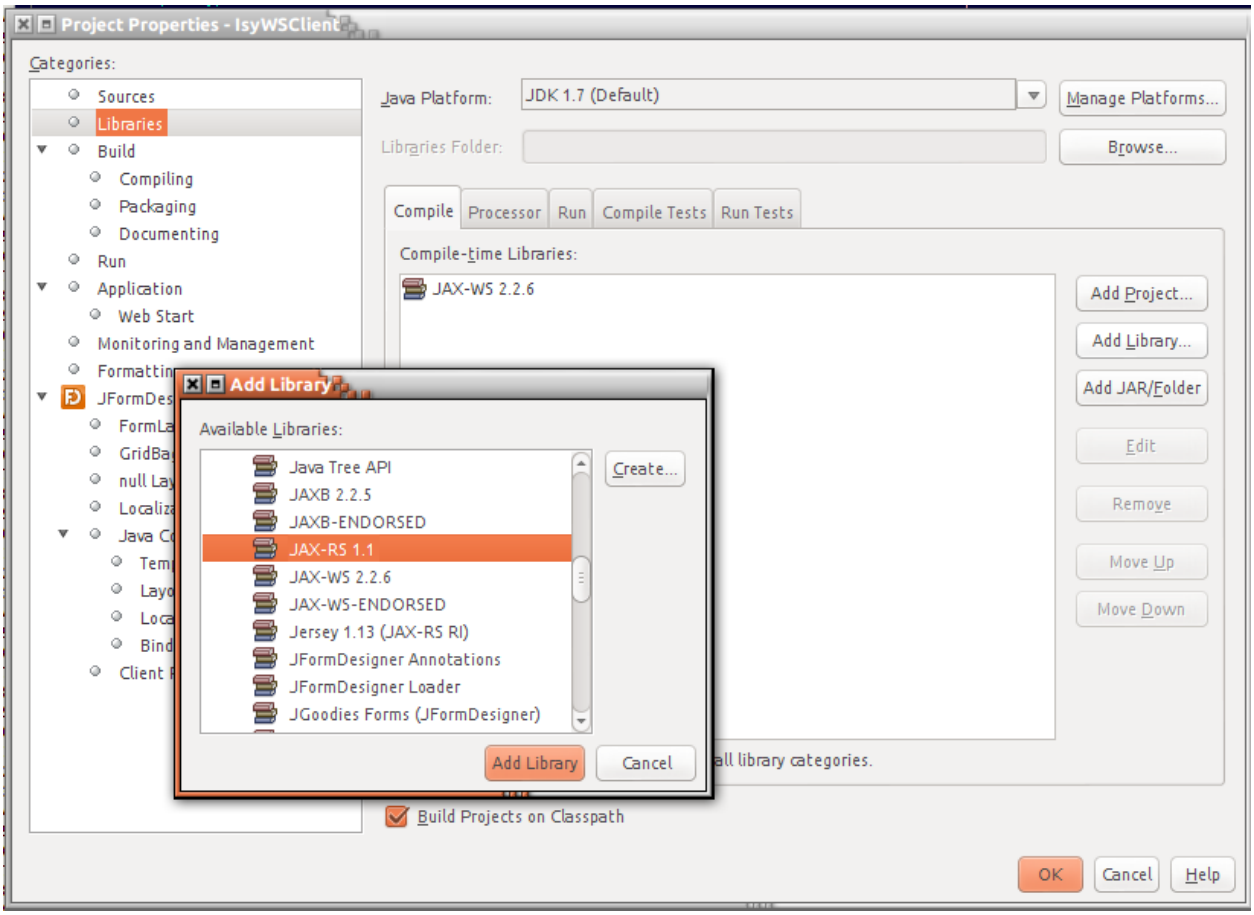


Figure 29: Add Project Library

You should now have a template Class (without a main method) in your editing window...

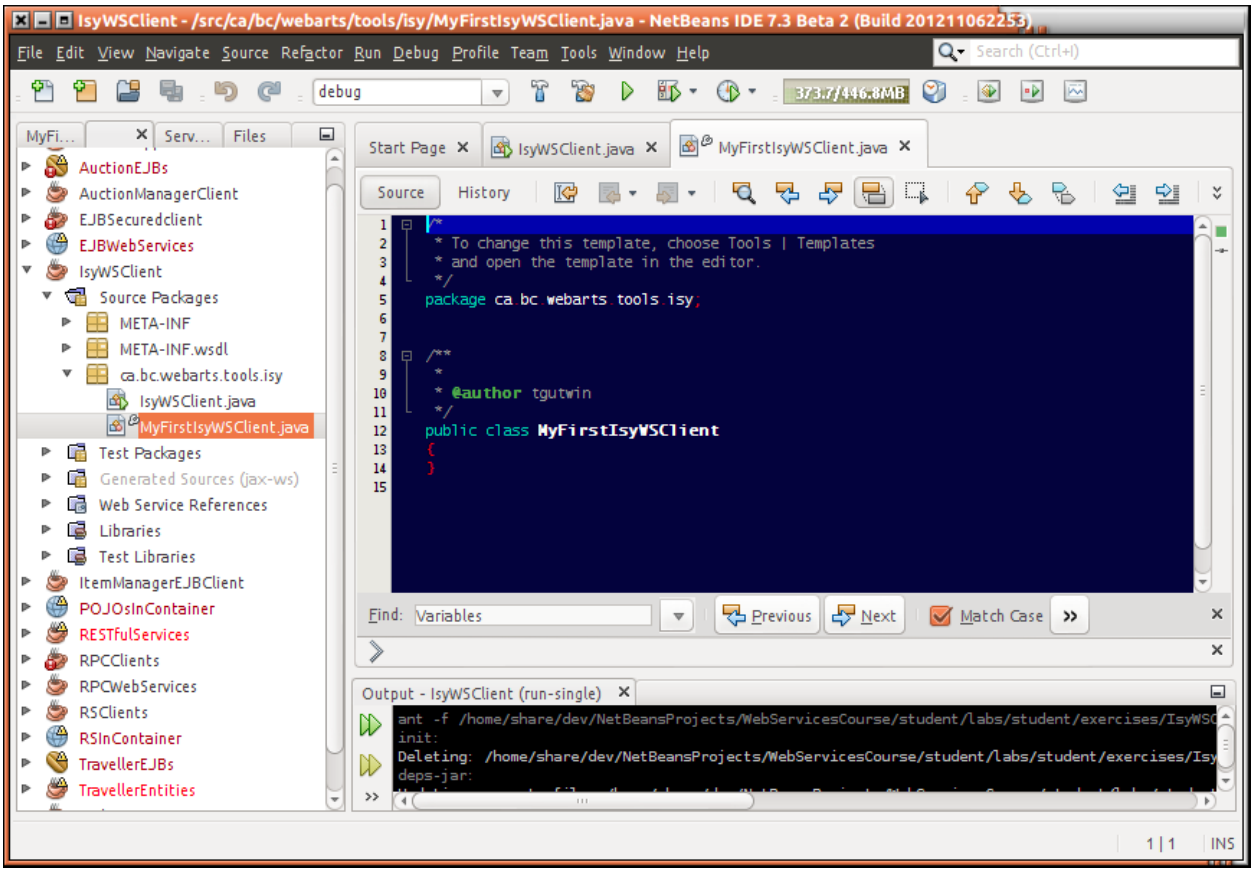


Figure 30: Template Class

Now before you start writing your Java app, we need to get the UDI Web Services loaded from the wsdl file.

6.1.5 WSDL Import

6.1.5.1Changes to the wsdl file

The 1st thing to note is that JAX-WS wsimport tool is very strict and will not process the default UDI wsdl file without a few tweaks. This is because the udi (http://isy99/services.wsdl) or the actual wsdl file udiws30.wsdl is not 100% compliant with the W3C specification for WS-I basic profile definition R2204 -A document-literal binding in a DESCRIPTION MUST refer, in each of its soapbind:body element(s), only to wsdl:part element(s) that have been defined using the element attribute. see http://www.ws-i.org/Profiles/BasicProfile-1.1.html#Bindings_and_Parts.

Many of the message parts refer to a 'type' instead of 'element' . For example...

```
<wsdl:message name="GetNodesConfigResponse">

 <wsdl:part name="nodes" type="uo:nodes"/>

</wsdl:message>
```

This is easily fixed with a small abstraction

```
<xsd:element name="nodes" type="uo:nodes"/>
```

and then use that element in the original message part...

```
<wsdl:message name="GetNodesConfigResponse">

 <wsdl:part name="nodes" element="u:nodes"/>

</wsdl:message>
```

These are simple changes.

There are ~16 times in the file. I went through and updated the udiws30.wsdl file to add the following xsd:elements:

```
<!--  RENAMED to add Type with the corresponding element that refers to it below -->

<xsd:complexType name="UDIDefaultResponseType">

  <xsd:annotation>
```

```
    <xsd:documentation>
      Default status info response
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="status" minOccurs="1" maxOccurs="1" type="xsd:string"/>
    <xsd:element name="info" minOccurs="0" maxOccurs="1" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
<!--  RENAMED to add Type with the corresponding element that refers to it below -->
<xsd:complexType name="UDITimeResponseType">
  <xsd:attribute name="val" type="xsd:dateTime" use="required">
    <xsd:annotation>
      <xsd:documentation>
        Timestamp in the form of YYYYMMDD HH:MM:SS
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
<!--  RENAMED to add Type with the corresponding element that refers to it below -->
<xsd:complexType name="UDIIntResponseType">
  <xsd:attribute name="val" type="xsd:int" use="required">
```

```
    <xsd:annotation>
      <xsd:documentation>
        Return value of type integer. Currently used for IsSubscribed
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
 </xsd:complexType>
<!-- PATCH - NEW Entries: Abstract types with the element attribute -->
  <xsd:element name="UDIDefaultResponse" type="u:UDIDefaultResponseType"/>
  <xsd:element name="UDITimeResponse" type="u:UDITimeResponseType"/>
  <xsd:element name="UDIIntResponse" type="u:UDIIntResponseType"/>
  <xsd:element name="nodes" type="uo:nodes"/>
  <xsd:element name="configuration" type="uo:configuration"/>
  <xsd:element name="SysStat" type="uo:SystemStatus"/>
  <xsd:element name="DT" type="uo:DateTime"/>
  <xsd:element name="SystemOptions" type="uo:SystemOptions"/>
  <xsd:element name="SMTPConfig" type="uo:SMTPConfiguration"/>
  <xsd:element name="DBG" type="uo:DBG"/>
  <xsd:element name="SceneProfiles" type="uo:SceneProfiles"/>
  <xsd:element name="SubscriptionResponse" type="uo:Subscription"/>
  <xsd:element name="LastError" type="uo:LastError"/>
  <xsd:element name="DDNSHost" type="uo:DDNSHost"/>
```

```
  <xsd:element name="Var" type="uo:Variable"/>

  <xsd:element name="Vars" type="uo:Variables"/>

<!-- PATCH End: Abstract types with the element attribute -->
```

Then update the message parts to refer to these new elements= instead of type= .

The following message parts with type attributes were changed:

- <wsdl:part name="response" element="u:UDIDefaultResponse"/>
- <wsdl:part name="timeResponse" element="u:UDITimeResponse"/>
- <wsdl:part name="intResponse" element="u:UDIIntResponse"/>
- <wsdl:part name="nodes" element="u:nodes"/>
- <wsdl:part name="configuration" element="u:configuration"/>
- <wsdl:part name="SysStat" element="u:SysStat"/>
- <wsdl:part name="DT" element="u:DT"/>
- <wsdl:part name="SystemOptions" element="u:SystemOptions"/>
- <wsdl:part name="SMTPConfig" element="u:SMTPConfig"/>
- <wsdl:part name="DBG" element="u:DBG"/>
- <wsdl:part name="SceneProfiles" element="u:SceneProfiles"/>
- <wsdl:part name="SubscriptionResponse" element="u:SubscriptionResponse"/>
- <wsdl:part name="LastError" element="u:LastError"/>
- <wsdl:part name="DDNSHost" element="u:DDNSHost"/>
- <wsdl:part name="Var" element="u:Var"/>
- <wsdl:part name="Vars" element="u:Vars"/>

This process should work on any version of the wsdl http://isy99/services.wsdl . Note: the version of the wsdl that you get from your ISY is a small version that imports the main wsdl and defines the service and port. For ease, I suggest downloading the UDI WSDK and work on the local files contained in it.

- A patched wsdl is attached udiws30_patched.wsdl
- If you want, do a diff between them to be clear what changes were made

6.1.5.2Import the updated udiws30_patched.wsdl file

Right click on the NetBeans project and click 'New Web Service Client'

- This will ask you for the location of the wsdl file
- It will also ask for a package to put the resulting code
    - I used ca.bc.webarts.tools.isy.webservices

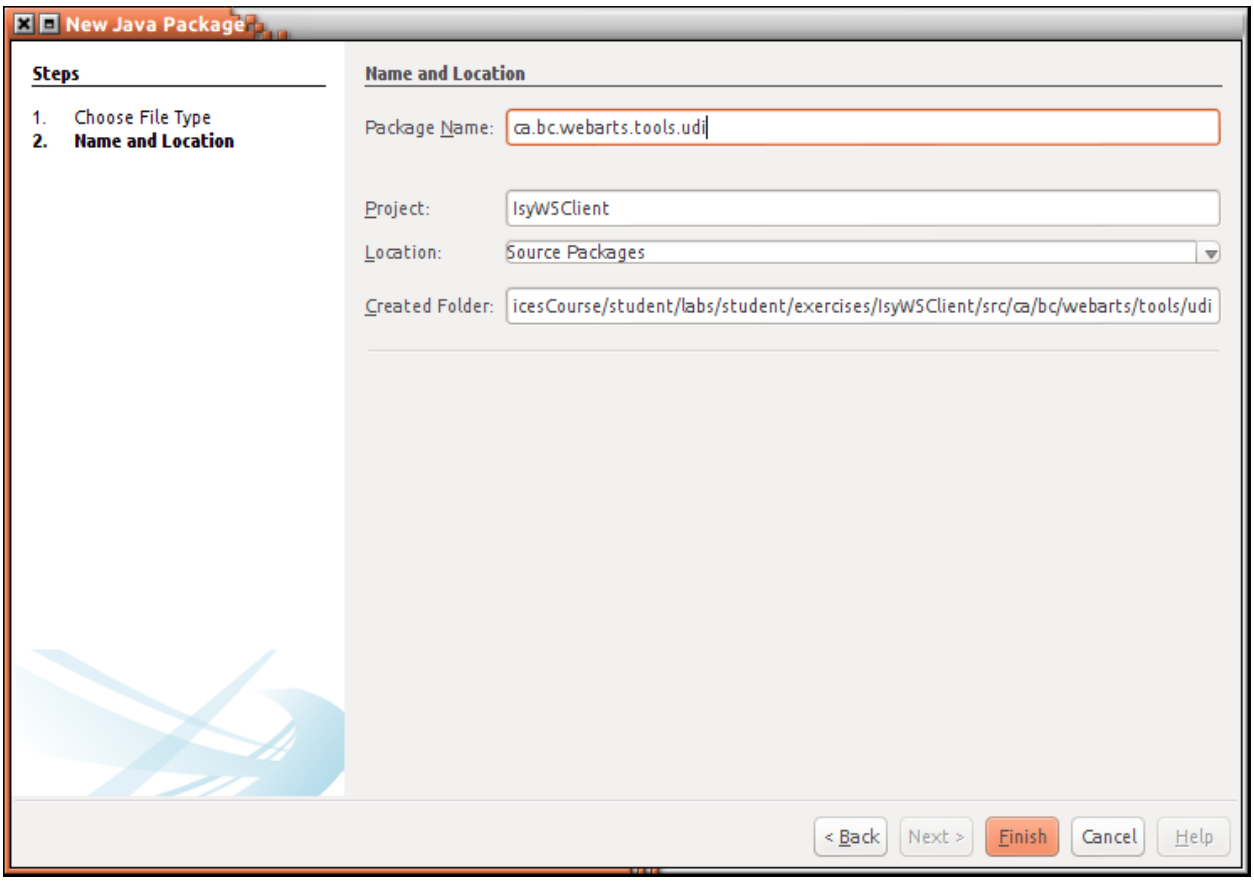


Figure 31: New Package

NetBeans will call the wsimport tool and automatically generate a bunch of new classes - one for each UDIServices service. You can then use/call them in your Java application.

See them along the left under Web Services References. See the Grey menu item named Generated Sources (jax-ws) - this is where all the generated Java classes reside.

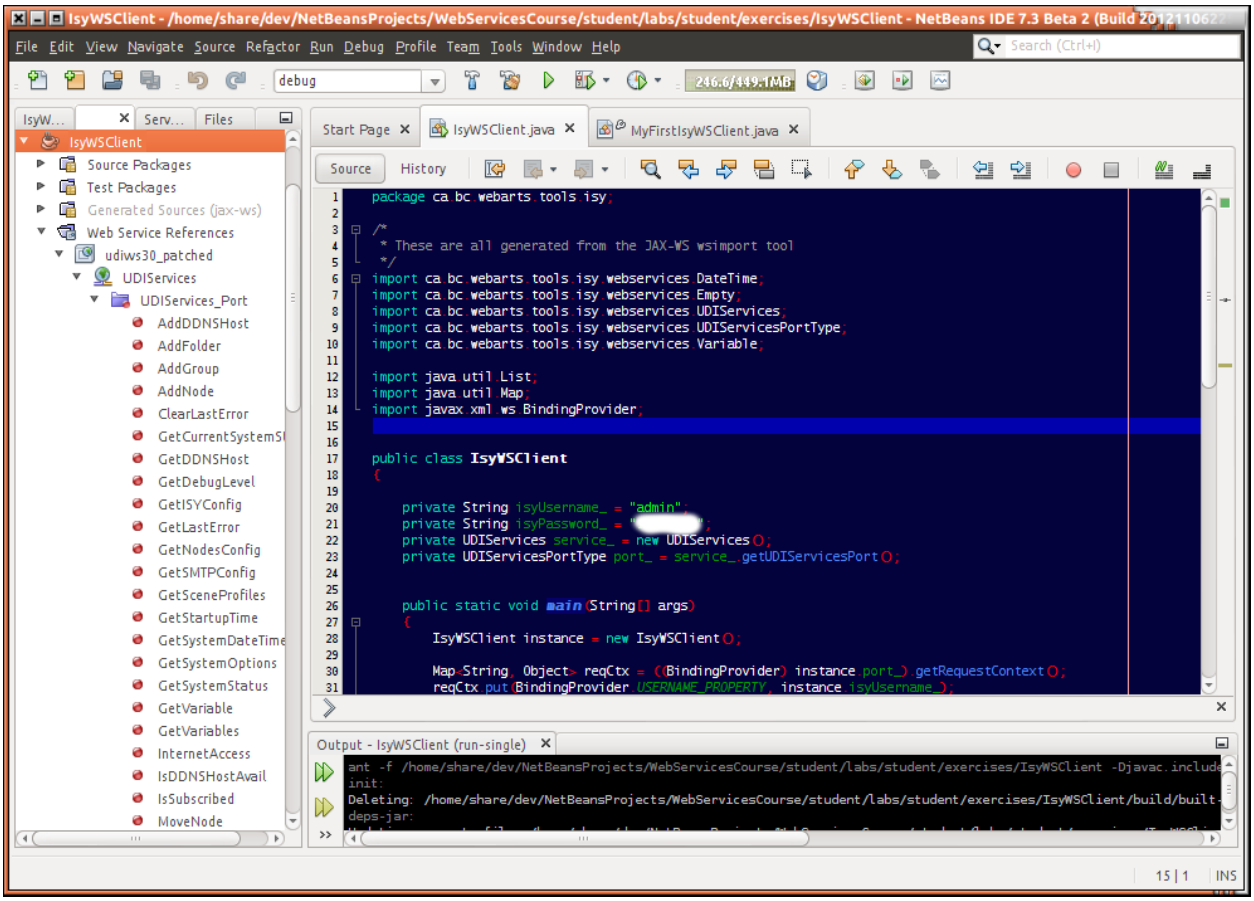


Figure 32: New Parsed Services

6.1.5.3Import the ELK WSDL file

The Elk web services definition file imports without any of these type/element errors!

Right click on the NetBeans project and click New Web Service Client and point it to the file.

udielkws1.wsdl

6.1.6 Write your Java Application

Now you can go back and write your code. Click on your Java app file in the Project/Source Packages . It should show an empty class. Create a main method.

6.1.6.1Create the Service And Port

All of the services get called from the UDIServices and UDIServicesPortType so I created a class objects for these.

```
import ca.bc.webarts.tools.isy.webservices.UDIServices;

import ca.bc.webarts.tools.isy.webservices.UDIServicesPortType;

...

UDIServices service_ = new UDIServices(); // this classname originally comes from the wsdl file

UDIServicesPortType port_ = service_.getUDIServicesPort();
```

Note: your import package names will be different than my example.

6.1.6.2HTTP Basic Authentication

The UDI ISY expects BASIC HTTP authentication that gets inserted into the HTTP header. This is easy in Java by getting the session context and adding the username/password to it.

```
private String isyUsername_ = "admin";

private String isyPassword_ = "yourPasswordHere";

...

/* These calls deal with hashing them into Base64 and putting them into the http header */

Map<String, Object> reqCtx = ((BindingProvider) instance.port_).getRequestContext();

reqCtx.put(BindingProvider.USERNAME_PROPERTY, instance.isyUsername_);

reqCtx.put(BindingProvider.PASSWORD_PROPERTY, instance.isyPassword_);
```

6.1.6.3Call Web Services

Netbeans has an easy code create feature that automatically creates the code to call your new web services.

In the editor window, right click and then click on Insert Code... and select call web service operation. a list off all the services come up, you choose which one and it will create the code to call it.

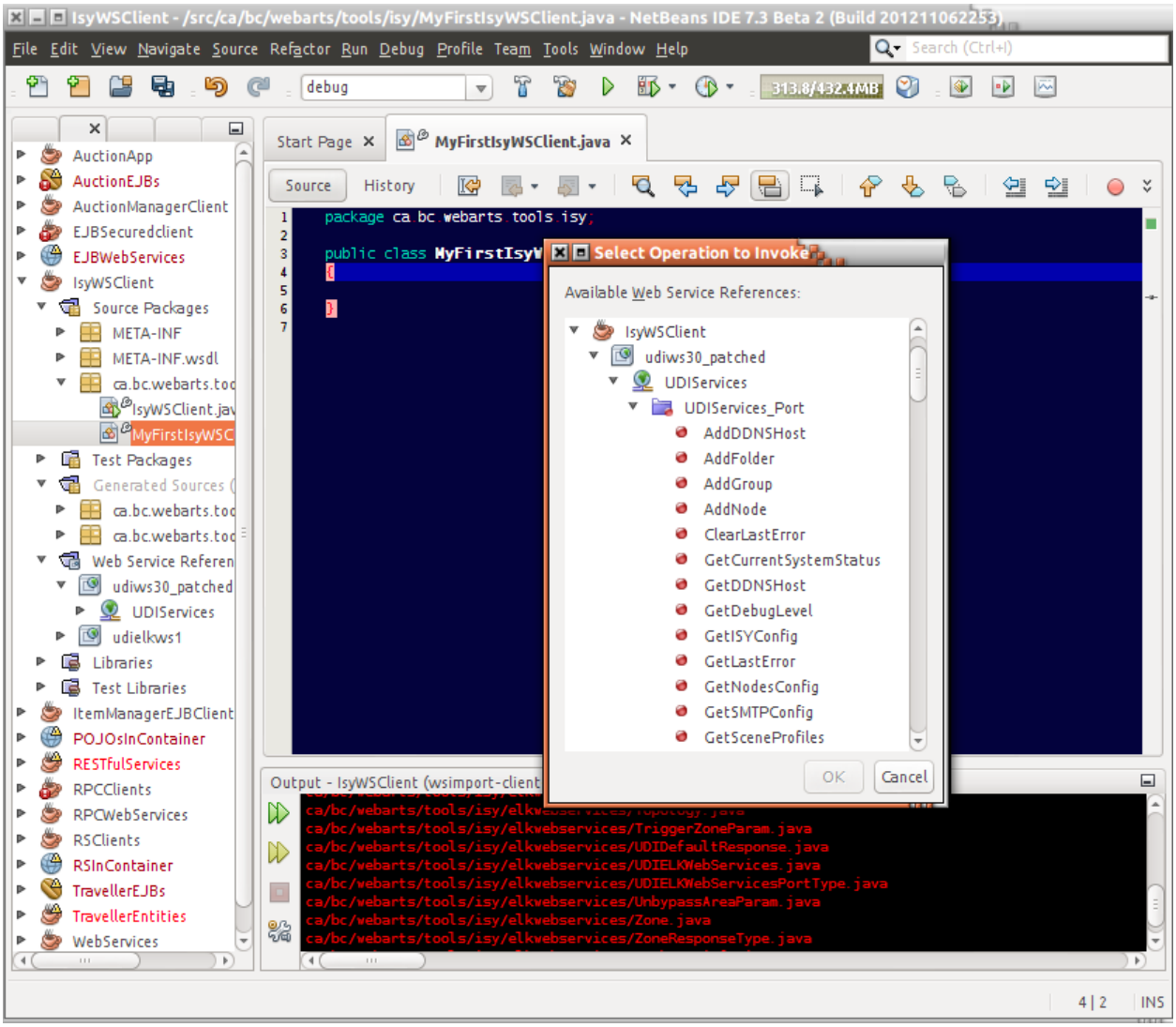


Figure 33: Insert Code Web Services

Now, the rest is the easy part because all the WebServices have been wrapped in Java methods. For example, to get the ISY Date and Time...

```
import ca.bc.webarts.tools.isy.webservices.DateTime;

import ca.bc.webarts.tools.isy.webservices.Empty;

 ...

DateTime dt = port_.getSystemDateTime(new Empty());

System.out.println("NTP time=" + dt.getNTP());
```

6.1.7 Sample Java App

Here is my full (very simple) UDI/ISY Web Services Client.

```
package ca.bc.webarts.tools.isy;

/* These are all generated from the JAX-WS wsimport tool */

import ca.bc.webarts.tools.isy.webservices.DateTime;

import ca.bc.webarts.tools.isy.webservices.Empty;

import ca.bc.webarts.tools.isy.webservices.UDIServices;

import ca.bc.webarts.tools.isy.webservices.UDIServicesPortType;

import ca.bc.webarts.tools.isy.webservices.Variable;

import java.util.List;

import java.util.Map;

import javax.xml.ws.BindingProvider;

public class IsyWSClient

{

  private String isyUsername_ = "admin";

  private String isyPassword_ = "admin";

  private UDIServices service_ = new UDIServices();
```

```
  private UDIServicesPortType port_ = service_.getUDIServicesPort();

   public static void main(String[] args)

   {

      IsyWSClient instance = new IsyWSClient();

      Map<String, Object> reqCtx = ((BindingProvider) instance.port_).getRequestContext();

      reqCtx.put(BindingProvider.USERNAME_PROPERTY, instance.isyUsername_);

      reqCtx.put(BindingProvider.PASSWORD_PROPERTY, instance.isyPassword_);

      DateTime dt = instance.getSystemDateTime();

      System.out.println("NTP time=" + dt.getNTP());

      //List<Variable> isyVars = instance.getVariables(1);// 1=Integer Variable 2=State
Variable

      //for (Variable variable : isyVars)

      //{

     //  System.out.println(variable.getId() + "=" + variable.getVal());

      //}

   }

   /**

    * gets a list of isy variables by type spec'd.

    * @param type 1=IntegerVariable 2=StateVariable

    * @return a list of vars

    */

   private List<Variable> getVariables(int type)
```

```
  {
    return port_.getVariables(type);
  }
  /**
   * Gets the ISY DateTime object.
   * @return the ISY DateTime object
   */
  private DateTime getSystemDateTime()
  {
    return port_.getSystemDateTime(new Empty());
 }
}
```

6.1.8 Debug SOAP Messages With TCPMon

If you want to watch the SOAP messages that are going-to --> and <--coming-back from your ISY, you can use a small (free) Java program called TCPMon.

It acts as a proxy. You point your Web Services app Endpoint Address at TCPMon and TCPMon echoes it to the screen and redirects it onto your ISY. To change your endpoint address, do either:

- edit the WSDL file to point at the TCPMon localPort such as: http://localhost:8080 and re-import it the WSDL to have it regenerate the code

OR

- Update the existing Service class with a new endpoint address
    - URL newServiceURL = new URL("http://localhost:8080/services");
    - UDIServices tcpMonService = new UDIServices(newServiceURL);
    - UDIServicesPortType tcpMonPort = tcpMonService.getUDIServicesPort();

- see http://stackoverflow.com/questions/2046790/change-webservice-endpoint-address-at-run-time

You can even manually type or edit the SOAP message and send it direct to your ISY.

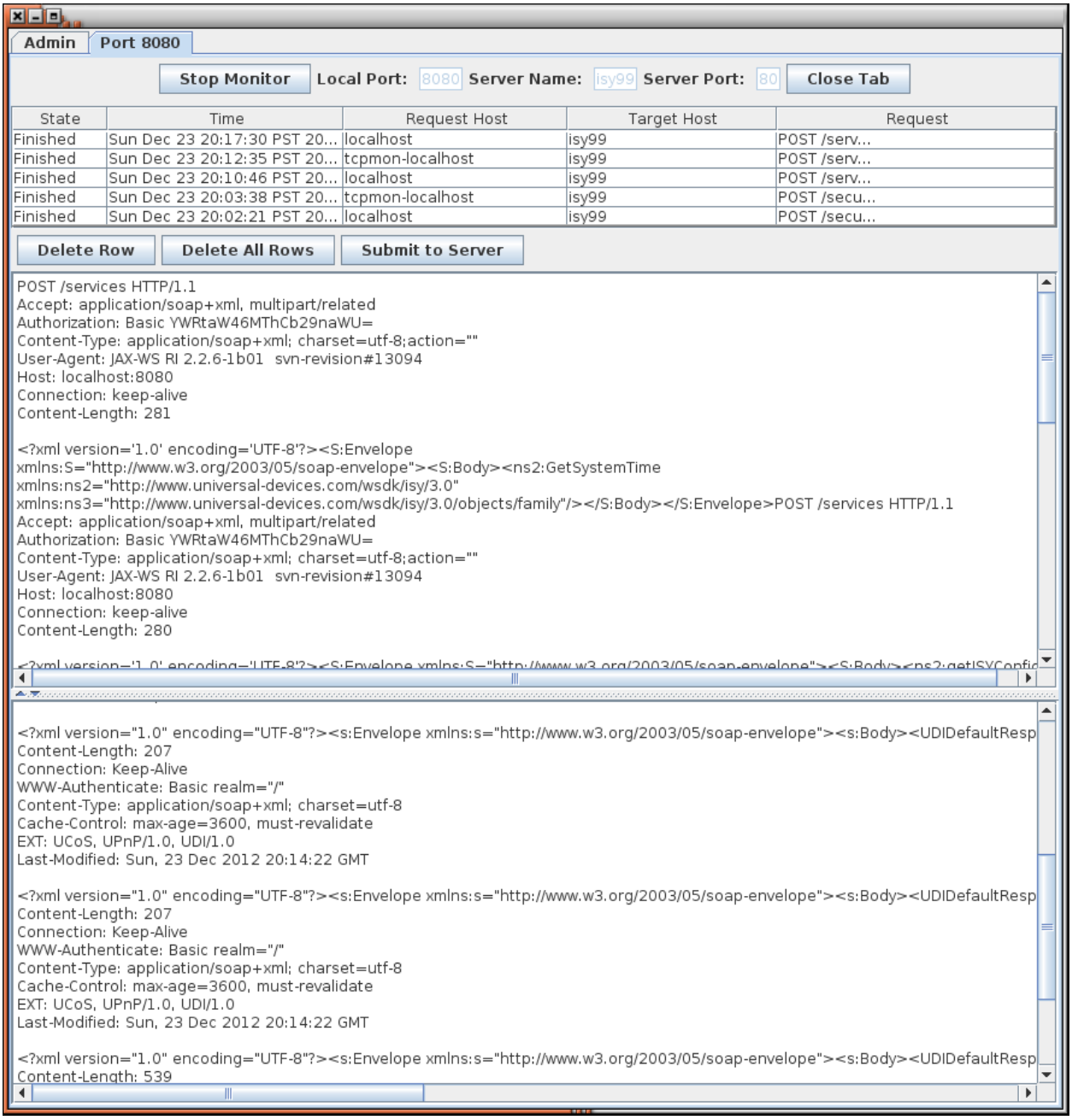


Figure 34: TCPMon

6.2 Java REST Services Requester Example[11]

Simple/Basic Java REST Web Services Client (with source)

This page presents the Java code for a basic REST Web services client. It also shows how to use it to wrap around ISY REST services, including the HTTP username/password Authentication.

It is kept as small and lightweight as possible but includes all the source code to get a working authenticated service connection with a restful web services server. Think of it as the "hello world"/getting started how-to Java code to be re-used and extended to wrap around the full processing of any published REST web services - specifically ISY REST services!

6.2.1 Background

If you are writing a Java application that needs to interact with the UDI ISY, you have a few robust options;

1. its full and robust Java SDK,
2. use the WebServices interface ISY-WSDK, or to keep it as light as possible
3. use the REST Interface services.

The author of this article was writing an Android app and wanted to keep it as small and lean as possible - without the inclusion of the UDI SDK Java libraries (https://www.universal-devices.com/isy-developers/). Using REST was perfect for this. This page shows the code to make a simple, but fully functional, authenticating REST client to access the ISY services. It is self-functioning, from a commandline, or can be used/included within a different Java application by instantiating the ISYRestRequester class and then making a REST call.

```
ISYRestRequester instance = new ISYRestRequester("baseUrl", "userName",
"userPasswd");

StringBuilder resp =  instance.serviceGet("/sys");

System.out.println(resp.toString()); System.out.println();
```

The full documentation is at http://links.webarts.ca/isyRestRequester

[11] (Gutwin)

6.2.2 Scope of this REST Client

- use as few external libraries as possible - keep it to the standard Java Library
    - unfortunately, Authentication requires the use of Base64 encoding so I used an Apache library for this
- provide a extendable class with the basic code to connect, send service requests, and receive responses
- has a basic set of exposed methods to setup and use published Rest Web Services
- handles Authentication to the service, if it is needed
- has a main method to allow testing or calling services from the commandline
- configurable server url
- handles all the connections to the web services URL
- POST requests or GET requests
- Minimal or no processing of the responses other than returning the results as a StringBuilder (and dumping to System.out when run from the commandLine)

6.2.3 Java Source Code

The source code is inline below and free to use and modify as needed. The latest version is also available at the authors Subversion repo viewer at http://svn.webarts.bc.ca/astroVersion in the WebARTS open repo: http://svn.webarts.bc.ca/astroVersion/jsp/goto.jsp?page=DirectoryList&repo=svn://svn.webarts.bc.ca/open&dir=/trunk/projects/WebARTS

- ca.bc.webarts.tools.RestRequester
- ca.bc.webarts.tools.isy.ISYRestRequester

The source is also contained in the isyRest.jar file at the bottom of this page.

Class Diagram

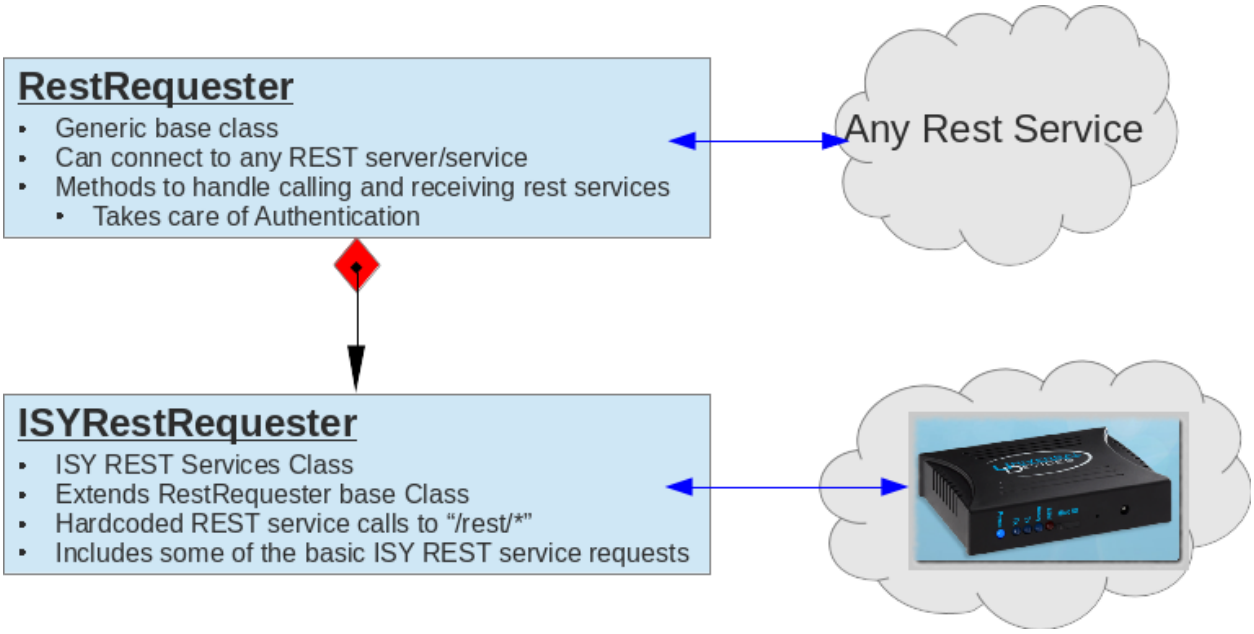


Figure 35: Class Diagram

6.2.3.1Extending this class to wrap around the ISY REST services

The generic class works out of the box on any configured Server URL, however, it is easier and cleaner to create a new class that extends the above base class with specific REST service parameters specific to the ISY services, such as

- ISY REST Service Base URL (for example http://isy994i/rest)
- Authentication username and password
- control the expected response type - plain XML or JSON
- response processing - pull out the ISY specific information from the XML responses and transcode it into whatever
- wrap individual service calls into callable Java methods that "do something" with the responses

6.2.3.2HTTP Authentication

This authentication is handled by the RestRequester class - callService method. It simply encodes the username and password with a Base64 encoder and puts it into the HTTP header. See the full base class code below for all the details.

Here is a simplified summary of what is done (in Java) to do the Authentication and send a REST Request:

```
String usrlStr = (baseUrl_+serviceName).replace(" " ,"%20");  // make sure any spaces are urlEncoded

URL url = new URL(usrlStr);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

conn.setRequestProperty("Accept", "application/xml");

/* The http header String needs to be in the following format */

String userpassword = username_ + ":" + password_;

/* Here are your options for what Base64 encoder to use */

//BASE64Encoder enc = new sun.misc.BASE64Encoder();  //  deprecated

//String encodedAuthorization = android.util.Base64.encodeToString( userpassword.getBytes(), android.util.Base64.DEFAULT );

// Encode the User/pass

String encodedAuthorization = new String(org.apache.commons.codec.binary.Base64.encodeBase64( (userpassword.getBytes()) ));

// Embed the Base64 encoded Authorization Header Property

conn.setRequestProperty("Authorization", "Basic "+ encodedAuthorization);

// Make the request and get the response

if (conn.getResponseCode() == 200)   // SUCCESS

{

  BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

```
 if (br!=null)

 {

  // do something with the Rest Response!

 }

}

conn.disconnect();

// Thats it!
```

6.2.4 Dependencies - Apache Commons Codec

The code depends (NOT Derivative) on a few Java files from the Apache Commons Codec library to perform Base64 encoding required by the http basic authentication. These files are licensed and re-distributed unmodified, as links below, under the Apache License, Version 2.0 - http://www.apache.org/licenses/LICENSE-2.0.

Everything is in the isyRest.jar file at the bottom of the page. so you don't have to download the files individually.

- org.apache.commons.codec.binary.Base64
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/binary/Base64.html
- org.apache.commons.codec.binary.BaseNCodec
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/binary/BaseNCodec.html
- org.apache.commons.codec.binary.StringUtils
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/binary/StringUtils.html
- org.apache.commons.codec.Decoder
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/Decoder.html
- org.apache.commons.codec.Charsets
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/Charsets.html
- org.apache.commons.codec.CharEncoding
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/CharEncoding.html

- org.apache.commons.codec.BinaryDecoder
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/BinaryDecoder.html
- org.apache.commons.codec.Encoder
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/Encoder.html
- org.apache.commons.codec.BinaryEncoder
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/BinaryEncoder.html
- org.apache.commons.codec.DecoderException
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/DecoderException.html
- org.apache.commons.codec.EncoderException
  - http://commons.apache.org/proper/commons-codec/apidocs/src-html/org/apache/commons/codec/EncoderException.html

6.2.4.1Using Android Base64 Instead

If you plan on using this (or other REST client) within an Android app, you can use the Android Base64 class in the Android SDK that will handle the Encoding instead of the Apache Commons Codec library files. I commented out the import android.util.Base64 in the source. I tried it it works fine as well.

6.2.5 Base Class - RestRequester.java

```
/*

*

*  Written by Tom Gutwin - WebARTS Design.

*  Copyright (C) 2014 WebARTS Design, North Vancouver Canada

*  http://www.webarts.ca

*

*  This program is free software; you can redistribute it and/or modify

*  it.

*
```

```
*  This program is distributed in the hope that it will be useful,

*  but WITHOUT ANY WARRANTY; without_ even the implied warranty of

*  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

*/

package ca.bc.webarts.tools;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.File;

import java.lang.StringBuilder;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLEncoder;

//import android.util.Base64;

import org.apache.commons.codec.binary.Base64;

/**

* A class to encapsulate the calls to Restful Web Services. It is kept very basic with low overhead to live in android apps.

* It might be best to extend this class with your specific REST servers details.

**/

public class RestRequester
```

```
{
 protected static String CLASSNAME = "ca.bc.webarts.tools.RestRequester";
 public static final String LOG_TAG = CLASSNAME;
 public static boolean debugOut_ = false;
 /**  A holder for this clients System File Separator.  */
 public final static String SYSTEM_FILE_SEPERATOR = File.separator;
 /**  A holder for this clients System line termination separator.  */
 public final static String SYSTEM_LINE_SEPERATOR =
                              System.getProperty("line.separator");
 protected static String baseUrl_ = ""; // http://isy994
 public static boolean authenticating_ = true;
 protected static String username_ = "";
 protected static String password_ = "";
 protected static boolean acceptJSON_ = false;
 public void setUsername(String uName){username_=uName;}
 public void setPassword(String uPasswd){password_=uPasswd;}
 public void setBaseUrl(String url){baseUrl_=url;}
 public void setAcceptJSON(boolean acceptJson){acceptJSON_=acceptJson;}
 public String getUsername(){return username_;}
 public String getPassword(){return password_;}
 public String getBaseUrl(){return baseUrl_;}
 public boolean getAcceptJSON(){return acceptJSON_;}
```

```
public RestRequester()

{

}

public RestRequester(String baseUrl)

{

  setBaseUrl( baseUrl);

  authenticating_=false;

}

/**

 * Constructor to take the service BASE url and authenticate with the passed userName
and Password.

 **/

public RestRequester(String baseUrl,String uName,String uPasswd)

{

  setBaseUrl( baseUrl);

  authenticating_=true;

  setUsername( uName);

  setPassword( uPasswd);

}

public boolean isInit()

{

  boolean retVal = true;
```

```
  if( baseUrl_.equals("") ||
     (authenticating_ &&
        (username_.equals("") || password_.equals(""))
     )
    )
    retVal=false;
  return retVal;
}
 /** Sends the rest service GET request off and retruns the results.
   * @param serviceName is the service (string) to append to the baseURL - example
/rest/sys
   * @return the serviceResult as a stringBuilder, null if error
   **/
 public StringBuilder serviceGet(String  serviceName)
 { return callService(serviceName, true);}
 /** Sends the rest service POST request off and retruns the results.
   * @param serviceName is the service (string) to append to the baseURL - example
/rest/sys
   * @return the serviceResult as a stringBuilder, null if error
   **/
 public StringBuilder servicePost(String  serviceName)
 { return callService(serviceName, false);}
 /** Sends the rest service request off and retruns the results.
```

```
   * @param serviceName is the service (string) to append to the baseURL - example
/rest/sys

   * @param getNotPost is a flag to tell this method to do a get or post based on this flag -
true does a GET, false does a POST

   * @return the serviceResult as a stringBuilder, null if error

  **/

  public StringBuilder callService(String  serviceName, boolean getNotPost)

  {

   StringBuilder retVal = null;

   if(isInit())

    try

    {

     String usrlStr = (baseUrl_+serviceName).replace(" " ,"%20");

     URL url = new URL(usrlStr);

     HttpURLConnection conn = (HttpURLConnection) url.openConnection();

     if(getNotPost)

      conn.setRequestMethod("GET");

     else

      conn.setRequestMethod("POST");

     if(acceptJSON_)

      conn.setRequestProperty("Accept", "application/json");

     else

      conn.setRequestProperty("Accept", "application/xml");
```

```
      //BASE64Encoder enc = new sun.misc.BASE64Encoder();

      String userpassword = username_ + ":" + password_;

      //String encodedAuthorization = android.util.Base64.encodeToString(
userpassword.getBytes(), android.util.Base64.DEFAULT );

      String encodedAuthorization = new String(Base64.encodeBase64(
(userpassword.getBytes()) ));

      conn.setRequestProperty("Authorization", "Basic "+ encodedAuthorization);

      if (conn.getResponseCode() == 200)

      {

        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

        if (br!=null)

        {

          retVal = new StringBuilder();

          String output;

          if (debugOut_) System.out.println("Output from Server .... \n");

          while ((output = br.readLine()) != null)

          {

            if (debugOut_) System.out.println(output);

            retVal.append(output);

            retVal.append("\n");

          }

        }
```

```
    } // valid http response code
     conn.disconnect();
   }
   catch (MalformedURLException e)
   {
    e.printStackTrace();
   }
   catch (IOException e)
   {
     e.printStackTrace();
   }
  return retVal;
}
public StringBuilder responseIndenter(StringBuilder sb)
{
  StringBuilder retVal = new StringBuilder("");
  int indent = -1;
  boolean opening = false;
  boolean closing = false;
  boolean lf = false;
  char [] sbChar = sb.toString().toCharArray();
  for (int i=0; i< sbChar.length;i++)
```

```
{
 opening = false;
 closing = false;
 lf = false;
 if(sbChar[i]=='<')
 {
  indent++;
  opening = true;
  retVal.append("\n");
  for (int j=0;j<indent;j++) retVal.append("  ");
  retVal.append("<");
 }
 else if (sbChar[i]=='/'&&sbChar[i-1]=='<')
 {
  indent--;indent--;
  closing = true;
  retVal.append("/");
 }
 else if (sbChar[i]=='>')
 {
  lf = true;
  retVal.append(">\n");
```

```
      for (int j=0;j<indent;j++) retVal.append("  ");
    }
    else
    {
      retVal.append(sbChar[i]);
    }
  }
  return retVal;
 }
}
```

6.2.6 ISY-994 Extension Class - ISYRestRequester.java

```
/*
 *
 * Written by Tom Gutwin - WebARTS Design.
 * Copyright (C) 2014 WebARTS Design, North Vancouver Canada
 * http://www.webarts.ca
 *
 * This program is free software; you can redistribute it and/or modify
 * it.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without_ even the implied warranty of
```

```
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 */
package ca.bc.webarts.tools.isy;
import ca.bc.webarts.tools.RestRequester;
public class ISYRestRequester extends RestRequester
{
 protected static String CLASSNAME = "ca.bc.webarts.tools.isy.ISYRestRequester";
 private static StringBuffer helpMsg_ = new StringBuffer(SYSTEM_LINE_SEPERATOR);
 /** Constructor to over-ride the base class with the ISY specifics **/
 public ISYRestRequester()
 {
  setBaseUrl( "http://isy994/rest");  //  <<<---  Add your ISY IP URL here
  authenticating_=true;
  setUsername( "admin");           //  <<<---  Add your ISY userID here
  setPassword( "*******");         //  <<<---  Add your ISY password here
 }
 /**
  * Class main commandLine entry method.
  **/
 public static void main(String [] args)
 {
  final String methodName = CLASSNAME + ": main()";
```

```
   ISYRestRequester instance = new ISYRestRequester();

   // could also instatiate with the RestRequester(String baseUrl,String uName,String
uPasswd)

   /* Simple way af parsing the args */

   if (args ==null || args.length<1)

    System.out.println(getHelpMsgStr());

   else

   {

    if (args[0].equals("test"))

    {

     System.out.println("Testing ISY Rest Service: "+ "/sys");

     StringBuilder resp =  instance.serviceGet("/sys");

     System.out.println(resp.toString());

     System.out.println();

    }

    else

    {

     // Parse the command

     String allcommands = args[0];

     for (int i=1;i< args.length;i++) allcommands+=" "+args[i];

     System.out.print("Sending ISY Rest Service: "+allcommands);

     String passedCommand = (allcommands.startsWith("/rest/") ?
allcommands.substring(5) : allcommands);
```

```
    System.out.println(" ("+passedCommand+")");
    StringBuilder resp =  instance.serviceGet(passedCommand);
    if (resp!=null)
    {
      System.out.println(instance.responseIndenter(resp).toString());
      System.out.println();
    }
    else
    {
      System.out.println("Response Error");
      System.out.println();
    }
   }
  }
} // main
  /** gets the help as a String.
 * @return the helpMsg in String form
 **/
private static String getHelpMsgStr() {return getHelpMsg().toString();}
/** initializes and gets the helpMsg_
class var.
 * @return the class var helpMsg_
```

```
 **/
 private static StringBuffer getHelpMsg()
 {
  helpMsg_ = new StringBuffer(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("---  WebARTS ISYRestRequester Class  ----------------------------------------
-------------");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("---  $Revision:$ $Date:$ ---");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("-------------------------------------------------------------------------------");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("WebARTS ca.bc.webarts.tools.isy.ISYRestRequester Class");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("SYNTAX:");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("   java ");
  helpMsg_.append(CLASSNAME);
  helpMsg_.append(" test or restCommand");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
  helpMsg_.append("Available Commands:");
  helpMsg_.append(SYSTEM_LINE_SEPERATOR);
```

```
   helpMsg_.append("see: http://wiki.universal-
devices.com/index.php?title=ISY_Developers:API:REST_Interface");

   helpMsg_.append(SYSTEM_LINE_SEPERATOR);

   helpMsg_.append("-----------------------------------------------------");

   helpMsg_.append("----------------------");

   helpMsg_.append(SYSTEM_LINE_SEPERATOR);

   return helpMsg_;

 }

}
```

6.2.7 isyRest.jar - source and classes

Both classes and source are included in the following:

isyRest.jar: http://links.webarts.ca/isyRestRequester/isyRest.jar?attredirects=0&d=1

Unzip it, make changes (isyURL/user/pass), javac and away you go!

You can also run this directly from the jar:

```
   java -jar isyRest.jar  /sys
```

7 Rest

7.1 ISY REST Interface[12]

7.1.1 Notes

- All calls use the HTTP GET method (Except for uploading configuration files)
- Not all features are implemented in the Rest API, for now, some operations are only available via the SOAP API

7.1.2 Return Values / Codes

- All calls will return the requested data (if there is any) or Boolean status for the command

Successful :

```
<RestResponse succeeded="true">

  <status>200</status>

</RestResponse>
```

Error :

```
<RestResponse succeeded="false">

  <status>404</status>

</RestResponse>
```

7.1.3 Configuration

/rest/config

returns the configuration of the system with permissible commands

See also Example config output: https://wiki.universal-devices.com/index.php?title=ISY_Developers:API:REST_Interface:config

[12] (Universal Devices)

/rest/sys

Returns system configuration

Example :

```
<SystemOptions>
 <MailTo/>
 <HTMLRole>3</HTMLRole>
 <CompactEmail>false</CompactEmail>
 <QueryOnInit>true</QueryOnInit>
 <PCatchUp>true</PCatchUp>
 <PGracePeriod>900</PGracePeriod>
 <WaitBusyReading>true</WaitBusyReading>
 <NTPHost>0.north-america.pool.ntp.org</NTPHost>
 <NTPActive>true</NTPActive>
 <NTPEnabled>true</NTPEnabled>
 <NTPInterval>43200</NTPInterval>
</SystemOptions>
```

/rest/time

Returns system time

Example :

```
<DT>
 <NTP>3578531154</NTP>
 <TMZOffset>-28800</TMZOffset>
 <DST>true</DST>
 <Lat>37.766701</Lat>
 <Long>122.416702</Long>
 <Sunrise>3578536351</Sunrise>
 <Sunset>3578588453</Sunset>
 <IsMilitary>true</IsMilitary>
</DT>
```

/rest/network

Returns network configuration

Example :

```
<NetworkConfig>
 <Interface isDHCP="true">
  <ip>10.1.2.36</ip>
 </Interface>
 <WebServer>
  <httpPort>80</httpPort>
  <httpsPort>443</httpsPort>
 </WebServer>
</NetworkConfig>
```

/rest/subscriptions

Returns the state of subscriptions

Example :

```
<Subscriptions>

 <Sub isExpired="yes" isPortal="no" sid="-1" sock="-1" isReusingSocket="no"
isConnecting="no"/>

 <Sub isExpired="no" isPortal="no" sid="40" sock="29" isReusingSocket="yes"
isConnecting="no"/>

 <Sub isExpired="yes" isPortal="no" sid="-1" sock="-1" isReusingSocket="no"
isConnecting="no"/>

  ....

</Subscriptions>
```

7.1.4 Nodes

/rest/nodes

returns nodes, scenes, types, and their status

/rest/nodes/scenes

returns scenes only

/rest/nodes/<node>

returns all the attributes & property values for a specific node

/rest/nodes/<node>?members=true|false

this only works on a scene. using members=true includes all the scene members in the result

/rest/nodes/<node>/enable

Enables a device

Returns: Success or Error status

/rest/nodes/<node>/disable

Disable a device

Returns: Success or Error status

/rest/nodes/<node>/notes

Returns the notes for the node

Schema (Notes):

```
<NodeProperties>
 <location>Free form text</location>
 <description>Free form text</description>
 <isLoad></isLoad>
 <spoken>The spoken name</spoken>
</NodeProperties>
```

Schema (Node):

```
<node flag="0">
 <address>The address of the node</address>
 <name>Friendly name</name>
 <parent type="see 3.5.1">the address of the parent</parent>
 <family>optionally defines device's family; see below</family>
 <type>device type; see below</type>
 <enabled>"true"|"false"</enabled>
 <deviceClass>1024</deviceClass>
 <wattage>2000</wattage>
 <dcPeriod>60</dcPeriod>
 <pnode>The primary node address (see below)</pnode>
 <sgid>Subgroup ID (see below)</sgid>
 <tx>Controller end-point bitmask</tx>
 <rx>Responder end-point bitmask</rx>
 <qry />
 <ctl />
</node>
```

Terms:

end-point : a single feature on a device such as the load or a KPL button.

subgroup : A subset of one or more end-points for a device

primary node : The node designated as the overall representative of a device

Flags:

NODE_IS_INIT 0x01 //needs to be initialized

NODE_TO_SCAN 0x02 //needs to be scanned

NODE_IS_A_GROUP 0x04 //it's a group!

NODE_IS_ROOT 0x08 //it's the root group

NODE_IS_IN_ERR 0x10 //it's in error!

NODE_IS_NEW 0x20 //brand new node

NODE_TO_DELETE 0x40 //has to be deleted later

NODE_IS_DEVICE_ROOT 0x80 //root device such as KPL load

Schema (Scene):

```
<group flag="(see Node flags)">
 <address>The address of the group</address>
 <name>Friendly name</name>
 <fmtDevId>Friendly name</fmtDevId>
 <members>
   <link type="relationship type">5 8A 37 1</link>
   <link type="relationship type">5 8A 37 3</link>
   …
 </members>
</group>
```

Node Type (Hierarchy) Flags:

NODE_TYPE_NOTSET 0 (unknown)

NODE_TYPE_NODE 1

NODE_TYPE_GROUP 2

NODE_TYPE_FOLDER 3

Schema (Folder):

```
<folder flag="(see Node flags)">

 <address>The address of the Folder</address>

 <name>Friendly name</name>

</folder>
```

Note:

1. A Node can belong to multiple Groups acting as Controller or Responder
2. A Node can belong only to ONE and only ONE Folder or another Node
3. A Group can belong only to ONE and only ONE Folder
4. A Folder can belong only to ONE and only ONE Folder

- Node/Scene/Folder/Program name are not unique
- Scene Folder addresses are unique only with in their group type

7.1.5 Properties

/rest/nodes/<node>/<property>

returns the specific property value for a given node id

Example :

```
<properties>

  <property id="ST" value="0" formatted="Off" uom="%/on/off"/>

</properties>
```

/rest/nodes/<node>/set/<property>/<value>

set a value such as OL/250

7.1.6 Commands

/rest/nodes/<node>/cmd/<command_name>/<param1>/<param2>/…/<param5>

eg:

/rest/nodes/<node>/cmd/DOF

turn off a device or a scene

/rest/nodes/<node>/cmd/DON/

turn on a device

- Insteon - /rest/nodes/<node-id>/cmd/DON/128 - turn on a scene to 50% (valid parameters = 0 - 255)
- UPB - /rest/nodes/<node-id>/cmd/DON/50 - turn on a scene to 50% (valid parameters = 0 - 100)

/rest/nodes/<node>/cmd/DFON

turn on a device fast

/rest/nodes/<node>/cmd/DFOF

turn off a device fast

/rest/nodes/<node>/cmd/BRT

increase brightness of a device by ~3%

/rest/nodes/<node>/cmd/DIM

decrease brightness of a device by ~3%

/rest/nodes/<node>/cmd/BMAN

begin manual dimming

/rest/nodes/<node>/cmd/SMAN

stop manual dimming

7.1.7 Status

/rest/status

returns the status for all the nodes

Example

```
<nodes>
 <node id="15 4B 53 1">
  <property id="ST" value="0" formatted="Off" uom="%/on/off"/>
 </node>
 <node id="16 38 C0 1">
  <property id="ST" value="0" formatted="Off" uom="%/on/off"/>
 </node>
 <node id="16 3C 43 1">
  <property id="ST" value="0" formatted="Off" uom="%/on/off"/>
 </node>
  ....
</nodes>
```

/rest/status/<node>

returns the status for the given node

Example

```
<properties>
  <property id="ST" value="0" formatted="Off" uom="%/on/off"/>
</properties>
```

7.1.8 Query

/rest/query

queries all the nodes

Returns: Success or Error status

/rest/query/<node>

queries the given node

Returns: Success or Error status

7.1.9 X10

/rest/X10/<Housecode>/<X10>

UnitCode and X10 command are both optional

List of X10 commands

Code	Function	Description	One Way	Two Way
13	All units off	Switch off all devices with the house code indicated in the message	X	
5	All lights on	Switches on all lighting devices (with the ability to control brightness)	X	
1	All lights off	Switches off all lighting devices	X	
3	On	Switches on a device	X	
11	Off	Switches off a device	X	
15	Dim	Reduces the light intensity	X	
7	Bright	Increases the light intensity	X	
9	Extended code	Extension code		X
14	Hail request	Requests a response from the device(s) with the house code indicated in the message		X
6	Hail acknowledge	Response to the previous command		X
12	Pre-set dim	Allows the selection of two predefined levels of light intensity		X
8	Status is on	Response to the Status Request indicating that the device is switched on		X
2	Status is off	Response indicating that the device is switched off		X
10	Status request	Request requiring the status of a device		X

Figure 36: X10 Commands

Please note this mapping is not equal to the actual X10 binary codes.

7.1.10 Programs

/rest/programs/<pgm>/<cmd>

Runs a command for a single program

Returns: Success or Error status

/rest/programs/<pgm>

returns single program, or folder with folders/programs within it

/rest/programs/<pgm>?folderContents=false

returns single program or folder

/rest/programs/<pgm>?subfolders=true

returns single program, or folder with folders/programs within it recursively

/rest/programs

returns all the programs in the root folder e.g. same as /rest/programs/<root>

/rest/programs?folderContents=false

returns root folder only (same as /rest/programs/<root>?folderContents=false)

/rest/programs?subfolders=true

returns all programs & folders (same as /rest/programs/<root>?subfolders=true)

Defaults:

- folderContents=true
- subfolders=false

Example:

/rest/programs/0032/runThen

pgm-cmd

- run|runThen|runElse|stop|enable|disable|enableRunAtStartup|disableRunAtStartup

'runIf' is supported as well, but 'run' should be used instead

Schema :

```
<program id="0053" parentId="0001" status="true" folder="true">
 <name>Am Dim</name>
 <lastRunTime></lastRunTime>
 <lastFinishTime></lastFinishTime>
</program>
```

7.1.11 Logs

/rest/log

Returns system/event log

/rest/log?reset=true

Clears all system log entries

Returns: Success or Error status

System log has the following format:

Node – the address of the node to which the log belongs

Control – the control that was impacted and which caused the log entry

Action – the value of the control

Time – in NTP format with epoch of 36524

UID* – the user or task which initiated the event :

SYSTEM_USER=0 | SYSTEM_DRIVER_USER=1 | WEB_USER=2 | SCHEDULER_USER=3 |

D2D_USER=4, ELK_USER=5 | SEP_DEVICE_UMETER_USER=6 |
SEP_DEVICE_UPRICE_USER |

SEP_DEVICE_UMSG_USER | SEP_DEVICE_UDR_USER | GAS_METER_USER

Log Type – the type of entry

/rest/log/error

Return error log

/rest/log/error?reset=true

Clears all error log entries

Returns: Success or Error status

Error log has the following format:

Time – in NTP format with epoch of 36524 (see section 7.3)

UID* – the user or task which initiated the event

SYSTEM_USER=0 | SYSTEM_DRIVER_USER=1 | WEB_USER=2 | SCHEDULER_USER=3 |

D2D_USER=4, ELK_USER=5 | SEP_DEVICE_UMETER_USER=6 |
SEP_DEVICE_UPRICE_USER |

SEP_DEVICE_UMSG_USER | SEP_DEVICE_UDR_USER |GAS_METER_USER

Log Type – the type of entry

Error Message – free form text

7.1.12 Variables

For the following command related to variables, <var-type> =

1 = Integer

2 = State

/rest/vars/init/<var-type>/<var-id>/<value>

Sets the initial value of variable at ISY startup

/rest/vars/set/<var-type>/<var-id>/<value>

Sets a variable given by var-id

/rest/vars/get/<var-type>/<var-id>

Retrieves a variable given by var-id

/rest/vars/get/<var-type>

Retrieves all variables of the given type

Schema :

```
<var id="<var-id>" type ="<var-type>">
   <val>value</val>
   <init>init-value</init>
   <ts>YYYYMDD HH:MM:SS</ts>
</var>
```

/rest/vars/definitions/<var-type>

Retrieves all variable definitions of the given type

Schema :

```
<CList type="VAR_INT">
   <e id="<var-id>" name="<var-name>" />
   …
   <e id="<var-idN>" name="<var-nameN>" />
</CList>
```

7.1.13 Subscriptions (Web Sockets)

Starting with firmware version 4.2.3, you can now use REST to subscribe to ISY using Web Sockets.

Web Sockets are persistent HTTP/S connections that can be used in HTML5 and Javascript.

For more information on Web Sockets: RFC6455

The process is quite simple: 1. Use a Web Socket request to /rest/subscribe 2. Ensure that the following headers are included a. Authorization. b. Sec-WebSocket-Protocol: ISYSUB c. Sec-WebSocket-Version: 13 d. Origin: com.universal-devices.websockets.isy 3. The initial return is the subscription response which includes the subscription id 4. ISY will continue publishing events to the client as long as the socket remains open.

Here is an example HTML page that implements a websocket connection to ISY, displaying the received events in the browser window. This page must either be hosted directly on the ISY user web space to function, or you must specially configure a product like Apache to proxy requests to your ISY, and exempt the path where your html file is hosted. For an example of this configuration, see here

```
<!DOCTYPE html>

<html lang="en">

<head>

 <title>WebSocket ISY Client</title>

 <meta charset="UTF-8" />

 <script>

  "use strict";

  // Initialize everything when the window finishes loading

  window.addEventListener("load", function(event) {

    var status = document.getElementById("status");

    var url = document.getElementById("url");

    var open = document.getElementById("open");

    var close = document.getElementById("close");

    var message = document.getElementById("message");

    var socket;



    status.textContent = "Not Connected";

    var isy_host = window.location.host;
```

```
var x = location.protocol;
if (x == "https:") {
  var ws_host = "wss://" + isy_host
} else {
    var ws_host = "ws://" + isy_host
}
url.value = ws_host + "/rest/subscribe";
close.disabled = true;

// Create a new connection when the Connect button is clicked
open.addEventListener("click", function(event) {
  open.disabled = true;
  socket = new WebSocket(url.value, "ISYSUB");

  socket.addEventListener("open", function(event) {
    close.disabled = false;
    status.textContent = "Connected";
        console.log("Connected");
  });

  // Display messages received from the server
```

```
    socket.addEventListener("message", function(event) {

        document.getElementById("log").innerText = event.data + String.fromCharCode(13) +
document.getElementById("log").innerText;

    });



    // Display any errors that occur

    socket.addEventListener("error", function(event) {

      message.textContent = "Error: " + event;

        console.log("Error: " + event)

    });



    socket.addEventListener("close", function(event) {

      open.disabled = false;

      status.textContent = "Not Connected";

        console.log("Not Connected");

    });

  });



  // Close the connection when the Disconnect button is clicked

  close.addEventListener("click", function(event) {

    close.disabled = true;

    message.textContent = "";
```

```
      socket.close();
    });
  });
 </script>
</head>
<body>
 Status: <span id="status"></span><br />
 <input id="url" type="hidden" />
 <input id="open" type="button" value="Connect" />
 <input id="close" type="button" value="Disconnect" /><br />
 <span id="message"></span>
 <p id="log"></p>
</body>
</html>
```

7.1.14 Modules

7.1.14.1Electricity

/rest/electricity

returns electricity module info

7.1.14.2Climate

/rest/climate

returns climate module info

Schema (example):

```
<?xml version="1.0" encoding="UTF-8"?>

<climate enabled="true" locationId="BRUCB"
rss="http://api.wxbug.net/getLiveWeatherRSS.aspx?ACode=AA2&stationId=BRUCB&unittype=0
">

  <Temperature>67.1 F</Temperature>

  <Temperature_High>81 F</Temperature_High>

  <Temperature_Low>60 F</Temperature_Low>

  <Feels_Like>67 F</Feels_Like>

  <Temperature_Rate>2.6 F/h</Temperature_Rate>

  <Humidity>29 %</Humidity>

  <Humidity_Rate>-11 %/h</Humidity_Rate>

  <Pressure>29.74 inches</Pressure>

  <Pressure_Rate>0 inches/h</Pressure_Rate>

  <Dew_Point>34 F</Dew_Point>

  <Wind_Speed>1 mph</Wind_Speed>

  <Wind_Average_Speed>0 mph</Wind_Average_Speed>

  <Wind_Direction>ESE</Wind_Direction>

  <Wind_Average_Direction>ESE</Wind_Average_Direction>
```

```
  <Gust_Speed>9 mph</Gust_Speed>
  <Gust_Direction>SSW</Gust_Direction>
  <Rain_Today>0 inches</Rain_Today>
  <Light>0 %</Light>
  <Light_Rate>0 %/h</Light_Rate>
  <Rain_Rate>0 inches/h</Rain_Rate>
  <Max_Rain_Rate>0 inches/h</Max_Rain_Rate>
  <Evapotranspiration>0.0637 inches/day</Evapotranspiration>
  <Irrigation_Requirement>9.8963 inches</Irrigation_Requirement>
  <Water_Deficit_Yesterday>0.0637 inches</Water_Deficit_Yesterday>
  <Elevation>935 </Elevation>
</climate>
```

7.1.14.3Networking

/rest/networking/resources

Returns the networking resources configuration / resource_id list

Schema (example):

```
<NetConfig>
 <NetRule>
  <name>Camera Start</name>
  <id>40</id>
  <isModified>false</isModified>
  <ControlInfo>
   <mode>URL-Encoded</mode>
   <protocol>http</protocol>
   <host>10.1.1.43</host>
   <port>80</port>
   <timeout>500</timeout>
   <method>GET</method>
   <encodeURLs>true</encodeURLs>
  </ControlInfo>
 </NetRule>
  ....
</NetConfig>
```

/rest/networking/resources/<resource_id>

Calls and executes net resource

Returns: Success or Error status

/rest/networking/wol

Returns the networking Wake On LAN configuration / wol_id list

Schema (example):

```
<NetConfig>
 <NetRule>
  <name>NAS</name>
  <id>1</id>
  <isModified>false</isModified>
  <mac>00-01-55-12-2E-AC</mac>
  <subnet>10.1.1.255</subnet>
 </NetRule>
 ....
</NetConfig>
```

/rest/networking/wol/<wol_id>

Calls and executes the WOL resource

Returns: Success or Error status

/rest/security

returns security module info

The following have been disabled for security reasons

- /rest/security/<code>/arm/stay
- /rest/security/<code>/arm/away
- /rest/security/<code>/disarm

7.1.14.4Misc

/rest/locations

returns the floorplan

Schema :

```
<UDFloorPlan>
 <locations>
  <location ISIMPORTEDFROMMODEL="false">
   <preferredBounds WIDTH="134" HEIGHT="208" Y="2" X="10"/>
   <preferredLocation Y="15" X="334"/>
   <address>0.45816479823871636</address>
   <name>dinn</name>
   <nodes>
    <node ISLABELONLY="false" ISCONTROLLER="true" TYPE="single">
```

```
   <preferredBounds WIDTH="150" HEIGHT="37" Y="0" X="0"/>

   <preferredLocation Y="42" X="18"/>

   <deviceUUID>uuid:00:21:b9:01:0c:e7</deviceUUID>

   <name>Dinning Room Light</name>

   <address>16 38 C0 1</address>

  </node>

   ....

 </nodes>

</location>

...

</locations>

</UDFloorPlan>
```

7.1.14.5Zigbee

Only applicable to ISY994 Z Series.

/rest/zb

Retrieves the status of Zigbee Network including Joined nodes if any

/rest/zb/scanNetwork

Sends a broadcast to all nodes already in the PAN so that they would announce themselves again. This is a good diagnostics tool for orphaned nodes

/rest/zb/ntable

Retrieves the neighbor table for the COO. This is a good diagnostics tool for retrieving LQI for each node in the PAN

/rest/zb/info

Retrieves Zigbee radio information such as EUID and firmware

/rest/zb/reset

Soft resets the Zigbee radio

/rest/zb/restoreDefaults

Factory resets the Zigbee radio

/rest/zb/nodes

Retrieves all the nodes which have joined the PAN

/rest/zb/nodes/[euid]

Retrieves information for the node with the given euid (joined only)

/rest/zb/nodes/[euid]/ping

Pings the node with the given euid (joined only)

/rest/zb/nodes/[euid]/remove

Removes the node with the given euid (joined only) from the PAN

/rest/zb/nodes/[euid]/ep

Retrieves the active endpoints for a node

7.1.14.6Z-Wave

Only applicable to ISY994 Series with the Z-Wave Dongle.

/rest/zwave/node/include?[power=true|false]&[nwi=true|false]

Add a device into the Z-Wave network.

power : True=High Power, False=Normal Power (default)

nwi : True=Network wide inclusion, False=Standard Inclusion (default)

/rest/zwave/node/exclude

Remove a device into the Z-Wave network.

/rest/zwave/node/cancel

Cancel include/exclude/replication.

/rest/zwave/sendPrimary

As Primary Controller, replicate to another controller and make it the new Primary Controller.

/rest/zwave/learnMode

Go into Z-Wave learn mode (to replicate, be added/removed from Z-Wave network, etc.).

/rest/zwave/sync?[id=<nodeAddress>]|[uid=<zwaveUnitId>]

Synchronize ISY with info on Z-Wave dongle for the given node.

id : The ISY node address

uid : The Z-Wave unit ID of the device

Specify either id or uid. If neither is specified then sync all new & deleted nodes.

/rest/zwave/sync/full?[id=<nodeAddress>]|[uid=<zwaveUnitId>]

Synchronize ISY with info on Z-Wave dongle for the given node.

id : The ISY node address

uid : The Z-Wave unit ID of the device

Specify either id or uid. If neither is specified then sync all nodes.

/rest/zwave/factoryReset/dongle?[force=true|false]

Factory reset the Z-Wave dongle.

force : True=Force factory reset, False=Factory reset if dongle is not part of existing Z-Wave network.

/rest/zwave/set/antenna?[0|1]

Sets active antenna; switches between internal and external.

0 - Internal Antenna

1 - External Antenna

Version 4.3.26+ and 5.0.6+ only

/rest/zwave/node/<nodeAddress>/config/query/<parameterNumber>

Query zwave device parameter

/rest/zwave/node/ZW003_1/config/query/2

returns something like:

<config paramNum="2" size="1" value="80"/>

/rest/zwave/node/<nodeAddress>/config/set/<parameterNumber>/<value>/<size>

Set zwave device parameter

The parameter size can be either 1,2, or 4 bytes

/rest/zwave/node/ZW003_1/config/set/1/77/1

/rest/zwave/node/ZW003_1/config/set/2/0xFFFFFFFF/4

7.1.14.7Gas

- *Only available on 992*

/rest/gmeter

Returns the status of the gas meter

/rest/gmeter/log

Returns gas meter log

/rest/gmeter?reset=true

Clears all gas meter log entries

7.1.15 Batch Commands

/rest/batch

Returns the Batch mode:

Example:

```
<batch>

 <status>[0|1]</status>

</batch>
```

/rest/batch/on

Turns on Batch mode. Does not write changes to device. Only internal configuration files are updated

/rest/batch/Off

Turns off Batch mode. Writes all pending changes to devices and no longer buffers changes

/rest/batteryPoweredWrites

Returns the status of Battery Powered device operations

```
<batteryPoweredWrites>
  <status>[0|1]</status>
</batteryPoweredWrites>
```

/rest/batteryPoweredWrites/on

Writes all pending changes to battery powered devices when Batch mode is off

/rest/batteryPoweredWrites/off

Does not write changes to battery powered devices when batch is off

Energy Monitoring

REST Interface for ECM

/rest/nodes/<nodeId>/stopRT

Stops Real Time reporting for an ECM

/rest/nodes/<nodeId>/startRT

Starts Real Time reporting for an ECM

/rest/nodes/<nodeId>/reset

Resets accumulated values for an ECM

/rest/nodes/<nodeId>/cfg

Retrieves all the configuration information for the given ECM the XML for

which is as follows:

```
<EMonConfig>
 <ct1 type="167" range="4"/>
 <ct2 type="167" range="4"/>
```

```
<pt type="131" range="6"/>

<rtInterval>real time interval (sec) </rtInterval>

<dlInterval>data logger interval (sec) </dlInterval>

<firmware>1026</firmware>

<id>unique id for the unit</id>

<serial>serial number for the unit</serial>

<aux constant="151" options="62">

 <trim1>0</trim1>

 <trim2>0</trim2>

 <trim3>0</trim3>

 <trim4>0</trim4>

 <trim5>0</trim5>

 <trim6>0</trim6>

</aux>

<k1 h="142" l="141"/>

<k2 h="142" l="141"/>

<kv0>212</kv0>

<option>0</option>

<trigger>200</trigger>

</EMonConfig>
```

/rest/nodes/<nodeId>/setCT?type=<T>&range=<R>

Sets CT configurations for a given channel (identified by the node).

Only channels 1 and 2 are supported.

T = Types as defined by ECM (100 | 167)

R = Range as defined by ECM (3 | 4 | 5)

/rest/nodes/<nodeId>/setPT?type=<T>&range=<R>

Sets PT configurations on the given ECM.

T = Types as defined by ECM (?)

R = Range as defined by ECM (?)

/rest/nodes/<nodeId>/setInt?value=<V>

Sets Real Time interval (in seconds) on the given ECM.

V = The interval in seconds

/rest/nodes/<nodeId>/setTrig?value=<T>

Sets the Trigger value (in watts) on the given ECM.

T = The trigger value in Watts

/rest/nodes/<nodeId>/setAux?value=<O>

Sets the Aux options on the given ECM.

O = A byte bitmap as follows (Gain is X2): Bit 0 = Aux 1 Gain Bit 1 = Aux 2 Gain Bit 2 = Aux 3 Gain Bit 4 = Aux 4 Gain Bit 5 = Aux 5 Gain Bit 5 = Aux 5 Is used for Counting when set Bit 6 = Aux 5 is DC bipoloar Bit 7 = ? Please note that these options are retrieved in the configuration in the option attribute of the aux element: <aux constant="151" options="62">

/rest/nodes/<nodeId>/toggPol

Toggles the polarity for the given channel (node) on the given ECM.

7.1.16 REST Interface for GreenEye Monitor

Prefix: /rest/nodes/<nodeId>/cmd

/rest/nodes/<nodeId>/stopRT

Stops Real Time reporting for an ECM

/rest/nodes/<nodeId>/startRT

Starts Real Time reporting for an ECM

/rest/nodes/<nodeId>/reset?type=<T>

Resets certain counters based on the type.

1: Reset Pulse Counter 1

2: Reset Pulse Counter 2

3: Reset Pulse Counter 3

4: Reset Pulse Counter 4

5: Reset All Pulse Counters

6: Reset All Counters

7: Reset All Seconds Counters

8: Reset Seconds Counter for the Node in <nodeId>

/rest/nodes/<nodeId>/cfg

Not implemented.

/rest/nodes/<nodeId>/setCT?type=<T>&range=<R>

Not Implemented

/rest/nodes/<nodeId>/setPT?type=<T>&range=<R>

Sets PT configurations on the given ECM.

T = Types as defined by ECM (?)

R = Range as defined by ECM (?)

/rest/nodes/<nodeId>/setInt?value=<V>

Sets Real Time interval (in seconds) on the given ECM.

V = The interval in seconds

/rest/nodes/<nodeId>/setTrig?value=<T>

Not Implemented

/rest/nodes/<nodeId>/setAux?value=<O>

Not Implemented

/rest/nodes/<nodeId>/toggPol

Not Implemented

7.1.17 REST Interface for EM3

/rest/nodes/<nodeId>/stopRT

Stops Real Time reporting for an EM3

/rest/nodes/<nodeId>/startRT

Starts Real Time reporting for an EM3

/rest/nodes/<nodeId>/reset?type=*

Resets various parameters in EM3.

*<RT> can be a bitwise OR of the following:

1 = Reset accumulated values

2 = Reset configuration parameters

4 = Reset Zigbee

8 = Reset Pulse Count

/rest/nodes/<nodeId>/cfg

Retrieves all the configuration information for the given EM3 the XML for which is as follows:

```
<EM3Config debug="Boolean" realtime="Boolean" tempUnit="F|C" kyzMode="Boolean">
  <powerReportInterval>integer (seconds)</powerReportInterval>
  <vaReportInterval>integer (seconds)</vaReportInterval>
  <tempReportInterval>integer (seconds)</tempReportInterval>
  <pulseReportInterval>integer (seconds)</pulseReportInterval>
  <powerStorageInterval>integer (seconds)</powerStorageInterval>
  <pulseStorageInterval>integer (seconds)</pulseStorageInterval>
</EM3Config>
```

/rest/nodes/<nodeId>/setCT?type=<T>&range=<R>

Currently not supported.

/rest/nodes/<nodeId>/setPT?type=<T>&range=<R>

Currently not supported.

/rest/nodes/<nodeId>/setInt?type=<T>&value=<V>

Sets Real Time interval (in seconds) for a various parameters, defined in T, on the given EM3.

T: could be any of the following:

1 = Power reporting interval

2 = Voltage/Current/Powerfactor (va) reporting interval

3 = Temperature reporting interval

4 = Pulse reporting interval

5 = Energy storage interval

6 = Pulse storage interval

V: The interval in seconds

/rest/nodes/<nodeId>/setTrig?value=<T>

Sets the Trigger value (in watts) on the given EM3.

T = The trigger value in Watts

/rest/nodes/<nodeId>/setOption?type=<OT>&value=<O>

Sets operating parameters for the EM3.

OT = is an option type which could be any of the following:

1 = Temp unit

Where O could be:

0=F

4=C

2 = Debug

Where O could be:

0 = Debug Off

1 = Debug On

3 = KYZ Mode

Where O could be:

0 = KYZ Mode Off

1 = KYZ Mode On

Please note that these options are retrieved in the configuration in the option attribute of the aux element:

<aux constant="151" options="62">

7.1.18 Portal Integration

/rest/whoami

This interface returns uniquely identifying attributes of the given ISY to which the Proxy Server is connected:

Schema :

```
<iam>
  <partner>A string representing the partner ID</partner>
  <type>A string representing the type of ISY</type>
  <product>ISY's product ID which represents a model number</product>
  <id>Hexadecimal MAC address</id>
  <SID>
    Subscription ID if any  this element will not be there if
    ISY has no active subscription to the Portal. Otherwise,
    the value is always: uuid:0
  </SID>
  <electricity>
    <providerId>
      A String representing the Utility provider ID if any
    </providerId>
    <enrollmentGroup>
      An Integer representing the Utility enrollment group if any
    </enrollmentGroup>
```

```
    </electricity>
</iam>
```

Example:

```
<iam>
    <partner>Portal-X-Partner</partner>
    <type>ISYv30</type>
    <product>1110</product>
    <id>0021b9030405</id>
    <SID>uuid:0</SID>
    <electricity>
        <providerId>SDGE</providerId>
        <enrollmentGroup>0</enrollmentGroup>
    </electricity>
</iam>
```

7.1.19 ELK Integration

7.1.19.1Area Commands

/rest/elk/areas/query

Queries all areas and changes to states are published through event infrastructure.

/rest/elk/area/<areaId>/get/status

Retrieves the status of the given area.

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

Response : elkobjs.xsd:uelk:AreaResponseType

/rest/elk/area/<areaId>/cmd/bypass?code=<accessCode>

Bypasses all violated burglar alarms in the area given in areaId

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

/rest/elk/area/<areaId>/cmd/unbypass?code=<accessCode>

Unbypasses all burglar alarms in the area given in areaId

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

/rest/elk/area/<areaId>/cmd/display?text=<eText>&beep=<boolean>&offTimerSeconds=<seconds>

Displays text given in eText to all keypad in the given area.

Optionally beep and turn off after offTimerSeconds.

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

eText : URL escaped text. 2 Lines with 16 characters per line max. Use "^" to separate lines if less than 16 characters.

e.g. text=”Hello^World”

beep : Optional; True then beep when displaying the message

seconds : Optional; Number of seconds to display the message

/rest/elk/area/<areaId>/cmd/display?id=<notifyId>&beep=<boolean>&offTimerSeconds=<seconds>

Displays formatted text to all keypad in the given area given a custom content ID.

This uses the same custom content entries as are used for email notifications.

Optionally beep and turn off after offTimerSeconds.

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

notifyId : The id of the customized content entry

beep : Optional; True then beep when displaying the message

seconds : Optional; Number of seconds to display the message

/rest/elk/area/<areaId>/cmd/arm?armType=<elkArmType>&code=<accessCode>

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

elkArmType : Defined by elkobjs.xsd:uelk:ArmType

accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

/rest/elk/area/<areaId>/cmd/disarm?code=<accessCode>

areaId : Defined in elkobjs.xsd:uelk:AreaIDType

accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

7.1.19.2Zone Commands

/rest/elk/zones/query

Queries all zones and changes to states are published through event .

/rest/elk/zones/<zoneId>/query/voltage

Queries the zone given by zoneId and changes to states are published through event infrastructure.

To query voltage for all zones, specify a zoneId of 0 (zero).

zoneId : Defined in elkobjs.xsd:uelk:ZoneIDType

/rest/elk/zone/<zoneId>/query/temperature

Queries the thermostats for the given zone and changes to states are published through event infrastructure.

To query temperature for all zones, specify a zoneId of 0 (zero).

zoneId : Defined in elkobjs.xsd:uelk:ZoneIDType

/rest/elk/zone/<zoneId>/cmd/trigger/open

Momentarily triggers a zone to the physical state of Open.

An error will occur if the zone is defined as normally open, or is currently open.

zoneId Defined in elkobjs.xsd:uelk:ZoneIDType

/rest/elk/zone/<zoneId>/cmd/toggle/bypass?code=<accessCode>

Toggles bypass for the zone given in zoneId

zoneId : Defined in elkobjs.xsd:uelk:ZoneIDType

accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

7.1.19.3General Commands

/rest/elk/query/all

Gets the status for all areas, zones, outputs, counters, etc. and publishes the state changes through event infrastructure.

/rest/elk/get/status

Returns the status of all of zones, areas, keypads, outputs, etc. configured in ELK.

Response : elkobjs.xsd:uelk:ELKAllStatus

/rest/elk/get/topology

Returns the representation of all the zones, areas, keypads, outputs, etc. configured in ELK.

Response : elkobjs.xsd:uelk:Topology

/rest/elk/refresh/topology

Causes ISY to recreate/refresh the topology. Returns the representation of all the zones, areas, keypads, outputs, etc. configured in ELK.

Response : elkobjs.xsd:uelk:Topology

7.1.19.4System Commands

/rest/elk/system/get/status

Returns the enabled and connected status.

Response : elkobjs.xsd:uelk:SystemResponseType

7.1.19.5Keypad Commands

/rest/elk/keypad/<kpId>/cmd/press/funcKey/<fkId>

Simulates pressing the fkId button on keypad given in kId

kpId : Keypad number (1-8)

fkId : Defined in elkobjs.xsd:uelk:FunctionKeyType

/rest/elk/keypad/<kpId>/query/temperature

Queries the status of temperature sensor on a keypad and changes to states are published through event infrastructure : To query temperature for all keypads, specify a kpId of 0 (zero).

/rest/elk/keypad/<kpId>/get/status

Retrieves the status of a keypad.

To get status for all keypads, specify a kpId of 0 (zero).

kpId : Keypad number (1-8)

Response : elkobjs.xsd:uelk:KeypadResponseType

Note: See Area Commands for displaying text on keypads

7.1.19.6Output Commands

/rest/elk/outputs/query

Queries all outputs and changes to states are published through event infrastructure

/rest/elk/output/<outputId>/get/status

Returns the status of the given output

To get status for all outputs, specify an outputId of 0 (zero).

outputId : elkobjs.xsd:uelk:OutputIDType

Response : elkobjs.xsd:uelk:OutputResponseType

/rest/elk/output/<outputId>/cmd/on?offTimerSeconds=<seconds>

Turns On an output defined by outputId. Optional offTimerSeconds can be defined such that the output is turned back off after the amount of seconds given in <seconds>.

outputId : elkobjs.xsd:uelk:OutputIDType

seconds : Optional; Interval after which the relay turns off

/rest/elk/output/<outputId>/cmd/off

Turns Off an output defined by outputId

outputId : elkobjs.xsd:uelk:OutputIDType

7.1.19.7Audio Commands

Audio commands allow communications with A/V equipment that have been configured and attached to ELK.

/rest/elk/audio/zone/<audioZone>/source/<audioSource>/cmd/<audioCommand>?value=<audioValue>

Sends the command given in audioCommand to zone given in audioZone.

audioZone : Audio Zone number (1-18)

audioSource : Audio Source number (1-12)

audioCommand : Defined by elkobjs.xsd:uelk:AudioCommandType

audioValue : 3-digit decimal Number, currently only used for Volume

7.1.19.8Voice Announcement Commands

Voice Announcement commands cause predefined words or phrases to be spoken by the ELK security system.

/rest/elk/speak/word/<wordId>

Instructs ELK to speak the word given by wordId.

wordId : Defined by elkobjs.xsd:uelk:VoiceWordType

/rest/elk/speak/phrase/<phraseId>

Instructs ELK to speak the phrase given by phraseId.

phraseId : Defined by elkobjs.xsd:uelk:VoicePhraseType

7.1.19.9Thermostat Commands

Thermostat commands impact a thermostat.

/rest/elk/tstat/<tstat_id>/query

Instructs ELK query the thermostat given by tstat_id and state changes are published as events.

tstat_id : Defined by elkobjs.xsd:uelk:ThermostatIDType

/rest/elk/tstat/<tstat_id>/get/status

Retrieves the status of the given thermostat.

tstat_id : Defined by elkobjs.xsd:uelk:ThermostatIDType

Response : elkobjs.xsd:uelk:ThermostatResponseType

/rest/elk/tstat/<tstat_id>/cmd/<cmd_id>?value=value

Instructs ELK to apply the value using command defined by cmd_id to thermostat defined by tstat_id.

tstat_id Defined by elkobjs.xsd:uelk:ThermostatIDType

cmd_id : Defined by elkobjs.xsd:uelk:ThermostatCommandType

value : Depends on the command:

Temperatures (such as setpoints): 1-99

Mode: elkobjs.xsd:uelk:ThermostatModeState

Fan: elkobjs.xsd:uelk:ThermostatFanStat

7.1.20 OpenADR (VEN)

/rest/oadr

receive full Payload

/rest/oadr/status

receive the connection status to VTN

/rest/oadr/<event-id>/optin

opt in to an event

/rest/oadr/<event-id>/optout

opt out to an event

/rest/oadr/clear

clear all events

/rest/oadr/party/register

Register to VTN

/rest/oadr/party/unregister

Cancel Registration to VTN

/rest/oadr/party/queryreg

Query Registration

/rest/oadr/party/reregister?type=val

Re-Registration

Where type is refresh | renew

refresh = uses the existing Registration ID

renew = does not use the existing Registration ID

/rest/oadr/report/register?type=val

Register report Meta Data

Where type is usage | status | history | all [default]

/rest/oadr/report/status

Get status of all reports (oadrobjs:OADRReportsStatus)

/OpenADR2/Simple/EiEvent

For 2.0a Push mode

/OpenADR2/Simple/2.0b/[Service]

For 2.0b Push mode, ISY endpoint is:

Where service is:

EiEvent - Event services

EiReport - Reporting services

EiOpt - Scheduling services

7.1.21 Billing

These REST URLs are used to provide utility billing information when used in conjunction with Smart Meters (ZS Series).

/rest/billing/today[/reset]

returns the billing information for today (WSDL/billobjs.xsd::Billing)

/rest/billing/yesterday[/reset]

returns the billing information for yesterday (WSDL/billobjs.xsd::Billing)

/rest/billing/cycle[/reset]

returns the billing information for the current cycle (WSDL/billobjs.xsd::Billing)

/rest/billing/month/year

returns the billing information for the given month/year cycle (WSDL/billobjs.xsd::Billing)

/rest/billing/report/today

returns the hourly billing information for today (WSDL/billobjs.xsd::BillingReport)

/rest/billing/report/yesterday

returns the hourly billing information for yesterday (WSDL/billobjs.xsd::BillingReport)

/rest/billing/report/month/year

returns the daily billing information for the given month/year cycle (WSDL/billobjs.xsd::BillingReport)

8 ISY Developer's Manual

Web Services SDK and REST Interface fort INSTEON/UPB/Z-Wave, based on firmware 4.3.9

8.1 Introduction

ISY is a sophisticated events-based network platform which affords its clients unprecedented levels of integration and functionality. Now, with the introduction of WSDK, most of ISY functions are externalized as Web Services and defined in a well formed WSDL which can immediately be imported into an IDE of choice.

At a high level, ISY operates and may be communicated with in the following order:

1. Upon power up, ISY sends out broadcast messages of its location to all the UPnP clients on the network
1. Interested clients may choose to:
    a. Search for a specific ISY (based on the device type it supports such as Insteon)
    b. Listen in for ISY generated announcements on the network
    c. Immediately start communicating with ISY using a predefined IP (static)
2. address and port, if one is already known
3. Upon discovery of an ISY – regardless of the method chosen – communications with ISY takes place through Web Services/SOAP 1.2 calls:
    a. All requests need to have an HTTP Basic Authentication Header (Realm="/")
    b. Optionally ***subscribe*** to the ISY events from which time ISY continuously notifies the subscriber(s) of the changes in its state. Upon successful subscription, ISY publishes all its current states to the client so that the client and ISY are in synch at the moment of subscription. In this respect, then, the clients are started wi the current state of ISY and are notified of all the changes as they occur and thus will never have to poll ISY
5. During application exit, the client must notify ISY that it wishes to terminate its
4. session. This is achieved by issuing:
    a. Unsubscribing from ISY
6. During normal operations, the client *must* always respond back (immediately) with an Ack to ISY's Heartbeat events otherwise ISY assumes a client malfunction (the client didn't exit gracefully as outlined in step 4) and terminates the associated session.

As mentioned before, ISY is event driven and thus every change in ISY is notified/published to all the ISY subscribed clients in real-time and almost immediately. In

this respect, then, one could use the default ISY User Interface (a signed Java applet) to effect a change while using one's own client to view all the changes that are taking place (and vice versa).

8.2 Basic Concepts

8.2.1 Control

A *Control* is the logical representation of either a state or a function that may be performed on a physical device (or a scene) linked to ISY. For example, "DON" is the name of the *Control* which instructs ISY to turn a "Deivce On" while "ST" is the name of the *Control* which holds that *state* of a device.

In essence, then, Control is what "captures" and "controls" changes in the states of physically linked devices or groups/scenes. Since Controls may be associated with states, thus, all ISY publications (publish) to all clients contain a Control parameter which identifies "what changed".

For example, a CLISP (Climate SetPoint) Control not only allows the client to effect a SetPoint change on a linked Thermostat but also, as soon as the change takes effect (or the state changes), ISY notifies all the clients of the change in "CLISP" and the current value thereto (see section **8.2.2 Action**) if any.

The most important attributes of a Control are:

A Name – this is the Control's only meaningful unit of communications with ISY such as "CLISP", "DON", "DIM", etc.

A Label – this is an optional label that the developer/manufacturer may ascribe to a Control such as "SetPoint", "On", "Dim", "Fast On", etc.

*Action*s – this is a list of optional while permissible actions which may be performed on a Control such as "50" which, when applied to "DON", means turn the "Device On to 50%". Or, when "HEAT" is applied to "CLIMD" (Climate Mode) it means change the thermostat "Mode to Heat". For more details, see section **8.2.2 Action**

8.2.2 Action

An *Action* is the permissible "value" which may be applied to a *Control*. A Control may have a set of permissible Actions which are captured by a list.

When communicating a state change request to ISY, Action may be null. This said, however, when ISY publishes (to its subscribers) changes to a Control – and if the Control is associated with some state – then this attribute holds the "current value" of the state. For example, when issuing a "DON" to ISY the "ST" Control (which is associated with a state) is updated and, as such, ISY shall notify all the subscribed clients of a change in "ST" with Action being the current value of "ST" such as "50"%.

The most important attributes of an Action are:

A Name – this is the Action's only meaningful unit of communications with ISY. Depending on the Control, this attribute may take the form of a free text/object field the value of which is filled in by ISY upon publications of events.

A Label – this is an optional label that the developer/manufacturer may ascribe to an Action such as "Heat".

8.2.3 Node

A Node is a logical representation of a physical device linked to ISY. So, for instance, KeypadLinc's button A is a node and so are its buttons B, C, D through H.

In essence – and when put in the context of a Control and Action – the Node is the only missing piece which, when all put together, enables effecting the desired change on a physical device linked to ISY.

The most important attributes of Node are:

An Address – this is the address which ISY uses to communicate with the actual physical device such as 4 E 52 1

A Name – this is the user friendly name which can be changed by any ISY client

States (device Variables) – this is the list of all the Controls for a Node and their current associated Actions (values)

A note on Insteon addresses:

Since, as mentioned before, every button is also considered a device within ISY, thus, each button shall have its own address conforming to the following syntax: X X X B – where X is the actual Insteon address for the device in hex and B is the button group number.

For instance, a 6 button KeypadLinc with address 04 E8 52 will have the following nodes within ISY:
4 E8 52 1 – the main [loaded] button
4 E8 52 A – Button A
4 E8 52 B – Button B
4 E8 52 C – Button C
4 E8 52 D – Button D

8.2.4 Group/Scene

A Group is a specialization of Node with the added capability of aggregating associated/linked Nodes. Just like a Node, a Group may also be used to effect a change in ISY. The only difference is that issuing a state change on a Group results in ISY sending notifications on the states of all the Nodes within that Group/Scene (if there were any changes).

8.2.5 Putting it Together

By having a triplet {control, action, [node or group/scene]} it's quite easy to effect change on the physical devices which are linked/attached to ISY. For instance:

1. To turn on the light at address 7 B0 B2 to 60%, a simple service call of the type *UDIService* ("DON", "60", "7 B0 B2 1"), is all it takes.
2. To turn off the scene at address 52626, a simple method call of the type *UDIService*("DOF", null, "52626"), is all it takes.

8.2.6 ISY Messages and Web Services

All messages, Web Services, Parameters, Objects, and Events are captured in a WSDL file stored on ISY.

8.3 Discovering ISY and its Resources

ISY Can be found using UPnP Search method. ISY also advertises its presence on the network every 30 seconds.

8.3.1 Discovering ISY Using UPnP Search

Send the following UDP Packet to UPnP Multicast group of 239.255.255.250 and port 1900

M-SEARCH * HTTP/1.1
HOST:239.255.255.250:1900
MAN:"ssdp.discover"
MX:1
ST:urn:udi-com:device:X_Insteon_Lighting_Device:1

Note: X_Insteon_Lighting_Device is the UPnP Device Type for INSTEON ISY devices

ISY replies with:

HTTP/1.1 200 OK
CACHE-CONTROL:max-age=30
EXT:
LOCATION:http://192.168.0.208/desc
SERVER:UCoS, UPnP/1.0, UDI/1.0
ST:urn:udi-com:device:X_Insteon_Lighting_Device:1
USN:uuid:00:03:f4:03:0f:61::urn:udicom:

device:X_Insteon_Lighting_Device:1

8.3.2 Listening for ISY Advertisements on the Network

As mentioned before, ISY advertises its existence on the network every 30 seconds. To receive these notification events, join the UPnP Multicast group at 239.255.255.250 and port 1900. ISY advertisements are as follows:

1. Root Device

NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL:max-age=30
LOCATION:http://192.168.0.208/desc
NT:upnp:rootdevice
NTS:ssdp:alive
SERVER:UCoS, UPnP/1.0, UDI/1.0
USN:uuid:00:03:f4:03:0f:61::upnp:rootdevice

2. Service

NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL:max-age=30
LOCATION:http://192.168.0.208/desc
NT:urn:udi-com:service:X_Insteon_Lighting_Service:1
NTS:ssdp:alive
SERVER:UCoS, UPnP/1.0, UDI/1.0
USN:uuid:00:03:f4:03:0f:61::urn:udicom:
service:X_Insteon_Lighting_Service:1

3. Device

NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL:max-age=30
LOCATION:http://192.168.0.208/desc
NT:urn:udi-com:device:X_Insteon_Lighting_Device:1
NTS:ssdp:alive
SERVER:UCoS, UPnP/1.0, UDI/1.0
USN:uuid:00:03:f4:03:0f:61::urn:udicom:
device:X_Insteon_Lighting_Device:1

8.3.3 Capturing ISY Resources

Regardless of how ISY is discovered, the LOCATION header defines where other ISY resource URIs are located (UPnP Description file). Simply do an HTTP Get on the URL defined by the LOCATION header. The following is an example of the contents of: **http://192.168.0.208/desc**; The most important elements are in **bold**:

```
<?xml version="1. 0" ?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<specVersion>
    <major>1</major>
    <minor>0</minor>
    </specVersion>
    <URLBase>http://192.168.0.208</URLBase>
    <device>
    <deviceType>urn:udi-com:device:X_Insteon_Lighting_Device:1</deviceType>
    <friendlyName>My Lighting</friendlyName>
    <manufacturer>Universal Devices Inc.</manufacturer>
    <manufacturerURL>http://www.universal-devices.com</manufacturerURL>
    <modelDescription>X_Insteon_Lighting_Device:1</modelDescription>
    <modelName>Insteon Web Control</modelName>
    <modelNumber>Insteon Web Control</modelNumber>
    <UDN>uuid:00:03:f4:03:0f:61</UDN>
    <UPC>uuid:00:03:f4:03:0f:61</UPC>
    <serviceList>
    <service>
    <serviceType>urn:udicom:
    service:X_Insteon_Lighting_Service:1</serviceType>
    <serviceId>urn:udi-com:serviceId:uuid:00:03:f4:03:0f:61</serviceId>
    <SCPDURL>/services.wsdl</SCPDURL>
    <controlURL>/services</controlURL>
    <eventSubURL>/eventing</eventSubURL>
    </service>
    … other services such as SEP
    </serviceList>
    <presentationURL>/</presentationURL>
    </device>
</root>
```

URLBase: is the absolute URL to ISY services. All the other URLs are relative to this URL

UDN: is the Unique Device Number which uniquely identifies ISY on the network

SCPDURL: is the location where the definition of ISY services are located (in WSDL).
Note: point your WebServices IDE to this URL to import all services. E.g. http://192.168.0.102:8080/services.wsdl

controlURL: is the URL to which all the Service requests are Posted

eventSubURL: is the URL to which clients subscribe and unsubscribe –

Deprecated. Please see section 8.4 **Communicating with ISY**

8.3.4 ISY Configuration Resource

ISY Configuration Resource defines how ISY is presently configured. The most important feature of this resource is that it defines the permissible Controls/Actions which may be invoked in ISY. Here's an example:

```
…
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    –
    <deviceSpecs>
        <make>Universal Devices Inc.</make>
        <manufacturerURL>http://www.universaldevices.
        com</manufacturerURL>
        <model>Insteon Web Controller</model>
        <icon>/web/udlogo.jpg</icon>
        <archive>/web/insteon.jar</archive>
        <chart>/web/chart.jar</chart>
        <queryOnInit>true</queryOnInit>
        <oneNodeAtATime>true</oneNodeAtATime>
    </deviceSpecs>
    <upnpSpecs>
        <upnpDevice>
            <utype>X_Insteon_Lighting_Device</utype>
            <version>1</version>
```

```
    </upnpDevice>
    <upnpService>
        <utype>X_Insteon_Lighting_Service</utype>
        <version>1</version>
    </upnpService>
</upnpSpecs>
<controls>
    <control>
        <name>ST</name>
        <label>Status</label>
        <readOnly>true</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>%</numericUnit>
    </control>
    <control>
        <name>OL</name>
        <label>On Level</label>
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>%</numericUnit>
    </control>
    <control>
        <name>RR</name>
        <label>Ramp Rate</label>
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>%</numericUnit>
    </control>
    <control>
        <name>DON</name>
        <label>On</label>
    </control>
    <control>
        <name>DFON</name>
        <label>Fast On</label>
    </control>
    <control>
        <name>DOF</name>
```

```
        <label>Off</label>
    </control>
    <control>
        <name>DFOF</name>
        <label>Fast Off</label>
    </control>
    <control>
        <name>BRT</name>
        <label>Brighten</label>
    </control>
    <control>
        <name>DIM</name>
        <label>Dim</label>
    </control>
    <control>
        <name>BMAN</name>
        <label>Fade Start</label>
    </control>
    <control>
        <name>SMAN</name>
        <label>Fade Stop</label>
    </control>
    <control>
        <name>BEEP</name>
        <label>Beep</label>
    </control>
    <control>
            <name>RESET</name>
            <label>Reset values</label>
    </control>
    <control>
        <name>CLISPH</name>
        <label>Heat Setpoint</label>
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>F</numericUnit>
    </control>
    <control>
        <name>CLISPC</name>
        <label>Cool Setpoint</label>
```

```
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>F</numericUnit>
    </control>
    <control>
        <name>CLIFS</name>
        <label>Fan State</label>
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>false</isNumeric>
        <actions>
            <action>
                <name>7</name>
                <label>On</label>
            </action>
            <action>
                <name>8</name>
                <label>Auto</label>
            </action>
        </actions>
    </control>
    <control>
        <name>CLIMD</name>
        <label>Thermostat Mode</label>
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>false</isNumeric>
        <actions>
            <action>
                <name>0</name>
                <label>Off</label>
            </action>
            <action>
                <name>1</name>
                <label>Heat</label>
            </action>
            <action>
                <name>2</name>
                <label>Cool</label>
            </action>
```

```
            <action>
                <name>3</name>
                <label>Auto</label>
            </action>
            <action>
                <name>4</name>
                <label>Fan</label>
            </action>
            <action>
                <name>5</name>
                <label>Program Auto</label>
            </action>
            <action>
                <name>6</name>
                <label>Program Heat</label>
            </action>
            <action>
                <name>7</name>
                <label>Program Cool</label>
            </action>
        </actions>
    </control>
    <control>
        <name>CLIHUM</name>
        <label>Humidity</label>
        <readOnly>true</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>%</numericUnit>
    </control>
    <control>
        <name>CLIHCS</name>
        <label>Heat/Cool State</label>
        <readOnly>true</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>false</isNumeric>
        <actions>
            <action>
                <name>0</name>
                <label>Off</label>
            </action>
```

```
            <action>
                <name>1</name>
                <label>Heat On</label>
                </action>
            <action>
                <name>2</name>
                <label>Cool On</label>
            </action>
        </actions>
    </control>
    <control>
        <name>UOM</name>
        <label>Unit</label>
        <readOnly>true</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>false</isNumeric>
        <actions>
            <action>
                <name>1</name>
                <label>Celsius</label>
            </action>
            <action>
                <name>2</name>
                <label>Fahrenheit</label>
            </action>
        </actions>
    </control>
    <control>
        <name>CPW</name>
        <label>Current Power Usage</label>
        <readOnly>true</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
        <numericUnit>W</numericUnit>
    </control>
    <control>
        <name>TPW</name>
        <label>Total Power Used</label>
        <readOnly>true</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>true</isNumeric>
```

```
            <numericUnit>kWs</numericUnit>
        </control>
    </controls>
    <app>Insteon_UD994</app>
    <app_version>2.8.11</app_version>
    <platform>ISY-C-994</platform>
    <build_timestamp>2011-01-20-01:09:20</build_timestamp>
    <root>
        <id>00:03:f4:03:65:96</id>
        <name>ISY994</name>
    </root>
    <product>
        <id>1100</id>
        <desc>ISY 994i 1024</desc>
    </product>
        <features>
        <feature>
            <id>21010</id>
            <desc>Open Auto-DR</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21011</id>
            <desc>Electricity Meter</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21012</id>
            <desc>Gas Meter</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21013</id>
            <desc>Water Meter</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>false</isAvailable>
        </feature>
        <feature>
```

```
<id>21020</id>
    <desc>Weather Information</desc>
    <isInstalled>true</isInstalled>
    <isAvailable>true</isAvailable>
</feature>
<feature>
    <id>21030</id>
    <desc>Network Modules</desc>
    <isInstalled>false</isInstalled>
    <isAvailable>false</isAvailable>
</feature>
<feature>
    <id>21040</id>
    <desc>Networking Module</desc>
    <isInstalled>true</isInstalled>
    <isAvailable>true</isAvailable>
</feature>
<feature>
    <id>21050</id>
    <desc>Utility Meter (Electricity)</desc>
    <isInstalled>false</isInstalled>
    <isAvailable>true</isAvailable>
</feature>
<feature>
    <id>21051</id>
    <desc>SmartMeter ESP</desc>
    <isInstalled>false</isInstalled>
    <isAvailable>false</isAvailable>
</feature>
<feature>
    <id>21060</id>
    <desc>A10/X10 for Insteon</desc>
    <isInstalled>false</isInstalled>
    <isAvailable>false</isAvailable>
</feature>
<feature>
    <id>21070</id>
    <desc>Portal Integration - Check-it.ca</desc>
    <isInstalled>true</isInstalled>
    <isAvailable>true</isAvailable>
</feature>
```

```
        <feature>
            <id>21014</id>
            <desc>Current Cost Meter</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21080</id>
            <desc>Broadband SEP Device</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21071</id>
            <desc>Portal Integration - GreenNet.com</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
    </features>
    <triggers>true</triggers>
    <security>SSL</security>
    <isDefaultCert>false</isDefaultCert>
</configuration>
```

8.3.4.1Modules (Features)

ISY allows for optional modules to be installed. The correct behavior of the client may depend on the installed modules such as Climate and A10/X10. Modules are identified by the **<feature>** element in the configuration resource (see section 3.4). As such, it's advised that the client code queries the modules to enable/disable functionality accordingly.

Currently available modules are listed herein under:

21010 - Open Auto-DR
21011 - Electricity Meter (Brultech)
21012 - Gas Meter
21013 - Water Meter
21014 - Current Cost Meter
21020 - Weather Information (WeatherBug)
21030 - Network Modules (NOT AVAILABLE)
21040 - Networking Module
21050 - Utility Meter (Electricity): Zigbee SEP
21051 - SmartMeter ESP
21060 - A10/X10 for Insteon
21070 - Portal Integration - Check-it.ca
21071 - Portal Integration - GreenNet.com
21073 – Portal Integration – MobiLinc Connect
21080 - Broadband SEP Device
21090 – ELK Module
21100 – Z-Wave Module
22000 – Zigbee RCS Module
23000 – Irrigation Module

8.3.5 ISY Nodes Configuration Resource

ISY Nodes Configuration Resource defines all the Nodes/Scenes as configured in ISY with their relationships.

There are 3 types of nodes in ISY:
Node … identifies and end or virtual device or button (<node>)
Group ... identifies a scene or a logical grouping between devices (<group>)
Folder … identifies a logical grouping of nodes and groups without regards to their relationships

Note:

1. A **Node** can belong to multiple **Groups** acting as Controller or Responder
2. A **Node** can belong only to ONE and only ONE **Folder** or another **Node**
3. A **Group** can belong only to ONE and only ONE **Folder**
4. A **Folder** can belong only to ONE and only ONE **Folder**

8.3.5.1Types of Nodes/Parents

UD_HIERARCHY_NODE_TYPE_NOTSET 0 (unknown)
UD_HIERARCHY_NODE_TYPE_NODE 1
UD_HIERARCHY_NODE_TYPE_GROUP 2
UD_HIERARCHY_NODE_TYPE_FOLDER 3

8.3.5.2Node (<node>)

This element defines a configured node in ISY. A node is anything that can be impacted upon or if it can impact some change in the environment. As such, a node could be a KeypadLinc's button or a Thermostat.

<**node** flag="0">
<**address**>The address of the node</**address**>
<**name**>Friendly name</name>
<**parent** type="see 3.5.1">the address of the parent</**parent**>
<**family**>optionally defines device's family; see below</**family**>
<**type**>device type; see below</**type**>
<**enabled**>"true"|"false"</enabled>
<**deviceClass**>1024</**deviceClass**>
<**wattage**>2000</**wattage**>
<**dcPeriod**>60</**dcPeriod**>
<**pnode>**The primary node address (see below)</**pnode>**
<sgid>Subgroup ID (see below)**</sgid>**
<tx>Controller end-point bitmask</**tx>**
<rx>Responder end-point bitmask</**rx>**
<qry />
<ctl />
<rsp [type="<rsp-type>"] />
</**node**>

Terms:

end-point a single feature on a device such as the load or a KPL button.
subgroup A subset of one or more *end-points* for a device
primary node The node designated as the overall representative of a device

e.g. Sample nodes for a single device

<node><address>**0015**</address><pnode>**0015**</pnode><sgid>**1**</sgid>…
<node><address>0029</address><pnode>**0015**</pnode><sgid>**2**</sgid>…
<node><address>0042</address><pnode>**0015**</pnode><sgid>**3**</sgid>…

<pnode>

The address of the primary node for the device partially represented by this node. If this node is the primary node then <pnode> will equal <address>

A device may be represented by one or more nodes. One of these nodes is designated the *primary node* and is used to help group the set of nodes for a device.

*Note: UPB Only

<sgid>

The ID identifying the subgroup of end-points for the device.

Each node representing a subset of the end-points for a device is identified by a unique *subgroup ID*. This is similar in function to the last digit in an Insteon Node Address.

*Note: UPB Only

<qry />

If present then this node may be queried.

*Note: UPB Only

<ctl />

If present then this node may be used as a scene controller.

*Note: UPB Only

<rsp [type="<rsp-type>"] />

If present then this node may be used as a scene responder.

type

U_DIM Standard dimmer (**Default Value**)
U_S_DIMT SAI Dimmer with Off Timer
U_RELAY Relay Device (on/off only)
U_LED_6 Button LEDS for 6-Button KPL
U_LED_8 Button LEDS for 8-Button KPL

*Note: UPB Only

<tx>

Bitmask indicating the controller endpoints represented by this node

*Note: UPB Only

<rx>

Bitmask indicating the responder endpoints represented by this node

*Note: UPB Only

"flag" Attribute

Defines the characteristics of the <node> as well as the <group> elements as follows (represented in decimal):

NODE_IS_INIT 0x01 //needs to be initialized
NODE_TO_SCAN 0x02 //needs to be scanned

//Node operations flags

NODE_IS_A_GROUP 0x04 //it's a group!
NODE_IS_ROOT 0x08 //it's the root group
NODE_IS_IN_ERR 0x10 //it's in error!
NODE_IS_NEW 0x20 //brand new node
NODE_TO_DELETE 0x40 //has to be deleted later
NODE_IS_DEVICE_ROOT 0x80 //root device such as KPL load

<deviceClass> Element *

Defines the class of device for energy management (as defined by SEP):

DC_HVAC 0x0001
DC_STRIP_HEATER 0x0002
DC_WATER_HEATER 0x0004
DC_POOL_PUMP 0x0008
DC_SMART_APPLIANCE 0x0010
DC_IRRIGATION_PUMP 0x0020
DC_MANAGED_LOAD 0x0040
DC_SIMPLE 0x0080
DC_EXTERIOR_LIGHTING 0x0100
DC_INTERIOR_LIGHTING 0x0200
DC_EV 0x0400
DC_GENERATION_SYSTEM 0x0800
DC_WASHER 0x1000
DC_DRYER 0x2000
DC_OVEN 0x4000
DC_FRIG 0x8000
DC_ALL 0xFFFF

*Available in ISY994 Series/EMS models only

<dcPeriod> Element*

Defines the Duty Cycle period in minutes.

*Available in ISY994 Series/EMS models only

<family> Element

Defines a device/node family such as ZWave, RCS, INSTEON, etc. This element is useful for multi-protocol configurations including with security systems.

Family is defined in **family.xsd** with accompanying **[id]_fam.xml** which defines the categories/subcategories for each family id.

<type> Element

Defines the Category and Subcategories of underlying devices as follows:

device category.device subcategory.version.reserved

For INSTEON Device Types, please checkout 8.10 Appendix A – INSTEON Device Categories/Subcategories

8.3.5.3Group/Scene (<group>)

Same as node but defines a scene with a list of members and their relationships to the scene.

```
<group flag="see 3.5.2 (Node)">
    <address>The address of the group</address>
    <name>Friendly name</name>
    <fmtDevId>Friendly name</fmtDevId>
    <members>
        <link type="relationship type">5 8A 37 1</link>
        <link type="relationship type">5 8A 37 3</link>
    …
    </members>
</group>
```

<fmtDevId>

A formatted device ID. In the case of UPB this contains the underlying UPB Link ID for the scene.

*Note: UPB Only

<members> Element

Is a list of members for the scene each one having different relationships defined by the **type** attribute of the **link** element.

<link> Element

Defines members of a group/scene.

"type" Attribute defines the role a node plays in relationship to other nodes in a scene:

NODE_IS_CONTROLLER 0x10 (decimal 16)

Other values should be considered Responders.

8.3.5.4Folder (<folder>)

Same as node but identifies a folder.

8.4 Communicating with ISY

To successfully communicate with ISY, the following steps must be taken

1. **Find ISY** and retrieve its resources by parsing the contents of the LOCATION header. See section: **8.3.3 Capturing ISY Resources**

a. Capture <controlURL> which should be the URL used for all the subsequent Web Services invocations
b. Point your WebServices IDE to SCPDURL to import web services

2. Authenticate – no longer needed as of release 2.7.5

Instead, you will use the **HTTP Basic Authorization** header to send the credentials to ISY with each request.

The format of the Authorization header is:
Authorization: Basic Base64(userid:password)

Where, base64 converts the string representation of userid followed by ':' and

followed by the password.

As an example, the Authorization header for userid=admin , password= admin is:
Base64(admin:admin)-> YWRtaW46YWRtaW4=

Therefore the Authorization header shall be:

Authorization: Basic YWRtaW46YWRtaW4=

3. Subscribe

```
POST /services HTTP/1.1
Host: 192.168.0.129:80
Authorization: Basic YWRtaW46YWRtaW4=
Content-Length: 193
Content-Type: text/xml; charset="utf-8"

<s:Envelope><s:Body><u:Subscribe xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><reportURL>REUSE_SOCKET</reportU
RL><duration>infinite</duration></u:Subscribe></s:Body></s:Envelope>
```

Note: you may provide a <reportURL> URL or you can keep this socket open (REUSE_SOCKET) to receive ISY events.

NOTE: Starting with 4.2.3, you can now subscribe to ISY using REST and Web Sockets. For more information, please see section **8.9 Web Sockets and Subscriptions** for more details.

4. Control Nodes/Scenes in ISY using UDIService

Simply provide the permissible values for <Control>, <Action>, and <Node>.

The <node> element is the address of the node/group to be impacted. The <flag> element **must be 4** if this is impacting a scene/group. Any other value is considered a node and not a group/scene.

Note: if you are using SOAP1.2 compliant IDE, **SOAPACTION** header is **not** required.

a. Turn Device Hall 2 On

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 210
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udicom:
service:X_Insteon_Lighting_Service:1#UDIService"

<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><control>DON</control><actio
n></action><flag>65531</flag><node>7 B0 A5
1</node></u:UDIService></s:Body></s:Envelope>
```

b. Turn Device Hall 2 On to 46%

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 213
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udicom:
service:X_Insteon_Lighting_Service:1#UDIService"

<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><control>DON</control><actio
n>117</action><flag>65531</flag><node>7 B0 A5
1</node></u:UDIService></s:Body></s:Envelope>
```

c. Turn Master Scene On Immediately (Fast On)

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 203
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udicom:
service:X_Insteon_Lighting_Service:1#UDIService"

<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><control>DFON</control><action></action><flag
>4</flag><node>34612</node></u:UDIService></s:Body></s:Envelope>
```

d. Increment Thermostat Setpoint

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 209
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udicom:
service:X_Insteon_Lighting_Service:1#UDIService"

<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><control>BRT</control><actio
n></action><flag>65531</flag><node>0 0 11
1</node></u:UDIService></s:Body></s:Envelope>
```

e. Start Fading (Up) Ceiling 1

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 212
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udicom:
service:X_Insteon_Lighting_Service:1#UDIService"

<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><control>BMAN</control><acti
on>1</action><flag>65531</flag><node>7 A5 95
1</node></u:UDIService></s:Body></s:Envelope>
```

For Fading Down, use <action>0</action>

f. Stop Fading Ceiling 1

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 211
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udicom:
```

```
service:X_Insteon_Lighting_Service:1#UDIService"

<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1"><control>SMAN</control><acti
on></action><flag>65531</flag><node>7 A5
951</node></u:UDIService></s:Body></s:Envelope>
```

5. Process Events/Notifications (see section 5)

ISY Events are of the form:

```
<?xml version="1.0"?><e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-
0"><e:property><ST>0</ST></e:property><e:property><node>9 89 8B
5</node><eventInfo>Event Specific elements</eventInfo></e:property>
</e:propertyset>
```

Where the first <property> element contains the Control which was changed, in this case ST (status) and the Action for this Control, in this case 0. The <node> element defines the node which was impacted.

8.5 Events

Events are very important feature of ISY and are published to subscribers. There can be a maximum of 10 simultaneous subscribers subscribing to ISY and all will get event notifications in real time. The following is a list of event definitions:

8.5.1 Device Status (control = Device Property)

action = Property value
Node = The address of the device

8.5.2 Heartbeat (control = "_0")

action = Duration in seconds
node = null
eventInfo = null

8.5.3 Trigger Events (control = "_1")

action = "0" -> **Event Status**

node = null
eventInfo =
<**eventInfo**>
<**id**>Hex representation of program id</**id**>
<**X/**> … X=**on** if enabled, and **off** if disabled
<**Y**/> … Y=**rr** if run at reboot and **nr** if not run at reboot
<**r**> last run time in YYMMDD HH:MM:SS</**r>**
<f> last finish time in YYMMDD HH:MM:SS</**f**>
<**s>** status* **</s>**
</**var**>
</**eventInfo**>

*Status is a bitwise OR of **RUN**_X and ST_X:
RUN_IDLE = 0x01
RUN_THEN = 0x02
RUN_ELSE = 0x03
ST_UNKNOWN = 0x10
ST_TRUE = 0x20
ST_FALSE = 0x30
ST_NOT_LOADED = 0xF0

action = "1" -> **Get Status (notifies subscribers to refresh)**

node = null
eventInfo = null

action = "2" -> **Key Changed**

node = key
eventInfo = null

action = "3" -> **Info String**

node = key
eventInfo = text

action = “4” -> **IR Learn Mode**

node = null
eventInfo = null

action = “5” -> **Schedule (schedule status changed)**

node = key
eventInfo = null

action = “6” -> **Variable Status (status of variable changed)**

node = key
<**eventInfo**>
<**var** id=”<var-id>” type =”<var-type>”>
<**val**>value</**val**>
<**ts**>YYYYMDD HH:MM:SS</**ts**>
</**var**>
</**eventInfo**>

action = “7” -> **Variable Initialized (initial value of a variable was set)**

node = key
<**eventInfo**>
<**var** id=”<var-id>” type =”<var-type>”>
<**init**>value</**init**>
</**var**>
</**eventInfo**>

action = “8” -> **Key**

This is sent after initial subscription and communicates the current key.

node = null
<**eventInfo**>
The Key
</**eventInfo**>

8.5.4 Driver Specific Events (control = "_2")

Depends on the underlying protocol driver

8.5.5 Node Changed/Updated (control = "_3")

action = "NN" -> **Node Renamed**

node = the address of the node
<**eventInfo>**
<**newName**>name</**newName**>
</eventInfo>

action = "NR" -> **Node Removed**

node = the address of the node removed
eventInfo = null

action = "ND" -> **Node Added**

node = the address of the node added
<**eventInfo>**
<**nodeName**>name</**nodeName**>
<**nodeType**>see section: **8.3.5.2Node (<node>)**</**nodeType**>
</**eventInfo**>

action = "MV" -> **Node Moved (into a scene)**

node = the address of the **scene** to which the node was moved to
<**eventInfo>**
<**movedNode**>the address of the moved node</**movedNode**>
<**linkType**>controller/responder</**linkType**> (see section: **8.3.5.3Group/Scene (<group>)**)
</**eventInfo>**

action = "CL" -> **Link Changed (in a scene)**

Not supported

action = “RG” -> **Removed From Group (scene)**

node = the address of the group/scene
<**eventInfo**>
<**removedNode**>the address of the removed node</**removedNode**>
</**eventInfo**>

action = “EN” -> **Enabled**

node = the address of the node that was enabled/disabled
<**eventInfo**>
<**enabled**>”true”|”false”</**enabled**>
</**eventInfo**>

action = “PC” -> **Parent Changed**

node = the address of the node whose parent changed
<**eventInfo**>
<**node**>the address of the node</**node**>
<**nodeType**>see section: **8.3.5.1Types of Nodes/Parents**</**nodeType**>
<**parent**>the address of tne new parent (NULL if none)</**parent**>
<**parentType**>s see section: **8.3.5.1Types of Nodes/Parents**</**parentType**>
</**eventInfo**>

action = “PI” -> **Power Info Changed**

node = the address of the node whose power info changed
<eventInfo>
<deviceClass> see section: **8.3.5.1Types of Nodes/Parents**</**deviceClass>**
<wattage> see section: **8.3.5.1Types of Nodes/Parents</wattage>**
<dcPeriod>s see section: **8.3.5.1Types of Nodes/Parents</dcPeriod>**
</eventInfo>

action = “DI” -> **Device ID Changed**

Not implemented.

action = “DP” -> **Device Property Changed**

UPB only

action = “GN” -> **Group Renamed**

node = the address of the group
<**eventInfo>**
<**newName**>name</**newName**>
</eventInfo>

action = “GR” -> **Group Removed**

node = the address of the group removed
eventInfo = null

action = “GD” -> **Group Added**

node = the address of the node added
<**eventInfo>**
<**groupName**>name</**groupName**>
<**groupType**>see section: **8.3.5.2Node (<node>)**</**groupType**>
</**eventInfo**>

action = “FN” -> **Folder Renamed**

node = the address of the folder
<**eventInfo>**
<**newName**>name</**newName**>
</eventInfo>

action = “FR” -> **Folder Removed**

node = the address of the group removed
eventInfo = null

action = “FD” -> **Folder Added**

node = the address of the folder added
eventInfo=null

action = “NE” -> **Node Error (Comm. Errors)**

node = the address of the node with communications errors

action = “CE” -> **Clear Node Error (Comm. Errors Cleared)**

node = the address of the node with communications errors

action = “SN” -> **Discovering Nodes (Linking)**

node = null

action = “SC” -> **Node Discovery Complete**

node = null

action = “WR” -> **Network Renamed**

node = the new name for network

action = “WH” -> **Pending Device Operation**

node = the address of the node for which there’s pending operations

action = “WD” -> **Programming Device**

node = the address of the node to which programming/write operations are being carried out

action = “RV” -> **Node Revised (UPB)**

Indicates the node may have been drastically changed therefore the UI should throw away any information about the node and reconstruct/repopulate it with the new information

node = the address of the node to which programming/write operations are being carried out
<**eventInfo**>standard node structure</**eventInfo**>

8.5.6 System Configuration Updated (control = “_4”)

action = “0” -> **Time Changed**

action = “1” -> **Time Configuration Changed**

action = “2” -> **NTP Settings Updated**

action = “3” -> **Notifications Settings Updated**

action = “4” -> **NTP Communications Error**

action = “5” -> **Batch Mode Updated**

node = null
<**eventInfo>**
<status>”1”|”0”**</status>**
</**eventInfo>**

action = “6” -> **Battery Mode Programming Updated**

node = null
<**eventInfo>**
<status>”1”|”0”**</status>**
</**eventInfo>**

8.5.7 System Status Updated (control = “_5”)

node = null

action = “0” -> **Not Busy**

action = “1” -> **Busy**

action = “2” -> **Idle**

action = “3” -> **Safe Mode**

8.5.8 Internet Access Status (control = “_6”)

action = “0” -> **Disabled**

action = “1” -> **Enabled**

node = null
<**eventInfo>**external URL<**/eventInfo>**

action = “2” -> **Failed**

8.5.9 Progress Report (control = “_7”)

node = null
eventInfo = text

action = “1” -> **Update**

action = “2.1” -> **Device Adder Info (UPB Only)**

action = “2.2” -> **Device Adder Warn (UPB Only)**

action = “2.3” -> **Device Adder Error (UPB Only)**

8.5.10 Security System Event (control = “_8”)

node = null
eventInfo = null

action = “0” -> **Disconnected**

action = “1” -> **Connected**

action = “DA” -> **Disarmed**

action = “AW” -> **Armed Away**

action = “AS” -> **Armed Stay**

action = “ASI” -> **Armed Stay Instant**

action = “AN” -> **Armed Night**

action = “ANI” -> **Armed Night Instant**

action = “AV” -> **Armed Vacation**

8.5.11 System Alert Event (control = “_9”)

Not implemented and should be ignored

8.5.12 OpenADR and Flex Your Power Events (control = “_10”)

**Only applicable to ISY994 Z Series.*

action = “1” -> **Open ADR Error**

node = null
eventInfo = null

action = “2” -> **Open ADR Status Update**

node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
    oadr:OpenADRState
</**eventInfo**>

action = “8” -> **OpenADR Registration Status**

node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
    oadr:OADRRegistration
</**eventInfo**>
*Profile 2b only

action = “9” -> **OpenADR Report Status**

node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
oadr:OADRReport
</**eventInfo**>
*Profile 2b only

action = “10” -> **OpenADR Opt Status**

node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
//not implemented
</**eventInfo**>
*Profile 2b only

action = “5” -> **Flex Your Power Error**

node = null
eventInfo = null

action = “6” -> **Flex Your Power Status Updated**

node = null
<**eventInfo**>
<**active**>whether or event has been issued</**active**>
</**eventInfo**>

8.5.13 Climate Events (control = “_11”)

*Requires WeatherBug Module

action = “0” -> **Error**

node = null
eventInfo = null

action = “1” -> **Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “2” -> **Temperature High**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “3” -> **Temperature Low**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “4” -> **Feels Like**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “5” -> **Temperature Rate**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “6” -> **Humidity**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “7” -> **Humidity Rate**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “8” -> **Pressure**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “9” -> **Pressure Rate**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “10” -> **Dew Point**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “11” -> **Wind Speed**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “12” -> **Average Wind Speed**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “13” -> **Wind Direction**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “14” -> **Average Wind Direction**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
See *udievnts.xsd:ClimateCoverage* for permissible values
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “15” -> **Gust Wind Speed**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “16” -> **Gust Wind Direction**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “17” -> **Rain Today**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “18” -> **Ambient Light**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “19” -> **Light Rate**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
See *udievnts.xsd:ClimateIntensity* for permissible values
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “20” -> **Rain Rate**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
See *udievnts.xsd:ClimateWeatherCondition* for permissible values
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “21” -> **Rain Rate Max**

Deprecated as of 4.2.4

node = null
<**eventInfo**>
See *udievnts.xsd:ClimateCloudCondition* for permissible values
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “22” -> **Evapotranspiration**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “23” -> **Irrigation Requirement**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “24” -> **Water Deficit Yesterday**

node = null
<**eventInfo**>
<**value**>(Signed) The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “25” -> **Elevation**

node = null
<**eventInfo**>
<**value**>(Signed) The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “26” -> **Coverage**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateCoverage* for permissible values
</**value**>
</**eventInfo**>

action = “27” -> **Intensity**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateIntensity* for permissible values
</**value**>
</**eventInfo**>

action = “28” -> **Weather Condition**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateWeatherCondition* for permissible values
</**value**>
</**eventInfo**>

action = “29” -> **Cloud Condition**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateCloudCondition* for permissible values
</**value**>
</**eventInfo**>

action = “30” -> **Tomorrow’s Forecast: Average Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “31” -> **Tomorrow’s Forecast: High Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “32” -> **Tomorrow’s Forecast: Low Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “33” -> **Tomorrow’s Forecast: Humidity**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “34” -> **Tomorrow’s Forecast: Wind Speed**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “35” -> **Tomorrow’s Forecast: Gust Speed**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “36” -> **Tomorrow’s Forecast: Rain**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “37” -> **Tomorrow’s Forecast: Snow**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “38” -> **Tomorrow’s Forecast: Coverage**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateCoverage* for permissible values
</**value**>
</**eventInfo**>

action = “39” -> **Tomorrow’s Forecast: Intensity**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateIntensity* for permissible values
</**value**>
</**eventInfo**>

action = “40” -> **Tomorrow’s Forecast: Weather Condition**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateWeatherCondition* for permissible values
</**value**>
</**eventInfo**>

action = “41” -> **Tomorrow’s Forecast: Cloud Condition**

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateCloudCondition* for permissible values
</**value**>
</**eventInfo**>

action = “42” -> **24 Hour Forecast: Average Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “43” -> **24 Hour Forecast: High Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “44” -> **24 Hour Forecast: Low Temperature**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “45” -> **24 Hour Forecast: Humidity**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “46” -> **24 Hour Forecast: Rain**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “47” -> **24 Hour Forecast: Snow**

node = null
<**eventInfo**>
<**value**>The value for this event</**value**>
<**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

action = “48” -> 24 Hour Forecast: Coverage

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateCoverage* for permissible values
</**value**>
</**eventInfo**>

action = “49” -> 24 Hour Forecast: Intensity

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateIntensity* for permissible values
</**value**>
</**eventInfo**>

action = “50” -> 24 Hour Forecast: Weather Condition

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateWeatherCondition* for permissible values
</**value**>
</**eventInfo**>

action = “51” -> 24 Hour Forecast: Cloud Condition

node = null
<**eventInfo**>
<**value**>
See *udievnts.xsd:ClimateCloudCondition* for permissible values
</**value**>
</**eventInfo**>

action = "100" -> **Last Updated Timestamp**

node = null
<**eventInfo**>
<**value**>
Timestamp of the last successful query
</**value**>
</**eventInfo**

8.5.14 AMI/SEP Events (control = "_12")

Only applicable to ISY994 Series!
See section: **14 ISY Energy Management System Developer's Manual**

8.5.15 External Energy Monitoring Events (control = "_13")

ISY994 Series currently support Brultech ECM1240 Energy Monitors.
Note: In ISY994 Series, all these events are incorporated into node events and no longer valid

action = "1" -> **Number of Channels**

node = null
<**eventInfo**>
<**numChannels**>The number of monitored channels</**numChannels**>
</**eventInfo**>

action = "2" -> **Channel Report**

node = null
<**eventInfo**>
<**channel** num="channel num" sampling=" ">
<power unit="unit">Instantaneous</power>
<total unit="unit">Total accumulative</total>
<voltage unit="unit">voltage</voltage>
<current unit="unit">current</current>
<polarized unit="unit">polarized power</polarized>
</**channel**>
</**eventInfo**>

action = “3” -> **Zigbee Status**

**Only applicable to ISY994 Z Series.*

node = null
<**eventInfo**>
"Down " |
"Establishing PAN" |
"Established"
</**eventInfo**>

Down = Zigbee Network Down
Establishing PAN = Trying to Establish Network
Established = Network Established and Operational

action = “7” -> **Raw Packet**

node = null
<**eventInfo**>
[![CDATA]
Raw binary packet directly from Brultech]
</**eventInfo**>

8.5.16 UPB Linker Events (control = “_14”)

Only applicable to UPB enabled units.

action = “1” -> **Device Status**
Status info for a device being linked

action = “2” -> **Pending Stop Find**
Device Adder Stop Find Pending

action = “3” -> **Pending Cancel Device Adder**
Device Adder Cancel Device Adder Pending

8.5.17 UPB Device Adder State (control = "_15")

8.5.18 UPB Device Status Events (control = "_16")

Only applicable to UPB enabled units.

action = "1" -> **Device Signal Report**

action = "2" -> **Device Signal Report Removed**

8.5.19 5.17 Gas Meter Events (control = "_17")

Only applicable to ISY994 Series.

action = "1" -> **Status**

node = null
<**eventInfo**>
<**total**>The actual meter value</**total**>
<**lastReadTS**>Timestamp of the last read</**lastReadTS**>
</**eventInfo**>

action = "2" -> **Error**

node = null
eventInfo = null

8.5.20 5.18 Zigbee Events (control = "_18")

Only applicable to ISY994 Series.
Namespace: **udiobjects (udiobjs.xsd)**

action = "1" -> **Status**
node = null
<**eventInfo**>
Zigbee.xsd:ZigbeeStatus
</**eventInfo**>

8.5.21 ELK Events (control = "_19")

Only applicable when ELK Module is installed.
Consult section: **10 ISY ELK Integration Developer's Manual**

8.5.22 Device Linker Events (control = "_20")

Device Linking events are all defined in **udievnts.xsd**.

action = "1" -> **Status**

node = null
<**eventInfo**>DeviceLinkerEventInfo</**eventInfo**>

action = "2" -> **Cleared**

Device linking list is now cleared
node = null
eventInfo = null

8.5.23 Z-Wave Events (control = "_21")

Only applicable when Z-Wave Module is installed.
Consult section: **11 ISY Z-Wave Integration Developer's Manual**

8.5.24 Billing Events (control = "_22")

Only applicable to ISY994 ZS Series!
See section: OR-WS-SDK-Manual-Energy Management.pdf. ***DEIDRICH***

8.5.25 Portal Events (control = "_23")

In case a portal module is installed, these events are published based on:

1. Portal socket connection status
2. Portal account registration status

See ISY-Portal-Integration-Manual.pdf. ***DEIDRICH***

8.6 REST Interface

REST is an easy to use URL based command set which allows the developer to communicate and control ISY.

Except for uploading configuration files, all REST commands use HTTP GET method.

8.6.1 Batch Commands

/rest/batch
Returns the Batch mode:

```
<batch><status>[0|1]</status></batch>
```

/rest/batch/on
Turns on Batch mode. Does not write changes to device. Only internal configuration files are updated

/rest/batch/Off
Turns off Batch mode. Writes all pending changes to devices and no longer buffers changes

/rest/batteryPoweredWrites
Returns the status of Battery Powered device operations

```
<batteryPoweredWrites>
    <status>[0|1]</status>
</batteryPoweredWrites>
```

/rest/batteryPoweredWrites/on
Writes all pending changes to battery powered devices when Batch mode is off

/rest/batteryPoweredWrites/off
Does not write changes to battery powered devices when batch is off

8.6.2 Configuration

For schema, please review the WSDL

/rest/config

Returns the configuration of the system with permissible commands

/rest/sys
Returns system configuration

/**rest/time**
Returns system time

/rest/network
Returns network configuration

/rest/subscriptions
Returns the state of subscriptions

8.6.3 Nodes

For schema, please review the WSDL

/rest/nodes
Returns nodes, scenes, types, and their status

/rest/nodes/devices
Returns only devices and their status but no scenes

/rest/nodes/scenes
Returns only scenes but no devices

/rest/nodes/<node-id>
Returns all the attributes & property values for a specific node

/rest/nodes/<node-id>?members=true|false
Works on a scene only. Using members=true includes all the scene members in the result

8.6.4 X10

/rest/X10/<Housecode[Unitcode]>/<X10 command>
UnitCode and X10 command are both optional

8.6.5 Properties

/rest/nodes/<node-id>/<property>
Returns the specific property value for a given node id

/rest/nodes/<node-id>/set/<property>/<value>
Insteon - OL/128 – Set On-Level to 50%
UPB - OL/50 – Set On-Level to 50%

/rest/nodes/<node-id>/write
Writes all pending changes to the device

/rest/nodes/<nodeid>/
cmd/<command_name>/<param1>/<param2>/.../<param5>
eg:
/rest/nodes/<node-id>/cmd/DOF - turn off a device or a scene
Insteon - /rest/nodes/<node-id>/cmd/DON/128 - turn on a scene to 50%
UPB - /rest/nodes/<node-id>/cmd/DON/50 - turn on a scene to 50%

/rest/nodes/<node-id>/enable
Enables a device

/rest/nodes/<node-id>/disable
Disables a device

/rest/nodes/<node-id>/notes
Returns the notes for the node

8.6.6 Status

/rest/status
Returns the status for all the nodes

/rest/status/<node-id>
Returns the status for the given node

8.6.7 Query

/rest/query
Queries all the nodes

/rest/query/<node-id>
Queries the given node

8.6.8 Programs

/rest/programs/<pgm-id>/<pgm-cmd>
e.g. /rest/program/0032/runThen -- Runs a command for a single program

/rest/programs/<pgm-id>
Returns single program, or folder with folders/programs within it

/rest/programs/<pgm-id>?folderContents=false
Returns single program or folder

/rest/programs/<pgm-id>?subfolders=true
Returns single program, or folder with folders/programs within it recursively

/rest/programs
Returns all the programs in the root folder e.g. same as /rest/programs/<rootpgm-id>

/rest/programs?folderContents=false
Returns root folder only (same as /rest/programs/<root-pgmid>?folderContents=false)

/rest/programs?subfolders=true
Returns all programs & folders (same as /rest/programs/<root-pgmid>?subfolders=true)

Defaults:
folderContents=true, subfolders=false

pgm-cmd:
run|runThen|runElse|stop|enable|disable|enableRunAtStartup|disableRunAtStartup
'runIf' is supported as well, but 'run' should be used instead.

8.6.9 Modules

/rest/electricity
**Only applicable to 994 Z Series.*

Returns electricity module info and specifically Energy Monitor, Open ADR and Flex Your Power status

/rest/climate –
Returns climate module info

/rest/networking/resources
Returns the networking resources configuration

/rest/networking/resources/<resource_id>
Calls and executes net resource

/rest/networking/wol
Returns the networking Wake On LAN configuration

/rest/networking/wol/<wol_id>
Calls and executes the WOL resource

8.6.10 Security

Please see section: **10 ISY ELK Integration Developer's Manual** documentation for complete integration of ELK security features with ISY.

8.6.11 Energy Management AMI/Smart Grid/SEP

Please see the Energy Management section. ***DEIDRICH***

8.6.12 Gas

*Only available on 992

/rest/gmeter
Returns the status of the gas meter

/rest/gmeter/log
Returns gas meter log

/rest/gmeter?reset=true
Clears all gas meter log entries

8.6.13 Logs

/rest/log
Returns system/event log

/rest/log?reset=true
Clears all system log entries

/rest/log/error
Return error log

/rest/log/error?reset=true
Clears all error log entries

8.6.14 Variables

/rest/vars/init/<var-type>/<var-id>/<value>
Sets the initial value of variable at ISY startup
var-type:

1. Integer
2. State

/rest/vars/set/<var-type>/<var-id>/<value>
Sets a variable given by var-id
var-type:

1. Integer
2. State

/rest/vars/get/<var-type>/<var-id>
Retrieves a variable given by var-id
var-type:

1. Integer
2. State

/rest/vars/get/<var-type>
Retrieves all variables of the given type
var-type:

1. Integer
2. State

Schema (*udiobjects:Variable & udiobjects:Variables*):
<**var** id=”<var-id>” type =”<var-type>”>
<**val**>value</**val**>
<**init**>init-value</**init**>
<**ts**>YYYYMDD HH:MM:SS</**ts**>
</**var**>
ts is the last changed timestamp of the current value

/rest/vars/definitions/<var-type>
Retrieves all variable definitions of the given type
var-type:

1. Integer
2. State

Schema (*udiobjects:VarDefinition & udiobjects:VarDefinitions*):
<**CList type**="VAR_INT">
<**e id**="<var-id>" **name**="<var-name>" />
…
<**e id**="<var-idN>" **name**="<var-nameN>" />
</**CList**>

type The low level data type (integer)
var-id The variable id
var-name The variable name

8.6.15 Zigbee

**Only applicable to ISY994 Z Series.*

/rest/zb
Retrieves the status of Zigbee Network including Joined nodes if any
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeStatus

/rest/zb/scanNetwork
Sends a broadcast to all nodes already in the PAN so that they would announce themselves again. This is a good diagnostics tool for orphaned nodes
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

/rest/zb/ntable
Retrieves the neighbor table for the COO. This is a good diagnostics tool for retrieving LQI for each node in the PAN
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

/rest/zb/nodes
Retrieves all the nodes which have joined the PAN
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeNodes

/rest/zb/nodes/[euid]
Retrieves information for the node with the given euid (joined only)
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeNode

/rest/zb/nodes/[euid]/ping
Pings the node with the given euid (joined only)
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeNode if successful, udiobjects.RestResponse otherwise

/rest/zb/nodes/[euid]/remove
Removes the node with the given euid (joined only) from the PAN
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

/rest/zb/nodes/[euid]/ep
Retrieves the active endpoints for a node
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

8.6.16 ELK

**Only applicable when ELK Module is installed.*

/rest/elk/…
Please refer to

8.6.17 Z-Wave

**Only applicable when Z-Wave Module is installed.*

/rest/zwave/…
Please refer to section: **10 ISY ELK Integration Developer's Manual**

8.6.18 Subscription using Web Sockets

/rest/subscribe/…
Please refer to section: **8.9 Web Sockets and Subscriptions** for more details.

8.7 Programs

You can setup and configure ISY programs using either SOAP or REST commands.

1. You can use the REST commands to retrieve programs or their status (see section: **8.6.8 Programs**)
2. You can update programs by POSTing a well-defined (see below for example) XML to /program/upload/ID?key=KEY
    - Where ID is the 4 digit hexadecimal representation of the program ID you retrieved in #1
    - KEY is used to let ISY know if there are concurrent changes (see below for more details).

Warning: please note that if you upload bad programs or bad keys you might render ISY inoperable

8.7.1 Program IDs

1. It is the clients responsibility (i.e. your responsibility) to manage program IDs for the full set of programs and folders
2. Every program has a unique ID from 1-1024
3. When a program is deleted its program ID is reused
4. Folders are programs and have a program ID (note: These are different from the folders used for organizing your Insteon devices on the main device page)
5. All programs reference a parent folder by program ID
6. The root folder has a parent of 0
7. The root folder may have any program ID, there is no guarantee it will be 1

8.7.2 Key

1. All changes to programs are scoped to the entire set of programs by a Key. In other words, if you change one thing it's like you've changed everything
2. When saving/deleting programs you do the following:
   - Create a new key, using "D2DCommand" "setKey". You must create the new key string: <timestamp>.<random number>
   - Save all new and changed folders and programs one at a time
   - Save the program: "/program/upload/<4_digit_hexadecimal_program_id>?key=<current key>"
   - Enable or Disable the program using "D2DCommand" "**enable**"/"**disable**"
   - Remove all deleted folders and programs, using "D2DCommand" "delete"
   - Create another new key as in step 1
   - Inform all other clients of the change by sending "D2DCommand" "broadcast". When the admin console receives one of these messages, it reloads the programs

Note: Programs may be implicitly deleted using save, for example, You delete program id=7 and then create a new program assigning it the newly available id=7, when it comes time to "Save Changes", you just save program id=7 and the other will implicitly be deleted.

8.7.3 Details

To send commands to programs you must first get all the programs. This gives you:

- A key value
- The content of all programs, allowing you to map between program name and program ID

The key-value ensures the client is using a current version of the programs. Whenever programs are added/changed/deleted, the key value changes, and thus all clients must get all the programs again.

To get all programs, issue the u:GetAllD2D command or you could use the REST equivalent (see section: **8.6.8 Programs**)

The XML you get back contains all the programs:

```
<key>key-value</key>
<triggers>
    <d2d>
        <trigger><id>1</id>
        <name>Living Room Program</name>
        ...
        </trigger>
    </d2d>
    <d2d>
        <trigger><id>3</id>
        <name>Kitchen Off</name>
        ... </trigger>
    </d2d> ...
</triggers>
```

To run a command for a program, use the u:D2DCommand, with the following content:
(Note: you do not need to specify the <mask> tag)

```
<u:D2DCommand xmlns:u="urn:udicom: service:X_Insteon_Lighting_Service:1">
    <id>pgm-id</id>
    <key>key-value</key>
    <CDATA><cmd>cmd-tag</cmd></CDATA>
</u:D2DCommand>
```

You can send the following commands:

```
<enable />
<disable />
<runif />
<runthen />
<runelse />
<runAtReboot />
<notRunAtReboot />
<stop />
```

eg. The following runs the 'Then path of program 'Kitchen Off' using the keyvalue of 914CEA.6967A0 returned from GetAllD2D

```
<u:D2DCommand xmlns:u="urn:udicom: service:X_Insteon_Lighting_Service:1">
    <id>3</id>
    <key>914CEA.6967A0</key>
    <CDATA><cmd><runthen /></cmd></CDATA>
</u:D2DCommand>
```

8.7.4 Examples

Setting the Key - Sets the current key
Use the <setKey> subcommand of <D2DCommand>
<key> - The current key
<setKey> - The new key

e.g.
<s:Envelope><s:Body><u:D2DCommand xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1">
<key>FFA019C6.7C7E</key>
<CDATA><cmd><setKey>FFA0A130.7C7E</setKey></cmd></CDATA>
</u:D2DCommand></s:Body></s:Envelope>

Deleting a program
Use the <delete /> subcommand of <D2DCommand>
<id> - The program ID (not in hexadecimal)
<key> - The current key
e.g.
<s:Envelope><s:Body><u:D2DCommand xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1">
<id>10</id>
<key>FFA0A130.7C7E</key>
<CDATA><cmd><delete /></cmd></CDATA>
</u:D2DCommand></s:Body></s:Envelope>

Enabling a program
Use the <enable /> subcommand of <D2DCommand>
<id> - The program ID (not in hexadecimal)
<key> - The current key
e.g.
<s:Envelope><s:Body><u:D2DCommand xmlns:u="urn:udicom:
service:X_Insteon_Lighting_Service:1">
<id>10</id>
<key>FFA0A130.7C7E</key>
<CDATA><cmd><enable /></cmd></CDATA>
</u:D2DCommand></s:Body></s:Envelope>

Saving a program (used for both creating and changing a program)
Use /program/upload/<4_digit_hexadecimal_program_id>?key=<current key>
e.g. This saves program id=10 (hexadecimal 000A):
https://isy.url/program/upload/000A?key=FFABF47E.7C7E

8.8 Logs

You can retrieve logs by using REST commands (see section **8.6.13 Logs**)
/rest/log – retrieves system log
/rest/log/error – retrieves error log

All logs are tab delimited with an new line (\n) at the end of each line.

8.8.1 System Log (/rest/log)

System log has the following format:
Node – the address of the node to which the log belongs
Control – the control that was impacted and which caused the log entry
Action – the value of the control
Time – in NTP format with epoch of 36524
UID* – the user or task which initiated the event

{
SYSTEM_USER=0 | *SYSTEM_DRIVER_USER*=1 | *WEB_USER*=2,
SCHEDULER_USER=3 | *D2D_USER*=4, *ELK_USER*=5 |
SEP_DEVICE_UMETER_USER=6 | *SEP_DEVICE_UPRICE_USER* |
SEP_DEVICE_UMSG_USER | *SEP_DEVICE_UDR_USER* |
GAS_METER_USER
}

Log Type – the type of entry
To clear System Log, use **/rest/log/reset=true**.
* *udiobjs.xsd:SystemActor*

8.8.2 Error Log (/rest/log/error)

Error log has the following format:
Time – in NTP format with epoch of 36524
UID* – the user or task which initiated the event

{
SYSTEM_USER=0 | *SYSTEM_DRIVER_USER*=1 | *WEB_USER*=2,
SCHEDULER_USER=3 | *D2D_USER*=4, *ELK_USER*=5 |
SEP_DEVICE_UMETER_USER=6 | *SEP_DEVICE_UPRICE_USER* |
SEP_DEVICE_UMSG_USER | *SEP_DEVICE_UDR_USER* |
GAS_METER_USER
}

Log Type – the type of entry
Error Message – free form text
To clear Error Log, use **/rest/log/error/reset=true**.
* *udiobjs.xsd:SystemActor*

8.8.3 Converting NTP Formatted Time

For efficiency, Time in the log is an unsigned integer (4 Bytes) formatted using NTP with the following parameters:

EPOCH_OFFSET = 36524 (01/01/200)
SEC_IN_DAY=86400
EPOCH_YEAR = 2000
EPOCH_DAY = 1
EPOCH_MONTH = 1
YEAR_STARTS_WITH_DAY=1

There are very many code libraries that support conversion from NTP/UTC time to a Time structure. This said, the following code snippet should get you started

```
/**
* Returns a <code>DateTime</code> object from an NTP
* representation of date/time
* @param cv - the NTP representation of date/time
* @param bt - the <code>DateTime</code> object which is
* returned (may be empty)
* @param epochOffset - the offset to be used from epoch ;
* USE EPOCH_OFFSET
* @return a <code>DateTime</code> object converted from NTP
*/
public static DateTime FromNTP( long cv, DateTime bt, long epochOffset )
{

int w;
long x = epochOffset*SEC_IN_DAY;
long tv = ( cv - x );
long tmp = 0;

bt.dow = (int)( ( ( cv / SEC_IN_DAY ) + 1 ) % 7 );

for ( w = EPOCH_YEAR; tmp <= tv; w++ )
{
    tmp += DAYS_IN_YEAR( w ) * SEC_IN_DAY;
}
```

```
        w--;

        tmp -= DAYS_IN_YEAR( w ) * SEC_IN_DAY;

        bt.year = w;
        tv -= tmp; /* Now we are ready for days */

        bt.day_of_year = (int)( tv / SEC_IN_DAY ) ;

        //+ YEAR_STARTS_WITH_DAY ) %
        // ( IsLeap( bt.year ) ? 366 : 365 );

        tv = tv % SEC_IN_DAY;

        bt.hour = (int) tv / 3600;
        bt.min = (int)( tv / 60 ) % 60;
        bt.sec =(int) tv % 60;

        bt = FixMonthDay( bt );

        if (YEAR_STARTS_WITH_DAY == 1)
            bt.day_of_year++;
        return bt;
    }
    /**
    * Fixes the day/month combination
    * @param bt - a <code>DateTime</code> object
    * @return the fixed <code>DateTime</code> object
    */
    public static DateTime FixMonthDay( DateTime bt )
    {
        bt.month = 0;
        bt.day = 0;
        long dn = bt.day_of_year;

        if ( IsLeap( bt.year ) )
        {
            if ( dn == 59 )
            {
                bt.month = 2;
                bt.day = 29;
```

```
                return bt;
            }
            else if ( dn > 59 )
            {
                dn--;
            }
        }
        /* Now we find the month */
        for ( bt.month = 1; bt.month < 12 && monthdays[bt.month + 1]
            <= dn; bt.month++ )
            ;

        bt.day = (int) dn - monthdays[bt.month] + 1;
        /* Month starts with 1 not 0 */
        return bt;
    }
    /**
    * Returns whether or not the year is a leap year
    * @param year - the year
    * @return - true if leap year, false otherwise
    */
    public static boolean IsLeap( int year )
    {
    return ( ( year % 4 ) == 0 );
    }
```

8.8.4 Log/Error Types

All Log related objects are defined in:
Schema: **log.xsd** and **udiobjs.xsd (***uo:LogInfo***)**; namespace=**ulog**
Instances:

1. **loginfo.xml** : defines all log information and messages
2. **insterr.xml** : sub definition for INSTEON error messages
3. **upberr.xml**: sub definition for UPB error messages
4. **smtperr.xml**: sub definition for SMTP error messages
5. **netstat.xml**: sub definition for Network Layer error messages

8.9 Web Sockets and Subscriptions

Starting with firmware version 4.2.3, you can now use REST to subscribe to ISY using Web Sockets.

Web Sockets are persistent HTTP/S connections that can be used in HTML5 and Javascript.

For more information on Web Sockets:
RFC6455: http://datatracker.ietf.org/doc/rfc6455/?include_text=1

The process is quite simple:

1. Use a Web Socket request to **/rest/subscribe**
2. Ensure that the following headers are included
    a. Authorization. Refer to section: **8.4 Communicating with ISY** for more details.
    b. **Sec-WebSocket-Protocol**: ***ISYSUB***
    c. **Sec-WebSocket-Version**: ***13***
    d. **Origin: *com.universal-devices.websockets.isy***
3. The initial return is the subscription response which includes the subscription id
4. ISY will continue publishing events to the client as long as the socket remains open. Refer to section: **8.5 Events** for more details on events.

8.10 Appendix A – INSTEON Device Categories/Subcategories

8.10.1 A1. Device Categories

All system wide and high level device categories are defined in:
Schema: **family.xsd**; namespace=**ufamily**
Instance: **cat.xml**

8.10.2 A2. Device Sub-Categories

Device sub-categories are defined in the respective family instances:
Schema: **family.xsd**; namespace=**ufamily**
Instance: **[familyId]_fam.xml**
e.g. INSTEON Family ID=1 -> **1_fam.xml**

8.11 Appendix B – UPB Device Types

Device types are stored in the type field of a node and are in the form <MID>.<PID>.<firmware_level>, where the MID is the manufacturer ID, the PID is a product ID within a MID.

8.11.1 B1. PCS Device Types

PCS Device Types currently supported:
WS1D,WS1E,WS1N,WMC6,DTC6,WMC8,DTC8,AM1,FMR1,KPC6,KPLR6,KPLR8,KPLD6,KPLD8,KPC8,LM1,LM2,FMD2,OCM,ICM2,DBM,ICM,SPR,TPR

PIDs for PCS (MID == 1)

```
WS1D 1    // Wall Switch - 1 Channel - Dimmer
WS1N 2    // Wall Switch - 1 Channel - Non-Dimmer
WMC6 3    // Wall Mount Controller - 6 Button
WMC8 4    // Wall Mount Controller - 8 Button
CRM 5     // Controlled Receptacle Module
OCM 6     // Output Control Module - 2 Channel
LCM1 7    // Load Control Module - 1 Channel
LCM2 8    // Load Control Module - 2 Channel
LM1 9     // Lamp Module - 1 Channel
LM2 10    // Lamp Module - 2 Channel
ICM2 11   // Input Control Module - 2 Channel
DTC6 13   // Desktop Controller - 6 Button
DTC8 14   // Desktop Controller - 8 Button
AM1 15    // Appliance Module - 1 Channel
CLC1 16   // Commercial Lighting Controller
CFC 17    // Commercial Fixture Controller
RRS 18    // Phase Repeater Slave
RRM 19    // Phase Repeater Master
MSC 20    // Motion Sensor Controller
HFC 21    // HID Fixture Controller
RNM 22    // Repeater Master Chip
RNS 23    // Repeater Slave Chip
WS1E 24   // Wall Switch - 1 Channel - Electronic low voltage Dimmer
LSM 25    // Load Shedding Module
TEC 26    // Timed-Event Controller
RFC 27    // Radio-Frequency Input Controller
RNC 28    // Repeater Network Controller
XUB 29    // X10-to-UPB Bridge
```

```
VHC 32    // Vacuum Handle Controller
VPC 33    // Vacuum Power Controller
VIM 34    // Vacuum Interrupt Module
VPM 35    // Vacuum Pan Module
DBM 36    // Doorbell Module
TCM 37    // Telephone Control Module
SPR 40    // Split-Phase Repeater
PIM 41    // Powerline Interface Module
DIAG1 49  // Diagnostics 1
DIAG2 50  // Diagnostics 2
DIAG3 51  // Diagnostics 3
DIAG4 52  // Diagnostics 4
TX4 53    // Test Transmitter 4
TM4 54    // Test Module 4
UTM 55    // UPB Test Module
ATM 56    // UPB Alpha Test Module
RIM 57    // Repeater Powerline Interface Module
FMD2 60   // Inline Fixture Dimmer Module - 2 Channel
FRM 61    // Inline Fixture Relay Module
WS2D 62   // Wall Switch Dimmer - Bar Graph
KPLD6 63  // Controller
SCM 64    // Shade Control Module
KPC6 65   // Controller
KPC8 66   // Controller
KPLD8 69  // 8-Button Keypad Light Dimmer
```

9 ISY Node Server Developer's Manual

REST Interface, based on firmware 5.0.4

9.1 Introduction

ISY is an award-winning platform for automation and energy management. With the introduction of Node Servers, the ISY now supports any protocol implemented by a third party in much that same way that INSTEON, Z-Wave and Zigbee are supported.
The concepts remain the same. The big difference is that instead of the ISY generating the events and running device commands, the node server does.

9.2 What is a Node?

A node represents all, or a subset of, a physical device such as lamp, switch and keypad, smoke detector, etc., or a conceptual device such as weather information or even stock quotes.

A node definition is used to describe a node. It contains the list of status values it maintains (e.g. the current temperature, heat/cool setpoints for a thermostat), the list of commands it accepts (e.g. on/off for a dimmer lamp), and a list of the commands it may send out (e.g. on/off for a dimmer switch).

A node server simply defines the set of nodes its supports, and provides the REST services to support them.

9.3 Node Server Configuration on ISY

9.3.1 Files

/editor	Contains all the XML editors files (.xml)
/nodedef	Contains all the XML node definitions (.xml)
/nls	Contains all the NLS properties files (.txt)
/version.txt	Contains the version of these files

These files are normally supplied as a .zip file by the node server developer and installed by the user through the ISY Admin console. In each directory, one or more files may be used. All filenames are restricted to 8.3 format.

If the node server developer creates a new version of the files, they can be installed over the old ones on the ISY. It is up to the node server developer to ensure any required backwards compatibility of nodes.

e.g. Example Zip File contents
/editor/edit.xml
/nodedef/ndef.xml
/nls/EN_US.txt
/version.txt

/editor

An editor defines the parameters for a widget in the client, such as a combobox, a numeric field etc. It defines the set of values and the unit(s) of measure available. An editor may contain multiple <range> entries, each of which must have a unique UOM.

/nodedef

A node definition defines the status and commands available to a node.

/nls

A single NLS file is used for each language. The naming convention is <language>_<countryCode>.txt (e.g. en_US.txt for USA English)
NLS is a set of name/value pairs used to display values in natural language (such as English).

9.3.2 Network Connection

9.3.2.1 From Isy to Node Server

The REST API is used to communicate with a node server when using a network connection. The ISY uses basic authentication with either http or https to communicate with the node server. A custom base URL is also prepended to the REST command, allowing the node server to customize the location of its REST support.

For example, if a base URL of /nodeservers/joe is configured, then the following URL would be sent to the node server to query a node:

/nodeservers/joe/nodes/<nodeAddress>**/query**

Having a base URL also allows a device to support multiple node servers, each with its own unique base URL.

9.3.2.2From Node Server to Isy

The node server must use basic authentication with either http or https to

communicate with the ISY. It must also know the profile number the node server has been assigned on the ISY because most REST API calls require this number in the URL. The ISY uses the profile number to ensure only the nodes owned by the profile can be modified, and to choose the ISY user number the node server should be using.
For example, if the node server has been assigned profile number 5, then something like the following URL would be used to update device status in ISY:

/rest/ns/5/nodes/n005_dimmer_2/report/status/ST/25.2/percent

9.3.2.3Responses

When a Node Server receives a REST command, one of the following responses must be sent out immediately, before processing the request. The ISY will send a similar response after processing a request.

200 - HTTP_OK
Valid request received, will run it

404 - HTTP_NOT_FOUND
Unrecognized request received and ignored.

503 - HTTP_SERVICE_UNAVAILABLE
Valid request received but ignored because system too busy to run it

If the userid/password is missing or incorrect

401 - HTTP_UNAUTHORIZED
User authentication failed

9.3.3 Serial Connection

Support may be added at a later time for node servers using serial connections.

9.4 Required API support in Node Server

9.4.1 General

Each node server is required to support a set of APIs that the ISY will use to manage the nodes being supplied by the node server. Primarily, these APIs are used to add/delete/rename nodes, send commands to nodes, and request node information. Other APIs request the node server to install or upgrade itself on

the ISY, and generally manage the configuration of the node server.

9.4.1.1Request IDs

<base>/….[?requestId=<requestId>]

On most API calls, the ISY can optionally supply a requestId. If a requestId appears on the URL then the node server must send a success or fail message back to the ISY after it has completed the requested action, and, after all messages from that completed action have been sent to the ISY.

This allows the ISY to run a command synchronously. For example, the ISY may need to query a device and use the results of the query to do some additional processing.

9.4.1.2Node Addresses

All node addresses are given a prefix assigned by the ISY. The prefix is unique to the node server thus guaranteeing that all node addresses on the ISY are unique.

The format of the node address prefix is:

naaa_

Where aaa is the profile number assigned to the node server in the ISY. A node address is made up of any combination of lowercase letters, numbers, and '_' character.

A node address for profile 5 could look something like:

n005_dimmer_3_1

The **dimmer_3_1** portion of the node address is completely defined by the node server or the user creating the node.

The maximum node length (including the prefix) is 19 characters.

9.4.2 Install

<base>/install/<profileNumber>

Instructs the node server to install all the profile files for the node server (rather than having the user do it through the ISY admin console). This is done by removing the old files and then adding all the files one by one, as follows:

- /rest/ns/<profileNumber>/profile/remove
- For each file:
    - /rest/ns/<profileNumber>/profile/upload/<dir>/<filename>
- /rest/ns/<profileNumber>/profile/reload

NOTE: In the current implementation, the ISY must be restarted for the new files to take effect.

9.4.3 Query node

<base>/nodes/<nodeAddress>/query[?requestId=<requestId>]

The node server must query the specified node, and send the results to the ISY using the Report Status Rest command.

If a requestId is specified, the status of the request must be sent to the ISY after all other messages are sent.

If a <nodeAddress> of “0” is specified, then all nodes must be queried.

9.4.4 Get Node Status Values

<base>**/nodes/<**nodeAddress>**/status[?requestId=<**requestId>]

The node server sends the current status values for the specified node to the ISY using the Report Status Rest command.

If a requestId is specified, the status of the request must be sent to the ISY after all other messages are sent.

If a <nodeAddress> of **“0”** is specified, then statis for all nodes must be sent.

9.4.5 Add All Nodes

<base>/add/nodes[?requestId=<requestId>]

Instructs the node server to add all of its nodes to the ISY (see Node Management).

If a requestId is specified, the status of the request must be sent to the ISY after all other messages are sent.

9.4.6 Reports from ISY

Reports provided by the ISY give the node server an opportunity to update its own database of nodes.

<base>/nodes/<nodeAddress>/report/add/<nodeDefId>?primary=<node Address>&name=<nodeName>

- Reports to the node server that the given node was added to the ISY.

<base>/nodes/<nodeAddress>/report/remove

- Reports to the node server that the given node was removed from the ISY.

<base>/nodes/<nodeAddress>/report/rename?name=<nodeName>

- Reports to the node server that the given node was renamed in the ISY.

<base>/nodes/<nodeAddress>/report/enable

- Reports to the node server that the given node was enabled in the ISY.

<base>/nodes/<nodeAddress>/report/disable

- Reports to the node server that the given node was disabled in the ISY.

NOTE: In the future, there may be additional APIs added that allow the node server more control over the actual creation and modification of nodes.

9.4.7 Run a command

<base>**/nodes/**<nodeAddress>**/cmd/**<command>
<base>**/nodes/**<nodeAddress>**/cmd/**<command>/<value>
<base>**/nodes/**<nodeAddress>**/cmd/**<command>/<value>/<uom>

[?<p1>.<uom1>=<val1>&<p2>...]**[requestId=**<requestId>]

The node server must run the specified command for the specified node. This command may have originated from an ISY program, the standard ISY REST API, the admin console, or any other client. The commands normally sent are those listed in the <accepts> section of the node definition used for the given node.

The numeric value of the UOM is always supplied and is never one of the common names. For example, **51** will be used instead of '**percent**'. For parameters in the <pX>.<uomX> format, the numeric uom value is always prefixed by '**uom**'

If a requestId is specified, the status of the running the command must be sent to the ISY after the command has completed or failed.

nodeAddress	The full address of the node (e.g. 'n005_switch_1)
command	The command to perform (e.g. 'DON', 'CLISPH', etc.)
pN	Nth Parameter name (e.g. 'level')
uomN	Unit of measure of the Nth parameter (e.g. 'uom58')
valN	The numeric value of the Nth parameter (e.g. '80', '80.01' etc,)

Commands may also have an unnamed parameter

value	The value of the unnamed parameter.
uom	Unit of measure of the value of the unnamed parameter (e.g. 51)

E.g.
/myserver/nodes/n005_switch_1/report/cmd/DON
/myserver/nodes/n005_switch_1/report/cmd/DON/80/51
/myserver/nodes/n005_switch_1/report/cmd/DON?level.uom51=80
/myserver/nodes/n005_switch_1/report/cmd/DON/80/percent?rate.uom58=0.3

9.5 REST support in ISY

REST is an easy to use URL based command set which allows the developer to communicate with the ISY.

Unless otherwise specified, all REST commands use HTTP GET method.

If no Response is provided, then UDIDefaultResponse must be assumed:

WSDL:zw:UDIDefaultResponse

Notes:

- URL Prefix**: /rest/ns/**<profileNumber>/
- The profileNumber specified on the URL determines which ISY userid/password will be accepted by the ISY for the request.

9.5.1 Reporting status updates

/nodes/<nodeAddress>**/report/status/**<driverControl>/<value>/<uom>

Updates the ISY with the current value of a driver control (e.g. the current temperature, light level, etc.)

nodeAddress	The full address of the node (e.g. 'n005_dimmer_1')
driverControl	The name of the status value (e.g. 'ST', 'CLIHUM', etc.)
value	The numeric status value (e.g. '80.5')
uom	Unit of measure of the status value

E.g. /rest/ns/5/nodes/n005_dimmer_2/report/status/ST/25.2/percent

9.5.2 Reporting a command

/nodes/<nodeAddress>**/report/cmd/**<command>
/nodes/<nodeAddress>**/report/cmd/**<command>/<value>
/nodes/<nodeAddress>**/report/cmd/**<command>/<value>/<uom>

[?<p1>.<uom1>=<val1>&<p2>.<uom2>=<val2>&<p3>…]

Sends a command to the ISY that may be used in programs and/or scenes. A common use of this is a physical switch that somebody turns on or off. Each time the switch is used, a command should be reported to the ISY. These are

used for scenes and control conditions in ISY programs.

nodeAddress	The full address of the node (e.g. 'n005_switch_1)
command	The command to perform (e.g. 'DON', 'CLISPH', etc.)
pN	Nth Parameter name (e.g. 'level')
uomN	Unit of measure of the Nth parameter (e.g. 'seconds', 'uom58')
valN	The numeric value of the Nth parameter (e.g. '80', '80.01' etc,)

Commands may also have an unnamed parameter

value	The value of the unnamed parameter.
uom	Unit of measure of the value of the unnamed parameter

E.g.
/rest/ns/5/nodes/n005_switch_1/report/cmd/DON
/rest/ns/5/nodes/n005_switch_1/report/cmd/DON/80/percent
/rest/ns/5/nodes/n005_switch_1/report/cmd/DON?level.percent=80
/rest/ns/5/nodes/n005_switch_1/report/cmd/DON/80/percent?rate.uom58=0.3

9.5.3 Node Management

/nodes/<nodeAddress>/add/<nodeDefId>**?primary**=<primary>&**name**=<nodeName>

Adds a node to the ISY. To make this node the primary, set primary to the same value as *nodeAddress*

nodeAddress	The full address of the node (e.g. 'n005_dimmer_1')
nodeDefId	The id of the node definition to use for this node
primary	The primary node for the device this node belongs to
nodeName	The name of the node
nls	(Optional) NLS key string for information specific to this node

E.g.
/rest/ns/5/nodes/n005_dimmer_2/add/MyDimmer?primary=n005_dimmer_1&name=Dimmer 2&nls=012B

/add/nodes

Sends a request to the node server to have it add all of its nodes to the ISY. This API is intended for ISY clients, and is never used by a node server.

E.g. /rest/ns/5/add/nodes

/nodes/*<nodeAddress>***/change/***<nodeDefId>****[?nls=****<nlsKey>****]***

Changes the node definition to use for an existing node. An example of this is may be to change a thermostat node from Fahrenheit to Celsius.

nodeAddress	The full address of the node (e.g. 'n005_dimmer_1')
nodeDefId	The id of the node definition to use for this node
nls	*(Optional)* NLS key string for information specific to this node

E.g. /rest/ns/5/nodes/n005_tstat_1/change/ThermostatCelsius?nls=4511

/nodes/<nodeAddress>**/remove**

Removes a node from the ISY. A node cannot be removed if it is the primary node for at least one other node.

nodeAddress	The full address of the node (e.g. 'n005_dimmer_1')

E.g. /rest/ns/5/nodes/n005_dimmer_2/remove

9.5.4 Reporting ISY Request status

/report/status/<requestId>**/fail**
/report/status/<requestId>**/success**

When the ISY sends a request to the node server, the request may contain a 'requestId' field. This indicates to the node server that when the request is completed, it **must** send a fail or success report for that request. This allows the ISY to in effect, have the node server synchronously perform tasks. This message must be sent after all other messages related to the task have been sent.

For example, if the ISY sends a request to query a node, all the results of the query must be sent to the ISY before a fail/success report is sent.

requestId The request ID the ISY supplied on a request to the node server.

E.g. /rest/ns/5/report/request/1234/success

9.6 Natural Language Support (NLS)

9.6.1 General

NLS support is defined for a node server by the set of properties files in the /nls subdirectory. They contain the name/value pairs used by the clients and the ISY to display commands, values, controls etc.. All NLS names must be in uppercase.

A naming convention is used to organize these values.

9.6.2 Naming Convention Terminology

The following table shows the various attributes from XML node definitions and editors that are used in this chapter to describe how to build the name of a particular NLS value.

<***node.nls***>	The 'nls' attribute specified when adding or changing a node. e.g. /rest/ns/5/nodes/n005_dimmer_2/add/MyDimmer?primary=n005_dimmer_1&name=Dimmer 2&**nls=012B**
<***nodedef.id***>	The 'id' attribute of a node definition. e.g. <nodeDef **id="Thermostat"** nls="tstat">
<***nodedef.nls***>	The 'nls' attribute of a node definition. e.g. <nodeDef id="Thermostat" **nls="tstat"**>
<***editor.id***>	The 'id' attribute of an editor e.g <editor **id="I_OL"**>
<***range.nls***>	The 'nls' attribute of a range e.g. <range uom="25" subset="0-32" **nls="IX_I_RR"** />
<***st.id***>	The 'id' attribute of a status e.g. <st **id="CLIHUM"** editor="I_TSTAT_HUM" />
<***cmd.id***>	The 'id' attribute of a command e.g. <cmd **id="DON"**>
<***p.id***>	The 'id' attribute of a command parameter e.g. <p **id="COLOR"** editor="I_COLOR_RGB" />

9.6.3 Device Name

The same node definition may be used for different products/models of a device. For example, there may be many different models of a dimmer lamp, but they are functionally equivalent and therefore use the same node definition. The device name is used to specify the actual product name/model etc. of the device for a specific node.

DEV-<node.nls>-NAME

e.g.

DEV-0102-NAME = (2475D) In-LineLinc Dimmer

9.6.4 Icons

The format and lookup order of the NLS entry for icons is:

DEV-<node.nls>-ICON
NDN-<nodedef.nls>-ICON
ND-<nodedef.id>-ICON

e.g.

DEV-0341-ICON = Thermostat
NDN-TStat-ICON = Thermostat
ND-MyThermostat-ICON = Thermostat

See **9.6.4 Icons** for the list of supported icons

9.6.5 Status Names

Some status values require different names for different node definitions. For example, ST for a dimmer should show up as 'Lamp', but ST for a drapery motor should show up as 'Drapes'. The format and lookup order of the NLS entry is:

ST-<nodedef.nls>-<st.id>-NAME
GEN-<nodedef.nls>-<st.id>-NAME
ST-<st.id>-NAME

e.g.

ST-ST-NAME = Lamp
ST-MYDRAPES-ST-NAME = Drapes

9.6.6 Command Names

The format and lookup order of the NLS entry for command names is:

CMD-<nodedef.nls>-<cmd.id>-NAME
CMD-<cmd.id>-NAME

e.g.

CMD-DON-NAME = On
CMD-MYDRAPES-DON-NAME = Open

9.6.7 Command Parameter Names

The format and lookup order of the NLS entry for command parameter names is:

GEN-<p.nls>-NAME
CMDP-<nodedef.nls>-<editor.id>-<p.id>-NAME
CMDPN-<nodedef.nls>-<p.id>-NAME
GEN-<nodedef.nls>-<p.id>-NAME
CMDP-<editor.id>-<p.id>-NAME
CMDP-<p.id>-NAME

e.g.
GEN-MYTIMER001-NAME = On/Off Timer

9.6.8 Other Names

node definition ND-<nodedef.id>-NAME

9.6.9 Name mapped Values (Index, Percent)

Some integer values may be displayed as names instead of numeric values. Index values (uom 25), and some percent values are commonly made into names. For example, displaying the values 0-31 for Insteon Ramp Rates is not very meaningful compared to names indicating the actual durations. 'On' and 'Off' are often displayed for percentage values, while the remaining values 1-99 are usually displayed numerically.

The format of the NLS entry for mapped values is:

<range.nls>-<value>

e.g. Insteon Ramp Rates

<range id="I_RR" uom="25" subset="0-31" nls="IX_I_RR" />
IX_I_RR-0 = 9.0 minutes
IX_I_RR-1 = 8.0 minutes
...
IX_I_RR-31 = 0.1 seconds

9.6.10 Formatting in Programs

Each line of a program is formatted and displayed in different way. Custom formatting entries are used for node conditions and commands, as follows:

9.6.11 Commands

The format and lookup order of the NLS entry for program command entries is:

PGM-CMD-<nodedef.nls>-<cmd.id>-FMT
PGM-CMD-<cmd.id>-FMT

e.g. (All on one line)
PGM-CMD-DON-FMT = /level/${c}/to ${v}/ /ramprate// in ${v}/ /offtimer//, turn off ${v} later/

/<param.id>/param text if omitted/param text if not omitted/ [.. next parameter, …]

e.g. /level/${c}/to ${v}/ /ramprate// in ${v}/

/ First character defines what character to use as separator, normally '/' is used

param.id Id of the parameter (e.g. 'level')
Note: This is empty for an unnamed parameter

param text if omitted
String to show if the parameter was omitted

param text if not omitted
String to show if the parameter was specified

The string for parameter text supports the following variables:

${c}	Name of the command
${v}	Formatted value of the parameter (including UOM)
${vo}	Formatted value of the parameter (without UOM)
${uom}	Formatted UOM without the value
${op}	Operator used (conditions only)

9.6.11.1 Command Formatting Examples

Assume commands are for node 'MyDevice'

1) A command with three named parameters, '*num*', '*val*', '*len*'

/num//${c} Parameter ${v}*/ /val/* default*/ =* ${v}*/ /len//* (${v} bytes)*/*

The following program action line would be shown for:

${c} = "Config", *num*=1, *val*=20, and *len*=4:
[Config Parameter 1][= 20][(4 bytes)]
--> "Set 'MyDevice' Config Parameter 1 = 20 (4 bytes)"

${c} = "Config", *num*=5, *val*=25, and *len* omitted:
[Config Parameter 5][= 25][]
--> "Set 'MyDevice' Config Parameter 5 = 25"

${c} = "Device", *num*=5, *val* omitted, and *len*=2:
[Device Parameter 5][default][(2 bytes)]
--> "Set 'MyDevice' Device Parameter 5 default (2 bytes)"

2) A command with one unnamed parameter

//default/${v}/

value of unnamed param omitted
[default] --> "Set 'MyDevice' default"

value of unnamed param = 50 percent
[50%] --> "Set 'MyDevice' 50%"

3) A command with no parameters shows just the command name and does not require and PGM-xxxxx entry

DFON --> "Set 'MyDevice' Fast On"

Another example:

PGM-CMD-DON-FMT = /level/${c}/to ${v}/ /ramprate// in ${v}/ /offtimer//, turn off ${v} later/

level=50%, ramprate=3 seconds, offtimer=5 minutes
"[to 50%][in 3 seconds][, turn off 5 minutes later]"
--> "Set 'MyDevice' to 50% in 3 seconds, turn off 5 minutes later"

9.6.12 Status Conditions

The format and lookup order of the NLS entry for status condition format entries is:

PGM-ST-<nodedef.nls>-<st.id>-FMT
PGM-ST-<st.id>-FMT

The value is a ***single param*** text string similar to that specified for a command parameter.

e.g.
PGM-ST-CLISPH-FMT = ${c} ${op} ${v}

If not specified, then the following is used: ${c} ${op} ${v}

9.6.13 Control Conditions

There are no custom entries for control conditions, because currently, control conditions do not include any of the command parameters.

9.7 Appendix

9.7.1 Editors

```
<editors>
   <editor id="I_OL">
      <range uom="51" subset="0-100" />
      <range uom="56" subset="0-255" />
   </editor>
   <editor id=”TEMP_C”>
      <range uom=”4” min=”4.5” max=”32” step=”0.5” prec=”1” />
   </editor>
   <editor id="I_RR">
      <range uom="25" subset="0-32" nls="IX_I_RR_" />
   </editor>
</editors>
```

editor	id	The name of the editor Note: Name must not begin with ‘_’ (reserved for encoded Editor IDs) .
range	uom	The unit of measure of the value (See Units of Measure) *Note: Must be unique for each range entry in an editor*
	nls	Used for *percent (51)* and *index (25)* unit of measures only. The NLS prefix to use for the name of value. e.g. for nls=”IX_I_RR” value 5, the NLS entry would be: IX_I_RR-5 = 8 seconds
range (1)	subset	The subset of values supported defined as a set of ranges and individual values. They must be in increasing value with no duplicates or overlap. The values are limited to positive integers. Ranges are separated by a ‘-‘, individual digits are separated by a ‘,’. e.g. subset=“0-5,7,9,11-14” means these numbers: 0,1,2,3,4,5,7,9,11,12,13,14
range (2)	min	The minimum value (inclusive)
	max	The maximum value (inclusive)
	step	The number to increment with each step (e.g. in a spinner type widget)
	prec	The number of decimal places to keep for the value

9.7.2 Encoded Editor ID

For simple editors, rather than referencing an editor defined within an xml file, the ID itself can be encoded in a way that fully defines the editor. The following describes the supported encodings. An encoded editor ID may be used anywhere an editor is referenced (e.g. status, command parameters, etc.)

Editor ID Encoded Format	**Implied XML (by example)**
<uom><prec>	<editor id=”_17_1”> <range uom=”17” prec=”1” min=”-2147483647” max=”2147483647” /> </editor>
<uom><prec>_N_<nls>	<editor id=”_17_1_N_IXN”> <range uom=”17” prec=”1” min=”-2147483647” max=”2147483647” nls=”IXN” /> </editor>
<uom><prec>_R_<min>_<max> Note: ‘m’ is used to indicate a negative min/max value	<editor id=”_17_2_R_m5_10”> <range uom=”17” prec=”2” min=”-5” max=”10” /> </editor>
<uom><prec>_R_<min>_<max>_N_<nls> Note: ‘m’ is used to indicate a negative min/max value	<editor id=”_17_0_R_0_10_N_IXRR”> <range uom=”17” prec=”0” min=”0” max=”10” nls=”IXRR” /> </editor>
<uom><prec>_S_<lowMask>	<editor id=”_17_1_S_FF00FF00”> <range uom=”17” subset=”8-15,24-31” /> </editor>
<uom><prec>_S_<lowMask>_N_<nls>	<editor id=”_17_1_S_FF00FF00_N_IXN”> <range uom=”17” subset=”8-15,24-31” nls=”IXN” /> </editor>
<uom><prec>_S_<lowMask>_<highMask>	<editor id=”_17_1_S_FF00FF00_03E”> <range uom=”17” subset=”8-15,24-31,33-37” /> </editor>
<uom><prec>_S_<lowMask>_<highMask>_ N_<nls>	<editor id=”_17_1_S_FF00FF00_03E_N_IXN”> <range uom=”17” subset=”8-15,24-31,33-37” nls=”IXN” /> </editor>

9.7.3 Node Definitions

```
<nodeDefs>
   <nodeDef id="Thermostat" nls="143">
      <sts>
         <st id="ST" editor="I_TEMP_DEG" />
         <st id="CLISPH" editor="I_CLISPH_DEG" />
         <st id="CLISPC" editor="I_CLISPC_DEG" />
         <st id="CLIMD" editor="I_TSTAT_MODE" />
         <st id="CLIHCS" editor="I_TSTAT_HCS" />
         <st id=”ERR” editor=”I_ERR” hide=”T” />
      </sts>
      <cmds>
         <sends>
            <cmd id="DON" />
            <cmd id="DOF" />
         </sends>
         <accepts>
            <cmd id="CLISPH">
               <p id="" editor="CLISPH_DEG" init="CLISPH" />
            </cmd>
            <cmd id="CLISPC">
               <p id="" editor="CLISPC_DEG" init="CLISPC" />
            </cmd>
            <cmd id="CLIMD">
               <p id="" editor="T_MODE" init="CLIMD" optional=”T” />
            </cmd>
            <cmd id="QUERY" />
         </accepts>
      </cmds>
   </nodeDef>
</nodeDefs>
```

<***nodeDef***>	id	Name of this node definition (e.g. “DimmerSwitch”)
	nls	NLS key string used to override names of commands, status and other elements.
<***st***>	id	One of the predefined driver controls e.g. “CLISPH”
	editor	The id of the editor to use
<***sends***>		The commands this node can send out. Used for control conditions in ISY programs and scene controllers.
<***accepts***>		The commands this node accepts. Used for buttons etc. in ISY clients, and actions in ISY programs.
<***cmd***>	id	Name of a command.
<***p***>	id	Name of a command parameter. A command may have one unnamed parameter, all others must be named.
	editor	The id of the editor to use for this parameter
	init	(Optional) id of the status value this parameter should be initialized and/or synchronized with. For example, CLISPH is both a status and a command.
	optional	Set to “T” or “True” if this is an optional parameter
	nls	NLS key string used to override name of parameter.

9.7.4 Icons

The predefined icon names (images here are just for clarity, the actual icons may be different in various GUIs)

	DoorLock	A door lock
	Electricity	Generic Electricity
	EnergyMonitor	Energy Monitor
	GenericCtl	Generic controller
	GenericRsp	Generic Responder
	GenericRspCtl	Generic Responder and Controller
	Input	A sensor input
	Irrigation	Generic Irrigation
	Lamp	A lamp
	LampAndSwitch	Represents both a lamp and a switch
	MotionSensor	A motion sensor
	Output	An output relay
	Switch	A switch
	SmokeSensor	A smoke sensor
	TempSensor	A temperature sensor
	Thermostat	A thermostat
	Weather	Generic Weather

9.7.5 Driver Controls

The predefined driver controls.

ADRPST	Auto DR Processing State
AIRFLOW	Air Flow
ALARM	Alarm
ANGLPOS	Angle Position
ATMPRES	Atmospheric Pressure
BARPRES	Barometric Pressure
BATLVL	Battery level
BEEP	Beep
BMAN	Deprecated - Use FDUP or FDDOWN
BRT	Brighten
BUSY	Device is Busy
CC	Current Current
CLIEMD	Energy Mode
CLIFRS	Fan Running State
CLIFS	Fan Setting
CLIFSO	Fan Setting Override
CLIHCS	Heat/Cool State
CLIHUM	Humidity
CLIMD	Thermostat Mode
CLISMD	Schedule Mode
CLISPC	Cool Setpoint
CLISPH	Heat Setpoint
CLITEMP	Current Temperature
CO2LVL	CO2 Level
CPW	Current Power Used
CV	Current Voltage
DEWPT	Dew Point
DFOF	Fast Off
DFON	Fast On
DIM	Dim
DISTANC	Distance
DOF	Off
DON	On
ELECCON	Electrical Conductivity
ELECRES	Electrical Resistivity
ERR	Error
FDDOWN	Fade Down

FDSTOP	Fade Stop
FDUP	Fade Up
GPV	General Purpose Value
GV0	Custom Control 0
GV1	Custom Control 1
GV10	Custom Control 10
GV11	Custom Control 11
GV12	Custom Control 12
GV13	Custom Control 13
GV14	Custom Control 14
GV15	Custom Control 15
GV16	Custom Control 16
GV17	Custom Control 17
GV18	Custom Control 18
GV19	Custom Control 19
GV2	Custom Control 2
GV20	Custom Control 20
GV3	Custom Control 3
GV4	Custom Control 4
GV5	Custom Control 5
GV6	Custom Control 6
GV7	Custom Control 7
GV8	Custom Control 8
GV9	Custom Control 9
GVOL	Water Volume
LUMIN	Luminance
MOIST	Moisture
OL	On Level
PF	Power Factor
PPW	Polarized Power Used
PULSCNT	Pulse Count
QUERY	Query Device
RAINRT	Rain Rate
RESET	Reset values
ROTATE	Rotation
RR	Ramp Rate
SECMD	Device secure mode
SEISINT	Seismic Intensity
SEISMAG	Seismic Magnitude
SMAN	Deprecated - Use FDSTOP
SOILT	Soil Temperature

SOLRAD	Solar Radiation
SPEED	Velocity
ST	Status
SVOL	Sound Volume
TANKCAP	Tank Capacity
TIDELVL	Tide Level
TIMEREM	Time remaining
TPW	Total Power Used
UAC	Valid user access code entered
UOM	Unit
USRNUM	User number
UV	Ultraviolet
VOCLVL	Volatile Organic Compound (VOC) level
WATERT	Water Temperature
WEIGHT	Weight
WINDDIR	Wind Direction
WVOL	Water Volume

9.7.6 Units of Measure

The units of measure include the scientific units of measure as well as custom types. A unit of measure is a numeric type that fully defines a value. Values in square brackets are keywords that may be specified for the unit of measure instead of the numeric value when sending a request to ISY. The ISY will always return the numeric value for the unit of measure.

0 = Unit of measure is unknown
1 = ampere (amp) *[amp,ampere]*
2 = boolean
3 = btu/h [btuh]
4 = celsius (C) *[C, celsius]*
5 = centimeter (cm) [cm]
6 = cubic feet
7 = cubic feet per minute (cfm)
8 = cubic meter
9 = day
10 = days
11 = Deadbolt status *(See below)*
12 = decibel (db) *[db]*
13 = decibel A (dbA) *[dbA]*
14 = degree
15 = Door lock alarm *(See below)*

16 = european macroseismic
17 = Fahrenheit (F) *[F]*
18 = feet
19 = hour
20 = hours
21 = absolute humidity
22 = relative humidity
23 = inches of mercury (inHg)
24 = inches/hour
25 = index
26 = kelvin (K) *[K]*
27 = keyword
28 = kilogram (kg) *[kg]*
29 = kilovolt (kV)
30 = kilowatt (kW)
31 = kilopascal (kPa) *[kpa]*
32 = kilometers/hour (KPH)
33 = kilowatts/hour (kWH) *[kwh]*
34 = liedu
35 = liter (l)
36 = lux *[lux]*
37 = mercalli
38 = meter (m)
39 = cubic meters/hour
40 = meters/sec (m/s)
41 = milliamp (mA)
42 = millisecond (ms)
43 = millivolt (mV)
44 = minute
45 = duration in minutes
46 = millimeters/hour (mm/hr)
47 = month
48 = miles/hour (MPH)
49 = meters/second (MPS)
50 = ohm *[ohm]*
51 = percent
52 = pound
53 = Power Factor
54 = Parts/Million (PPM)
55 = pulse count
56 = The raw value as reported by the device

57 = second
58 = Duration in seconds
59 = seimens/meter
60 = body wave magnitude scale
61 = Richter scale
62 = moment magnitude scale
63 = surface wave magnitude scale
64 = shindo
65 = SML
66 = Thermostat heat/cool state *(See below)*
67 = Thermostat mode *(See below)*
68 = Thermostat fan mode *(See below)*
69 = US gallon
70 = User number
71 = UV index
72 = volt *[V, volt]*
73 = watt *[W, watt]*
74 = watts/square meter
75 = weekday
76 = Wind Direction in degrees
77 = year
78 = 0-Off 100-On
79 = 0-Open 100-Close
80 = Thermostat fan run state *(See below)*
81 = Thermostat fan mode override
82 = millimeter *[mm]*
83 = kilometer
84 = Secure Mode 0-Unlock 1-Lock
85 = Ohm Meter (Electrical resistivity)
86 = KiloOhm
87 = cubic meter/cubic meter
88 = Water activity
89 = rotations/Minute (RPM)
90 = Hertz (Hz)
91 = Angle Position degrees relative to North Pole
92 = Angle Position degrees relative to South Pole
93 = Power Management Alarm (See below)
94 = Appliance Alarm (See below)
95 = Home Health Alarm (See below)
96 = VOC Level (See below)
97 = Barrier Status (See below)

98 = Insteon Thermostat Mode (See below)
99 = Insteon Thermostat Fan Mode (See below)
100 = A Level from 0-255 e.g. brightness of a dimmable lamp
101 = Degree multiplied by 2 (for Insteon compatibility)
102 = Kilowatt Second (kWs)
103 = Dollar
104 = Cents
105 = Inches
106 = Millimeters per day
107 = Raw 1-byte unsigned value
108 = Raw 2-byte unsigned value
109 = Raw 3-byte unsigned value
110 = Raw 4-byte unsigned value
111 = Raw 1-byte signed value
112 = Raw 2-byte signed value
113 = Raw 3-byte signed value
114 = Raw 4-byte signed value

Special Values
11 = Deadbolt status
- 0 - Unlocked
- 100 - Locked
- 101 - Unknown
- 102 - Jammed

15 = Door lock alarm
- 1 - Master Code Changed
- 2 - Tamper Code Entry Limit
- 3 - Escutcheon Removed
- 4 - Key/Manually Locked
- 5 - Locked by Touch
- 6 - Key/Manually Unlocked
- 7 - Remote Locking Jammed Bolt
- 8 - Remotely Locked
- 9 - Remotely Unlocked
- 10 - Deadbolt Jammed
- 11 - Battery Too Low to Operate
- 12 - Critical Low Battery
- 13 - Low Battery
- 14 - Automatically Locked
- 15 - Automatic Locking Jammed Bolt

16 - Remotely Power Cycled
17 - Lock Handling Completed
19 - User Deleted
20 - User Added
21 - Duplicate PIN
22 - Jammed Bolt by Locking with Keypad
23 - Locked by Keypad
24 - Unlocked by Keypad
25 - Keypad Attempt outside Schedule
26 - Hardware Failure
27 - Factory Reset

66 = Thermostat heat/cool state
0 - Idle
1 - Heating
2 - Cooling
3 - Fan Only
4 - Pending Heat
5 - Pending Cool
6 - Vent
7 - Aux Heat
8 - 2nd Stage Heating
9 - 2nd Stage Cooling
10 - 2nd Stage Aux Heat
11 - 3rd Stage Aux Heat

67 = Thermostat mode
0 - Off
1 - Heat
2 - Cool
3 - Auto
4 - Aux/Emergency Heat
5 - Resume
6 - Fan Only
7 - Furnace
8 - Dry Air
9 - Moist Air
10 - Auto Changeover
11 - Energy Save Heat
12 - Energy Save Cool
13 - Away

68 = Thermostat fan mode

- 0 - Auto
- 1 - On
- 2 - Auto High
- 3 - High
- 4 - Auto Medium
- 5 - Medium
- 6 - Circulation
- 7 - Humidity Circulation

80 = Thermostat fan running state

- 0 - Off
- 1 - On
- 2 - On High
- 3 - On Medium
- 4 – Circulation
- 5 - Humidity Circulation
- 6 - Right/Left Circulation
- 7 - Up/Down Circulation
- 8 - Quiet Circulation

93 = Power Management Alarm

- 1 - Power Applied
- 2 - Ac Mains Disconnected
- 3 - Ac Mains Reconnected
- 4 - Surge Detection
- 5 - Volt Drop Or Drift
- 6 - Over Current Detected
- 7 - Over Voltage Detected
- 8 - Over Load Detected
- 9 - Load Error
- 10 - Replace Battery Soon
- 11 - Replace Battery Now
- 12 - Battery Is Charging
- 13 - Battery Is Fully Charged
- 14 - Charge Battery Soon
- 15 - Charge Battery Now

94 = Appliance Alarm
- 1 - Program Started
- 2 - Program In Progress
- 3 - Program Completed
- 4 - Replace Main Filter
- 5 - Failure To Set Target Temperature
- 6 - Supplying Water
- 7 - Water Supply Failure
- 8 - Boiling
- 9 - Boiling Failure
- 10 - Washing
- 11 - Washing Failure
- 12 - Rinsing
- 13 - Rinsing Failure
- 14 - Draining
- 15 - Draining Failure
- 16 - Spinning
- 17 - Spinning Failure
- 18 - Drying
- 19 - Drying Failure
- 20 - Fan Failure
- 21 - Compressor Failure

95 = Home Health Alarm
- 1 - Leaving Bed
- 2 - Sitting On Bed
- 3 - Lying On Bed
- 4 - Posture Changed
- 5 - Sitting On Edge Of Bed

96 = VOC Level
- 1 - Clean
- 2 - Slightly Polluted
- 3 - Moderately Polluted
- 4 - Highly Polluted

97 = Barrier Status
0 - Closed
1-99 - Percent Closed (1% = almost Closed, 99% = almost Open)
100 - Open
101 - Unknown
102 - Stopped
103 - Closing
104 – Opening

98 = Insteon Thermostat mode
0 - Off
1 - Heat
2 - Cool
3 - Auto
4 – Fan Only
5 – Program Auto
6 – Program Heat
7 – Program Cool

99 = Insteon Thermostat fan mode
7 – On
8 - Auto

10 ISY ELK Integration Developer's Manual

Web Services SDK and REST Interface, based on firmware 4.3.1

10.1 Introduction

ISY is an award-winning platform for automation and energy management. With the introduction of ELK Integration Module (21090), all ELK events can now be acted upon as well as published to clients. Furthermore, developers can use Web Services/REST interface for direct communications with the ELK system.

10.2 Getting Started

Communications and event infrastructure follow the same paradigm as those defined in ISY's WSDK Developer's guide. Additional events, Web Services, and REST interface are defined herein. If you have not yet reviewed ISY's WSDK Developer's guide, please send an email to sales@universal-devices.com.

If you do not already have ELK Integration module (21090) installed on ISY, please send an email to sales@universal-devices.com with your UUID (Help | About) or purchase the module through Help | Purchase Modules on the Admin Console.

Once you are successfully communicating with ISY and have ELK Module installed, then:

1. Go to http://isy:port/desc (or http://your.isy.ip.address:port/desc)

   You will be presented with the description of services provided by Orchestrator. In the <serviceList> element, look for **UDIELKWebServices** as the <serviceType>. What you are looking for is the URL for Web services binding. This URL is defined <SCPDURL> (see below):

```
<serviceList>
    <service>
        Default ISY Framework service descriptions
    </service>
    <service>
        <serviceType>UDIELKWebServices</serviceType>
        <serviceId>
            uuid:00:03:f4:03:65:96-UDIELKWebServices
        </serviceId>
        <SCPDURL>/elkServices.wsdl</SCPDURL>
        <controlURL>/security/elk</controlURL>
```

```
        </service>
    </serviceList>
```

2. Now, all you need to do is point your SOAP client to: http://your.isy.ip.address:port/[value for SCPDURL] and import ELK web services

3. Get ISY Configuration (GetISYConfig Web Service or /rest/config) and ensure that ELK Integration module is installed (id = 21090). For more information on Configuration Resources and Modules, please consult ISY WSDK Developer's Guide, section: 17 **Soap / Web Services (WSDK)**

4. All ELK web services are defined in the WSDL returned by step 2. All objects are defined in **elkobjs.xsd** and are imported into the WSDL

5. If you are using SOAP UI, you can immediately communicate and issue ELK services to ISY Please note that all requests require the Authorization header. Furthermore, if you wish to receive ELK events, please do make sure that you have subscribed to ISY (please consult WSDK Developer's Guide), section: 17 **Soap / Web Services (WSDK)**

For offline tests, the WSDL file is: ***udielk1.ws***

10.3 ELK Events (control = "_19")

In addition to all the events published by ISY framework, ELK module has its own set of events that are specific to ELK Security System.

*All events are defined in ***elkobjs.xsd***.

10.3.1 Topology Changed (action = "1")

Topology is the complete configuration for ELK and thus, in the case of this event, it's best to use corresponding Web Services (GetTopology) or REST (/get/topology) to retrieve the topology.

10.3.2 Area Event (action = “2”)

This event is published when something in a define Area changes. These include **Alarm** status changes, **Arming** up status changes, and **Armed** status changes.

EventInfo structure will be of the form:
<eventInfo>
    <**ae** …. />
</eventInfo>

Where **ae** is defined in:

Schema File	**elkobjs.xsd**
Namespace	**uelk**
Class	**ELKAreaEventInfo**

10.3.3 Zone Event (action = “3”)

This event is published when something in a define Zone changes. These include **Logical** status changes, **Physical** status changes, and **Voltage** changed.

EventInfo structure will be of the form:
<eventInfo>
    <**ze** …. />
</eventInfo>

Where **ze** is defined in:

Schema File	**elkobjs.xsd**
Namespace	**uelk**
Class	**ELKZoneEventInfo**

10.3.4 Keypad Event (action = “4”)

This event is published when something in a defined Keypad changes. These include **Access Code** status changes, **Key** status changes, and **LED** status changes.

EventInfo structure will be of the form:
<eventInfo>
    <**ke** …. />
</eventInfo>

Where **ke** is defined in:

Schema File	**elkobjs.xsd**
Namespace	**uelk**
Class	**ELKKeypadEventInfo**

10.3.5 Output Event (action = “5”)

This event is published when something in a defined Output changes. These include **Output** status changes.

EventInfo structure will be of the form:
<eventInfo>
    <**oe** …. />
</eventInfo>

Where **oe** is defined in:

Schema File	**elkobjs.xsd**
Namespace	**uelk**
Class	**ELKOutputEventInfo**

10.3.6 System Event (action = “6”)

This event is published in two cases: 1) ISY/ELK connection events and 2) Module enable/disable events.

EventInfo structure will be of the form:
<eventInfo>
    <**se** …. />
</eventInfo>

Where **se** is defined in:

Schema File	**elkobjs.xsd**
Namespace	**uelk**
Class	**ELKSystemEventInfo**

10.3.7 Thermostat Event (action = “7”)

This event is published in the case of thermostat changes of state.
EventInfo structure will be of the form:

<eventInfo>
    <**te** …. />
</eventInfo>

Where **te** is defined in:

Schema File	**elkobjs.xsd**
Namespace	**uelk**
Class	**ELKThermostatEventInfo**

10.4 REST Interface

REST is an easy to use URL based command set which allows the developer to communicate and control ELK services through ISY.

All REST commands use HTTP GET method.

If no Response is provided, then UDIDefaultResponse must be assumed:

WSDL:isyelk:UDIDefaultResponse

URL Prefix: /rest/elk/

10.4.1 Area Commands

areas/query
Queries all areas and changes to states are published through event infrastructure. see section:
10.3.2 Area Event (action = "2")

area/<*areaId***>/get/status**
Retrieves the status of the given area.

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**

Response	**elkobjs.xsd:uelk:AreaResponseType**

area/<*areaId***>/cmd/bypass?code=<***accessCode***>**
Bypasses all violated burglar alarms in the area given in areaId

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**
accessCode	Optional; Defined by **elkobjs.xsd:uelk:AccessCode**

area/<*areaId***>/cmd/unbypass?code=<***accessCode***>**
Unbypasses all burglar alarms in the area given in areaId

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**
accessCode	Optional; Defined by **elkobjs.xsd:uelk:AccessCode**

area/<*areaId*>/cmd/display?text=<*eText*>&beep=<boolean> &offTimerSeconds=<*seconds*>
Displays text given in eText to all keypad in the given area. Optionally beep and turn off after offTimerSeconds.

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**
eText	URL escaped text. 2 Lines with 16 characters per line max. Use '^' to separate lines if less than 16 characters. e.g. text="Hello^World" ***beep*** Optional; True then beep
beep	Optional; True then beep when displaying the message
seconds	Optional; Number of seconds to display the message

area/<*areaId*>/cmd/display?id=<*notifyId*>&beep=<boolean> &offTimerSeconds=<*seconds*>
Displays formatted text to all keypad in the given area given a custom content ID. This uses the same custom content entries as are used for email notifications. Optionally beep and turn off after offTimerSeconds.

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**
notifyId	The id of the customized content entry
beep	Optional; True then beep when displaying the message
seconds	Optional; Number of seconds to display the message

area/<*areaId*>/cmd/arm?armType=<*elkArmType*> &code=<*accessCode*>

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**
elkArmType	Defined by **elkobjs.xsd:uelk:ArmType**
accessCode	Optional; Defined by **elkobjs.xsd:uelk:AccessCode**

area/<*areaId*>/cmd/disarm?code=<*accessCode*>

areaId	Defined in **elkobjs.xsd:uelk:AreaIDType**
accessCode	Optional; Defined by **elkobjs.xsd:uelk:AccessCode**

10.4.2 Zone Commands

zones/query
Queries all zones and changes to states are published through event infrastructure. see section: **10.3.3 Zone Event (action = "3")**

zones/<*zoneId*>/query/voltage
Queries the zone given by zoneId and changes to states are published through event infrastructure. see section: **10.3.3 Zone Event (action = "3")**
To query voltage for all zones, specify a *zoneId* of 0 (zero).

zoneId	Defined in **elkobjs.xsd:uelk:ZoneIDType**

zone/<*zoneId*>/query/temperature
Queries the thermostats for the given zone and changes to states are published through event infrastructure. see section: **10.3.3 Zone Event (action = "3")**
To query temperature for all zones, specify a *zoneId* of 0 (zero).

zoneId	Defined in **elkobjs.xsd:uelk:ZoneIDType**

zone/<*zoneId*>/cmd/trigger/open
Momentarily triggers a zone to the physical state of *Open.* An error will occur if the zone is defined as normally open, or is currently open.

zoneId	Defined in **elkobjs.xsd:uelk:ZoneIDType**

zone/<*zoneId*>/cmd/toggle/bypass?code=<*accessCode*>
Toggles bypass for the zone given in zoneId

zoneId	Defined in **elkobjs.xsd:uelk:ZoneIDType**
accessCode	Optional; Defined by **elkobjs.xsd:uelk:AccessCode**

10.4.3 General Commands

query/all
Gets the status for all areas, zones, outputs, counters, etc. and publishes the state changes through event infrastructure. see section:**10.3 ELK Events (control = "_19")**

get/status
Returns the status of all of zones, areas, keypads, outputs, etc. configured in ELK.

Response	**elkobjs.xsd:uelk:ELKAllStatus**

get/topology
Returns the representation of all the zones, areas, keypads, outputs, etc. configured in ELK.

Response	**elkobjs.xsd:uelk:Topology**

refresh/topology
Causes ISY to recreate/refresh the topology. Returns the representation of all the zones, areas, keypads, outputs, etc. configured in ELK.

Response	**elkobjs.xsd:uelk:Topology**

10.4.3.1System Commands

system/get/status
Returns the enabled and connected status.

Response	**elkobjs.xsd:uelk:SystemResponseType**

10.4.4 Keypad Commands

keypad/<*kpId*>/cmd/press/funcKey/<*fkId*>
Simulates pressing the fkId button on keypad given in kId

kpId	Keypad number (1-8)
fkId	Defined in **elkobjs.xsd:uelk:FunctionKeyType**

keypad/<*kpId*>/query/temperature
Queries the status of temperature sensor on a keypad and changes to states are published through event infrastructure. see section: **10.3.3 Zone Event (action = "3")**
To query temperature for all keypads, specify a *kpId* of 0 (zero).

keypad/<*kpId*>/get/status
Retrieves the status of a keypad.
To get status for all keypads, specify a *kpId* of 0 (zero).

kpId	Keypad number (1-8)
Response	**elkobjs.xsd:uelk:KeypadResponseType**

Note: See Area Commands for displaying text on keypads

10.4.5 Keypad Commands

outputs/query
Queries all outputs and changes to states are published through event infrastructure. see section: **10.3.5 Output Event (action = "5")**

output/<*outputId*>/get/status
Returns the status of the given output
To get status for all outputs, specify an *outputId* of 0 (zero).

outputId	**elkobjs.xsd:uelk:OutputIDType**)
Response	**elkobjs.xsd:uelk:OutputResponseType**

output/<*outputId*>/cmd/on?offTimerSeconds=<*seconds*>
Turns On an output defined by outputId. Optional offTimerSeconds can be defined such that the output is turned back off after the amount of seconds given in <seconds>.

outputId	**elkobjs.xsd:uelk:OutputIDType**)
seconds	Optional; Interval after which the relay turns off

output/<*outputId*>/cmd/off
Turns Off an output defined by outputId

outputId	**elkobjs.xsd:uelk:OutputIDType**)

10.4.6 Audio Commands

Audio commands allow communications with A/V equipment that have been configured and attached to ELK.

audio/zone/<*audioZone***>/source/<***audioSource***>/cmd/<***audioCommand***>?value=<***audioValue***>**
Sends the command given in audioCommand to zone given in audioZone.

audioZone	Audio Zone number (1-18)
audioSource	Audio Source number (1-12)
audioCommand	Defined by **elkobjs.xsd:uelk:AudioCommandType**
audioValue	3-digit decimal Number, currently only used for Volume

10.4.7 Voice Announcement Commands

Voice Announcement commands cause predefined words or phrases to be spoken by the ELK security system.

speak/word/<*wordId***>**
Instructs ELK to speak the word given by wordId.

wordId	Defined by **elkobjs.xsd:uelk:VoiceWordType**

speak/phrase/<*phraseId***>**
Instructs ELK to speak the phrase given by phraseId.

phraseId	Defined by **elkobjs.xsd:uelk:VoicePhraseType**

10.4.8 Thermostat Commands

Thermostat commands impact a thermostat.

tstat/<*tstat_id***>/query**
Instructs ELK query the thermostat given by tstat_id and state changes are published as events. see section: **10.3.7 Thermostat Event (action = "7")**

tstat_id	Defined by **elkobjs.xsd:uelk:ThermostatIDType**

tstat/<*tstat_id***>/get/status**
Retrieves the status of the given thermostat.

tstat_id	Defined by **elkobjs.xsd:uelk:ThermostatIDType**
Response	**elkobjs.xsd:uelk:ThermostatResponseType**

tstat/<*tstat_id***>/cmd/<***cmd_id***>?value=***value***
Instructs ELK to apply the value using command defined by cmd_id to thermostat defined by tstat_id.

tstat_id	Defined by **elkobjs.xsd:uelk:ThermostatIDType**
cmd_id	Defined by **elkobjs.xsd:uelk:ThermostatCommandType**
value	Depends on the command: Temperatures (such as setpoints): 1-99 Mode: **elkobjs.xsd:uelk:ThermostatModeState** Fan: **elkobjs.xsd:uelk:ThermostatFanState**

11 ISY Z-Wave Integration Developer's Manual

Web Services SDK and REST Interface, based on firmware 5.0.2

11.1 Introduction

ISY is an award-winning platform for automation and energy management. With the introduction of Z-Wave Integration Module (21100), ISY can now support Z-Wave devices as well as an additional protocol (such as INSTEON or Zigbee).

The concepts remain the same. Controls are the same as those already defined in base ISY framework with only some additions. Events are also the same as those already defined in base ISY with one addition.

As such, this document will only list Z-Wave only additions. The rest are all captured in ISY-WS-SDK-Manual (.docx and .pdf).

11.2 Getting Started

Communications and event infrastructure follow the same paradigm as those defined in ***ISY's WSDK Developer's guide*** and the ***ISY Node Server Developer's Manual***. Additional events, Web Services, and REST interface are defined herein. If you have not yet reviewed these guides, please send download from http://www.universal-devices.com/isy-developers.
.
If you do not already have Z-Wave dongle/module installed on your ISY, please consult http://wiki.universal-devices.com/index.php?title=Z-Wave:_Ordering/Assembly_Instructions.

Once you are successfully communicating with ISY and have Z-Wave Module installed, then:

1. Go to http://isy:port/desc (or http://your.isy.ip.address:port/desc)

   You will be presented with the description of services provided by Orchestrator. In the <serviceList> element, look for **UDIZWaveWebServices** as the <serviceType>. What you are looking for is the URL for Web services binding. This URL is defined <SCPDURL> (see below):

   ```
   <serviceList>
       <service>
           Default ISY Framework service descriptions
       </service>
       <service>
   ```

```
            <serviceType>UDIZWaveWebServices</serviceType>
            <serviceId>
                uuid:00:03:f4:03:65:96-UDIZWaveWebServices
            </serviceId>
            <SCPDURL>/zwaveServices.wsdl</SCPDURL>
            <controlURL>/zwaveServices</controlURL>
        </service>
    </serviceList>
```

2. Now, all you need to do is point your SOAP client to: http://your.isy.ip.address:port/[value for SCPDURL] and import Z-Wave web services
3. Get ISY Configuration (GetISYConfig Web Service or /rest/config) and ensure that Z-Wave Integration module is installed (id = 21100). For more information on Configuration Resources and Modules, please consult ISY WSDK Developer's Guide, section: 17 **Soap / Web Services (WSDK)**
4. All Z-Wave web services are defined in the WSDL returned by step 2. All objects are defined in **zwobjs.xsd** and are imported into the WSDL
5. If you are using SOAP UI, you can immediately communicate and issue Z-Wave services to ISY

Please note that all requests require the Authorization header. Furthermore, if you wish to receive Z-Wave events, please do make sure that you have subscribed to ISY (please consult WSDK Developer's Guide), section: 17 **Soap / Web Services (WSDK)**

For offline tests, the WSDL file is: ***udizw1.ws (ISY)* or *udizw1.wsdl (Web).***

11.3 Z-Wave Control Events

Control events sent by Z-Wave device use two addition attributes on the **action** tag, as follows:

uom	The unit of measure for the action value
prec	The precision of the action value indicating the number of implied decimal points. E.g. <action uom="33" prec="2">12345</action> The actual value is 123.45

e.g.

```
<Event seqnum="95" sid="uuid:26">
    <control>TPW</control>
    <action uom="33" prec="2">12345</action>
    <node>ZW022_143</node>
    <eventInfo/>
</Event>
```

11.4 Z-Wave Events (control = "_21")

In addition to all the events published by ISY framework, Z-Wave module has its own set of events that are specific to Z-Wave Operations.

*All events are defined in ***zwobjs.xsd****.*

11.4.1 System Status Events (action = "1.3")

This event is published when something dongle status changes such as changes from primary to secondary or connection states.

EventInfo structure will be of the form:
<eventInfo>
    <**zwave** …. />
</eventInfo>

Where **zwave** is defined in:

Schema File	**zwobjs.xsd**
Namespace	**uzw**
Class	***ZWaveStatusEventObject***

11.4.2 Discovery - Inactive (action = "2.1")

This event is published when Z-Wave subsystem is not in discovery mode.

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.4.3 Discovery - Inclusion (action = "2.2")

This event is published when Z-Wave subsystem is in Inclusion mode.

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.4.4 Discovery - Exclusion (action = “2.3”)

This event is published when Z-Wave subsystem is in Exclusion mode.

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.4.5 Discovery - Primary Replication (action = “2.4”)

This event is published when Z-Wave subsystem is the current primary controller and replicates to another controller to make it the new Primary.

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.4.6 Discovery - Learn Mode (action = “2.5”)

This event is published when the Z-Wave subsystem is in learn mode

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.4.7 General Status (action = “3.x.y”)

Reports status, generally a successful completion of a task.

3.1.0	Backup of Z-Wave dongle succeeded
3.2.0	Restore of Z-Wave dongle succeeded
3.3.0	Network Heal completed
3.4.0	Repair Links completed

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.4.8 General Error (action = “4.x.y”)

Reports an error, generally a failure of a task.

4.1.0	Secure inclusion if Z-Wave device was prevented by ISY (See **/security/key/protect**)
4.2.0	Secure inclusion of Z-Wave device failed
4.3.0	Backup of Z-Wave dongle failed
4.3.1	- Backup/Restore System is busy
4.3.2	- Failed to read data from the dongle
4.3.3	- Failed to write backup file to file system
4.4.0	Restore of Z-Wave dongle failed
4.4.1	- Backup/Restore System is busy
4.4.2	- No Backup file available
4.4.3	- Invalid/corrupt Backup file
4.4.4	- Could not Factory Reset the Z-Wave dongle
4.4.5	- Failed to write data to the Z-Wave dongle

EventInfo structure will be of the form:
<eventInfo>
</eventInfo>

11.5 Common REST Interface

/rest/nodes/*<nodeAddress>*
Returns information describing the specified node.

nodeAddress	The ISY address of the Z-Wave node (e.g. “ZW012_1”)

Example response:
<**nodeinfo**>
…
<devtype>
<gen>4.64.3</gen>
<mfg>144.1.1</mfg>
<cat>111</cat>
<model>4</model>
</devtype>
…
<**/nodeinfo**>

devtype	gen	The *basic.generic.specific* Z-Wave type as reported by the device
	mfg	The M*anufacturerID.ProductTypeID.ProductID* as reported by the device (for devices that provide this information)
	cat	The node type defining the subset of the device functionality provided by this node (See Node Types, section: **11.7.3 Node Types**)
	model	A single number indicating the specific device model for some devices known by the ISY (e.g. A Schlage BE369 door lock) (See Model Types, section: **11.7.4 Model Types**)

/rest/nodes/<*nodeAddress*>/**cmd**
See '***Run a command***' in the section: **9.4.7 Run a command** in the '***ISY Node Server Developer's Manual***'

11.6 REST Interface

REST is an easy to use URL based command set which allows the developer to communicate and control Z-Wave services through ISY.

All REST commands use HTTP GET method.

If no Response is provided, then UDIDefaultResponse must be assumed:

WSDL:zw:UDIDefaultResponse

Notes:

- URL Prefix**: /rest/zwave**
- For optional values, the default value is shown in **bold**
- Objects are defined in **zwobjs.xsd**

11.6.1 Adding and Removing devices

/node/include?power=<*boolean***>&nwi=<***boolean***>**
Add a device into the Z-Wave network.

power	Optional,**True**=High Power, False=Normal Power
nwi	Optional,**True**=Use network wide inclusion, False= Use Standard Inclusion

/node/exclude
Remove a device from the Z-Wave network.

/node/cancel
Cancel include/exclude/replication.

/node/*<nodeAddress>***/remove**
Remove an unresponsive node from the Z-Wave network. If the node is responsive then it will not be removed.

nodeAddress	ISY node address of node to be removed e.g. "ZW005_1"

11.6.2 General Commands

/backup
Backs up the network information on the Z-Wave dongle to a file on the ISY.

/restore
Restores the Z-Wave dongle using the network information stored in the backup file on the ISY.

/sendPrimary
As Primary Controller, replicate to another controller and make it the new Primary Controller.

/learnMode
Go into Z-Wave learn mode (to replicate, be added/removed from Z-Wave network, etc.).

/security/key/protect
Prevents the ISY from sending any network keys to devices when they are included. This is a simple precaution that prevents the network keys from being transmitted to an unknown device that has been accidentally included (e.g. a neighbor puts his Z-Wave controller in learn mode after you put your ISY into include mode).

/security/key/unprotect
Allows the ISY to send network keys to devices when they are included. This is required, and is only required, when including secure devices.

/firmware/upgrade
Updates the Z-Wave dongle with new firmware.

/sync?[id=<*nodeAddress*>]|[uid=<*zwaveNodeId*>]

Synchronize ISY with info on Z-Wave dongle for the given node. Specify either id or uid. If neither is specified then sync all new & deleted nodes.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
zwaveNodeId	Optional, Z-Wave node id e.g. 5

/sync/full?[id=<*nodeAddress*>]|[uid=<*zwaveNodeId*>]
Synchronize ISY with info on Z-Wave dongle for the given node. Specify either id or uid. If neither is specified then sync all nodes.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
zwaveNodeId	Optional, Z-Wave node id e.g. 5

/heal/network?[id=<*nodeAddress*>]|[uid=<*zwaveNodeId*>]
Heal the Z-Wave network. Specify either id or uid. If neither is specified then heal the network communications between all nodes.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
zwaveNodeId	Optional, Z-Wave node id e.g. 5

/repair/links?[id=<*nodeAddress*>]
Remove dead links (associations) from the specified device, or all Z-Wave devices if *id* is not specified.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"

/factoryReset/dongle?[force=<*boolean*>]
Factory reset the Z-Wave.

force	Optional, True=Force factory reset, **False**=Factory reset if dongle is not part of existing Z-Wave network

/set/antenna/<antennaType>
Sets active antenna; switches between internal and external.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"

/get/version
Get the Z-Wave dongle version information.

Example response:
<**version bootloader**="1.02" **zwave**="4.55" **library**="Static Controller"/>

version	bootloader	Version of the low-level bootloader
	zwave	Z-Wave firmware version
	library	Name of the Z-Wave firmware library

/get/status
Get general Z-Wave dongle status information.

Example response:
<**zwave nodeid**="1" **primary**="true" **suc**="false" **sis**="false" **networkEmpty**="false" **connected**="true" **extAntenna**="true" />

zwave	nodeid	The Z-Wave node ID (sometimes referred to as unit id)
	primary	True then the dongle is the primary controller
	suc	True if the dongle is also a Static Update Controller (SUC)
	sis	True if the dongle is also a SUC ID Server (SIS)
	networkEmpty	True if the Z-Wave network is empty (i.e. no devices have been added to the network)
	connected	True if communications with the dongle are working
	extAntenna	True if the external antenna is being used instead of the internal antenna
	keyProtected	True if the network keys are protected during device inclusion

11.6.3 Node Information

/node/<*nodeAddress*>**/def/get**
Returns the node definition the specified node. If a node address of '**0**' is specified then it returns definitions for all Z-Wave nodes. See the '***ISY Node Server Developer's Manual***' for description of ISY node definitions.

nodeAddress	The ISY address of the Z-Wave node (e.g. "ZW012_1"), or 0 for all Z-Wave nodes

/node/0/def/get/key
Returns a key that can be used to determine if any of the node definitions have changed since they were last retrieved. This allows clients to cache the node definitions and only retrieve them again if the key has changed.

Example response:
<**nodedefkey**>
1000000007
</**nodedefkey**>

node/show[/links|/network|/all]?[id=<*nodeAddress***>]|[uid=<***zwa veNodeId***>]**

/show - Shows Z-Wave device details
/show/links - Shows associations
/show/network - Shows neighboring devices
/show/all - Shows all information (details, links, and network)

e.g. node/show/links?id=ZW005_1

Sends detailed information about the Z-Wave device as progress messages (i.e. messages that appear in the Admin Console event viewer). If no device is specified then information for all nodes currently in the Z-Wave network are shown.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
zwaveNodeId	Optional, Z-Wave node id e.g. 5

/node/neighbors?[id=<*nodeAddress***>]|[uid=<***zwaveNodeId***>]**

Returns the neighbors of the specified device. If no device is specified then neighbors for all nodes currently in the Z-Wave network are returned.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
zwaveNodeId	Optional, Z-Wave node id e.g. 5

Example response:
<**neighbors isyuid**="1">
    <**node id**="" **uid**="1">
        <**n id**="ZW010_1" **uid**="10"/>
        <**n id**="ZW012_1" **uid**="12" **rep**="Y"/>
    </**node**>
    <**node id**="ZW010_1" **uid**="10">
    ...
</**neighbors**>

neighbors	isyuid	Z-Wave node id of the Isy Dongle
node	id	Isy node address (e.g. "ZW005_1")
	uid	Z-Wave node id (e.g. 5)
n	id	Isy node address of this neighbor (e.g. "ZW007_1")
	uid	Z-Wave node id of this neighbor (e.g. 7)
	rep	**Y – neighbor is a repeater** **N** (or omitted) – neighbor is not a repeater

11.6.4 Device Configuration Commands

/node/<*nodeAddress*>**/config/set/**<*paramNum*>**/**<*value*>**[/**<*size*>**]**
Set a Z-Wave device configuration value

/node/<*nodeAddress*>**/config/query/**<*paramNum*>
Query a Z-Wave device configuration value

11.6.5 Node Properties

/node/<*nodeAddress*>**/set/time**
Synchronizes the current time in the specified device with that of the ISY.

/node/<*nodeAddress*>**/properties/set/**<*name*>**/**<value>
Sets the value of a property for a given node.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
name	Name of the property
value	New value for the property

/node/<*nodeAddress*>**/properties/query[/**<*name*>**]**
Queries the value of a property for a given node. If the name is not specified, then all properties from the node are queried.

nodeAddress	Optional, ISY Node address e.g. "ZW005_1"
name	Name of the property

Example response:

```
<ps>
   <p id="MODE">
      <val>0</val>
   </p>
   <p id="TIMEOUT">
      <val>1</val>
   </p>
</ps>
```

p	id	Name of the property
val		Value of the property

11.6.6 Schedule and Security Properties

Schedule and security property support, currently only applies to door locks.

/node/*<nodeAddress>***/security/user/***<userNum>***/set/code/***<code >*
Sets the access code for the specified user. Supported by all door locks.

nodeAddress	ISY Node address e.g. “ZW017_1” ”
userNum	Door Lock user number e.g. 3
code	The new access code e.g. 123456

/node/*<nodeAddress>***/security/user/***<userNum>***/set/role/***<role>*
Sets the user role for the specified user. Currently only supported by Kwikset locks.

nodeAddress	ISY Node address e.g. “ZW017_1” ”
userNum	Door Lock user number e.g. 3
role	The user role, either **owner**, **guest**, or **worker**.

/node/*<nodeAddress>***/security/user/***<userNum>***/enable/schedu le**
Enable the schedule for the specified user.

nodeAddress	ISY Node address e.g. “ZW017_1” ”
userNum	Door Lock user number e.g. 3

/node/<*nodeAddress*>**/security/user/**<*userNum*>**/disable/sched ule**
Disable the schedule for the specified user.

nodeAddress	ISY Node address e.g. "ZW017_1" "
userNum	Door Lock user number e.g. 3

/node/<*nodeAddress*>**/security/user/**<*userNum*>**/delete/schedul e/**<*slot*>
Remove the schedule for the given user. This also changes the user role to *Guest.*

e.g. node/ZW017_1/security/user/3/delete/schedule/1

nodeAddress	ISY Node address e.g. "ZW017_1" "
userNum	Door Lock user number e.g. 3
slot	The schedule slot to update. e.g. 1

node/<*nodeAddress*>**/security/user/**<*userNum*>**/set/schedule/**<
slot>**?start=**<*datetime*>**&stop=**<*datetime*>
Add or update a schedule for the given user. This also changes the user role to *Guest.* Currently only supported by Kwikset door locks.

e.g. node/ZW017_1/security/user/3/set/schedule/1?start=2014-05- 23T08:45&stop=2014-05-23T18:55

nodeAddress	ISY Node address e.g. "ZW017_1" "
userNum	Door Lock user number e.g. 3
slot	The schedule slot to update. Currently only slot 1 is supported by the door locks
start	The date and time the schedule starts. e.g. 2014-06-12T09:30
stop	The date and time the schedule stops. e.g. 2014-06-12T17:30

/node/<*nodeAddress*>**/security/user/**<*userNum*>**/set/schedule/**<
slot>**?start=**<*time*>**&stop=**<*time*>***&day=***<*dayOfWeek*>
Add or update a schedule for the given user. This also changes the user role to *Worker.*
Currently only supported by Kwikset door locks.

e.g. /node/ZW017_1/security/user/1/set/schedule/2?start=09:05&stop=14:15&day= fri

nodeAddress	ISY Node address e.g. "ZW017_1" "
userNum	Door Lock user number e.g. 3
slot	The schedule slot to update. Currently only slots 1 through 7 are supported by the door locks
start	The time the schedule starts e.g. 09:05
stop	The time the schedule stops e.g. 14:15
day	The day of the week the schedule runs. Possible values: **mon tue wed thu fri sat sun**

/node/<*nodeAddress*>**/security/user/**<*userNum*>**/set/schedule/**<*slot*>**?start=**<*time*>**&duration=**<*timeDuration*>***&day=***<*daysOfWe ek*>
Add or update a schedule for the given user. Currently only supported by Yale door locks.

e.g. /node/ZW012_1/security/user/3/set/schedule/1?day=SUN,WED&start=09:30&duration=03:00

nodeAddress	ISY Node address e.g. "ZW017_1" "
userNum	Door Lock user number e.g. 3
slot	The schedule slot to update. Currently only slot 1 is supported by the door locks
start	The time the schedule starts e.g. 09:30
stop	The duration of time the schedule runs. e.g. 03:00
day	One or more days of the week the schedule runs. Possible values (separated by comma): **mon tue wed thu fri sat sun**

11.7 Appendix

11.7.1 Driver Details

See the '***ISY Node Server Developer's Manual***' for description of ISY Driver Controls. See section: **8 ISY Developer's Manual**

11.7.2 Units of Measure

The units of measure include the scientific units of measure as well as custom types. A unit of measure is a numeric type that fully defines a value. Values in square brackets are keywords that may be specified for the unit of measure instead of the numeric value when sending a request to ISY. The ISY will always return the numeric value for the unit of measure.

See the '***ISY Node Server Developer's Manual***' for description of ISY units of measure. See section: **8 ISY Developer's Manual**

11.7.3 Node Types

A Z-Wave device is represented by one or more ISY nodes. Each Z-Wave ISY node has a unique type that represents a subset of the features for the device. For example, a thermostat that also includes energy monitoring would have two nodes, one of node type 140-Thermostat, and the other 143-Energy Meter.

From 4_fam.xml <isyNodeTypes>

0 = Uninitialized	144 = Pulse Meter
101 = Unknown	145 = Water Meter
102 = Alarm	146 = Gas Meter
103 = AV Control Point	147 = Binary Switch
104 = Binary Sensor	148 = Binary Alarm
105 = Class A Motor Control	149 = Aux Alarm
106 = Class B Motor Control	150 = CO2 Alarm
107 = Class C Motor Control	151 = CO Alarm
108 = Controller	152 = Freeze Alarm
109 = Dimmer Switch	153 = Glass Break Alarm
110 = Display	154 = Heat Alarm
111 = Door Lock	155 = Motion Sensor
112 = Doorbell	156 = Smoke Alarm
113 = Entry Control	157 = Tamper Alarm
114 = Gateway	158 = Tilt Alarm
115 = Installer Tool	159 = Water Alarm
116 = Motor Multiposition	160 = Door/Window Alarm
117 = Climate Sensor	161 = Test Alarm
118 = Multilevel Sensor	162 = Low Battery Alarm
119 = Multilevel Switch	163 = CO End Of Life Alarm
120 = On/Off Power Strip	164 = Malfunction Alarm
121 = On/Off Power Switch	165 = Heartbeat
122 = On/Off Scene Switch	166 = Overheat Alarm
123 = Open/Close Valve	167 = Rapid Temp Rise Alarm
124 = PC Controller	168 = Underheat Alarm
125 = Remote	169 = Leak Detected Alarm
126 = Remote Control	170 = Level Drop Alarm
127 = AV Remote Control	171 = Replace Filter Alarm
128 = Simple Remote Control	172 = Intrusion Alarm
129 = Repeater	173 = Tamper Code Alarm
130 = Residential HRV	174 = Hardware Failure Alarm
131 = Satellite Receiver	175 = Software Failure Alarm
132 = Satellite Receiver	176 = Contact Police Alarm
133 = Scene Controller	177 = Contact Fire Alarm
134 = Scene Switch	178 = Contact Medical Alarm
135 = Security Panel	179 = Wakeup Alarm
136 = Set-Top Box	180 = Timer
137 = Siren	181 = Power Management
138 = Smoke Alarm	182 = Appliance
139 = Subsystem Controller	183 = Home Health
140 = Thermostat	184 = Barrier
141 = Toggle	185 = Notification Sensor
142 = Television	186 = Color Switch
143 = Energy Meter	

11.7.4 Model Types

The model of device is usually determined by comparing a series of manufacturer specific IDs. This list makes it easier to determine specific devices by assigning them a simple integer value.

```
From 4_fam.xml <isyDeviceModels>

 1 = Schlage BE369
 2 = Schlage BE469
 3 = Schlage FE599
 4 = Kwikset Door Lock
 5 = Yale Door Lock
 6 = FortrezZ Water Valve
 7 = First Alert Smoke Detector
 8 = First Alert CO and Smoke Detector
 9 = Aeon Multisensor
10 = RCS PMC40
11 = FortrezZ MIMOLite
```

12 ISY OpenADR (VEN) Developer's Manual

Web Services SDK and REST Interface, based on firmware 4.3.1

12.1 Introduction

ISY is an energy management system platform based on ISY framework. As such, all ISY interfaces, services, and events are applicable to ISY as well.

ISY adds support for energy management use cases as well as native support for Smart Grid standards and interfaces such as SEP (Smart Energy Profile) and OpenADR 2.0(a) and (b).

OpenADR is a set of standardized messages which convey load, price, and other desired events from the utility to the facility. In OpenADR vernacular, the server sending the events is called a VTN and the recipient of the events is called a VEN. ISY is a VEN.

ISY can natively process and orchestrate the OpenADR events/instructions. To help expedite OpenADR application development based on ISY, complex OpenADR messages/events are transformed to simple XML payloads that can be understood easily. The schema for the transformation is captured in *oadrobjs.xsd*.

12.2 Putting it Together

ISY is based on the same framework as ISY and therefore communications and event infrastructure follow the same paradigm. If you have not yet reviewed ISY's WSDK Developer's guide, please send an email to sales@universal-devices.com.

If you do not already have OpenADR (21010) Module installed on ISY, please send an email to sales@universal-devices.com with your UUID (Help | About) and your desire to have Broadband SEP module activated.

For more information on finding ISY on the network and communications thereto, please consult ISY WS-SDK Manual.

Once you are successfully communicating with ISY and have the Module installed, then:

12.2.1 Retrieve Configuration

Configuration is represented by oadr:AutoDRConfig element (defined in oadrobjs.xsd). To retrieve configuration, you need to send a GetSysConf SOAP request to retrieve **/CONF/ADR.CFG**, an example of which is as follows:

```
POST /services HTTP/1.1\r\n
Host: 192.168.0.112:80\r\n
Authorization: Basic YWRtaW46YWRtaW4=\r\n
Content-Length: 157\r\n
Content-Type: text/xml; charset="utf-8"\r\n
SOAPACTION:"urn:udi-com:service:X_Insteon_Lighting_Service:1#GetSysConf"
\r\n\r\n
<s:Envelope><s:Body><u:GetSysConf xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><name>/CONF/ADR.CFG</name></u:GetSysCo
nf ></s:Body></s:Envelope>
```

12.2.2 Save Configuration

To save OpenADR Configuration, an instance of oadr:AutoDRConfig element (defined in oadrobjs.xsd) must be saved into **/CONF/ADR.CFG** an example of which is as follows:

```
POST /file/upload/CONF/ADR.CFG?load=y HTTP/1.1\r\n
Host: 192.168.0.112:80\r\n
Authorization: Basic YWRtaW46YWRtaW4=\r\n
Content-Length: 400\r\n
Content-Type: text/xml; charset="utf-8"\r\n
\r\n\r\n
<AutoDRConfig>
….
</AutoDRConfig>
```

12.2.3 Retrieve Reporting Configuration (2.0b)

Configuration is represented by oadr:OADRReportsConfig element (defined in oadrobjs.xsd). To retrieve configuration, you need to send a GetSysConf SOAP request to retrieve **/CONF/ADRR.CFG**, an example of which is as follows:

```
POST /services HTTP/1.1\r\n
Host: 192.168.0.112:80\r\n
Authorization: Basic YWRtaW46YWRtaW4=\r\n
Content-Length: 157\r\n
Content-Type: text/xml; charset="utf-8"\r\n
SOAPACTION:"urn:udi-com:service:X_Insteon_Lighting_Service:1#GetSysConf"
\r\n\r\n
<s:Envelope><s:Body><u:GetSysConf xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><name>/CONF/ADRR.CFG</name></u:GetSysC
on f></s:Body></s:Envelope>
```

12.2.4 Save Reporting Configuration (2.0b)

To save OpenADR Reporting Configuration, an instance of oadr:OADRReportsConfig element (defined in oadrobjs.xsd) must be saved into **/CONF/ADRR.CFG** an example of which is as follows:

```
POST /file/upload/CONF/ADRR.CFG?load=y HTTP/1.1\r\n
Host: 192.168.0.112:80\r\n
Authorization: Basic YWRtaW46YWRtaW4=\r\n
Content-Length: 400\r\n
Content-Type: text/xml; charset="utf-8"\r\n
\r\n\r\n
<AutoDRConfig>
….
</AutoDRConfig>
```

12.2.5 Save Opt Schedules (2.0b)

To save OpenADR Opt Schedules, an instance of oadr:OADROptSchedules element (defined in oadrobjs.xsd) must be saved into **/CONF/ADROPT.CFG** an example of which is as follows:

POST /file/upload**/CONF/ADRR.CFG**?load=y HTTP/1.1\r\n
Host: 192.168.0.112:80\r\n
Authorization: Basic YWRtaW46YWRtaW4=\r\n
Content-Length: 400\r\n
Content-Type: text/xml; charset="utf-8"\r\n
\r\n\r\n
<OADROptSchedules>
….
</OADROptSchedules>

12.2.6 Get notified of OpenADR Events

ISY is based on publish subscribe framework and, therefore, the client can subscribe to ISY and then be notified of all events from then onwards and without the need to keep polling ISY.

To learn about subscribing to ISY, please consult section: **8.4 Communicating with ISY**

When subscribed, events with ***Control*=_10** and the following actions are those that relate to OpenADR:

action = "1" -> **OpenADR Error**
node = null
eventInfo = null
action = "2" -> **OpenADR Status Update**
node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
oadr:OpenADRState
</**eventInfo**>
action = "8" -> **OpenADR Registration Status**
node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
oadr:OADRRegistration
</**eventInfo**>
*Profile 2b only

action = “9” -> **OpenADR Report Status**
node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
oadr:OADRReport
</**eventInfo**>
*Profile 2b only
action = “10” -> **OpenADR Opt Status**
node = null
Namespace: **oadr (oadrobjs.xsd)**
<**eventInfo**>
oadr:oadrOpt
</**eventInfo**>
*Profile 2b only

12.2.7 Retrieving OpenADR 2.0 Full Payload

OpenADRState which is published as event to clients, shows only the overall status of OpenADR subsystem (or any changes thereto).

For Profile 2(a), it’s usually very important to know more details especially about Signals and Intervals. Since the payload for OpenADR 2.0 profile is much larger than the amount of information ISY wishes to publish to clients, it’s best for clients to retrieve the payload if and only if the information in the event is not sufficient.

To do so, an http GET on:
/rest/oadr

To help expedite OpenADR application development based on ISY, complex OpenADR messages/events are transformed to simple XML payloads that can be understood easily. As such, the result of /rest/oadr is an oadr:eiEvents element

the schema for which is captured in *oadrobjs.xsd.*

12.2.8 Opting In and Out of Events (2.0a/2.0b)

You can opt in and out of events.

To Opt-In, do a GET on:
/rest/oadr/<event-id>/optin

To Opt-Out, do a GET on:
/rest/oadr/<event-id>/optout

12.2.9 Clearing All Events

To clear all events, do an http GET on:
/rest/oadr/clear

12.2.10 Registration Service URLs (2.0b)

Register to VTN
/rest/oadr/party/register

Cancel Registration to VTN
/rest/oadr/party/unregister

Query Registration
/rest/oadr/party/queryreg

Re-Registration
/rest/oadr/party/reregister?type=val
Where type is *refresh* | *renew*
refresh = uses the existing Registration ID
renew = does not use the existing Registration ID

12.2.11 Report Service URLs (2.0b)

Register report Meta Data
/rest/oadr/report/register?type=val
Where type is *usage* | *status* | *history* | *all [default]*

Get status of all reports (oadrobjs:OADRReportsStatus)
/rest/oadr/report/status

12.2.12 Opt Service URLs (2.0b)

Retrieve all schedules (oadrobjs:OADROptSchedules)
/rest/oadr/opts

Cancel an Opt Schedule
/rest/oadr/opts/cancel/<opt-id>
Where opt-id is the unique id for the opt to be canceled (oadrobjs:oadrOpt)

Cancel all Opt schedules
/rest/oadr/opts/cancelAll

Clear all inactive Opt schedules
/rest/oadr/opts/clear

12.2.13 Push URLs

For 2.0a Push mode, ISY endpoint is:
/OpenADR2/Simple/EiEvent

For 2.0b Push mode, ISY endpoint is:
/OpenADR2/Simple/2.0b/[Service]
Where service is:
EiEvent – Event services
EiReport – Reporting services
EiOpt – Scheduling services

13 ISY Portal Integration Developer's Manual

Based on firmware 4.3.1

13.1 Introduction

ISY is an automation and energy management platform with well-defined Web Services and REST interfaces that provide the developers with a complete set of management and event handling functionality. For information on Web Services/REST/Events please consult ISY WSDK Development Guide, section: 17 **Soap / Web Services (WSDK)**

Since ISY has its own web server thus network traffic must be forwarded to the ports (HTTP and HTTPS) that the web server listens to. For local access, this process is as trivial as pointing the client to ISY's IP address/web server port combination. On the other hand, for remote access a port forwarding rule must be created in the router to forward Internet traffic to ISY's IP address/web server port which might become a daunting task for anyone not intimately knowledgeable about routers and networking.

ISY's Portal Integration Module enables integration of Cloud based/Portal services to act as proxy for communications with ISY. With Portal Integration Module, ISY makes a persistent outbound connection to a predefined server the socket for which can be used to proxy client requests back to ISY and using the same Web Services and REST interfaces. In this manner, then, ISY will be accessible remotely without the need for port forwarding rules in the router.

Please note that Portal Integration Module is project/portal specific and has to be approved by UDI before going to production. If you are interested in developing a Portal solution for ISY, please contact sales@universal-devices.com and we'll activate a development Portal Integration Module for your authorized ISYs.

13.2 Definitions and Portal URLs

Portal Integration Module is designed to be as simple, efficient, and transparent as possible. The following URLs must be implemented at the portal and accessible by ISY:

13.2.1 Dispatcher URL

Dispatcher URL is the URL for a server which may be able to provide some level of load balancing and is the first URL/server to which ISY tries to connect to. The URL must support HTTPS GET method and must return (perhaps based on some load balancing algorithm) a list of one or more Proxy Servers (see section: 13.2.2 Proxy URL) separated by a newline (\n) such as:

https://www1.universal-devices.com/isy
https://www2.universal-devices.com/isy
….
and so no and so forth.

Dispatcher URLs are different for each specific Portal Integration Module and, upon approval, are built into ISY's firmware.

This said, for development purposes, Dispatcher URL can be modified through the shell by the following commands

SPU – Sets the Dispatcher URL to a user defined value

DPU – Reverts the Dispatcher URL to the Default value as defined by that specific Portal Integration Module

13.2.2 Proxy URL

Proxy URL is the actual URL to which ISY makes a connection to, keeps the socket open indefinitely (or till there's a connection error), and is defined as one of the entries in the output of doing a GET on the Dispatcher URL (see section: **13.2.1 Dispatcher URL**)

It's up to the server to make and maintain a mapping between the socket connection and a specific ISY. In most cases, it's best to use ISY's MAC/UUID address as the unique identifier for the mapping since the MAC address uniquely identifies an ISY and never changes. Please see section **/rest/whoami** for information on how to retrieve ISY's MAC address.

***Important Note*:** If the portal is going to proxy Admin Console requests, please do make sure that there's an **/isy** *anywhere* in the proxy URL.

13.2.3 Service/Account Request URL

This URL is served by the portal, it's relative to the Dispatcher URL, and returns a list of Account/Service Authorization requests from the portal to ISY:

1. The URL must be of the form:

   <Dispatcher URL>/api/auth/requests?uuid=<ISY's UUID>; e.g.
   http://my.portal.com/dispatcher/api/auth/requets?uuid=00:11:22:33:44:55

2. Will return the payload as defined in **portal.xsd::authRequests ;** e.g.:

```
<authRequests>
    <account>
        <id>Unique ID for the account</id>
        <name>Name of the account</name>
        <description>Description for this account</description>
        <address/>
        <website>The actual website for the portal</website>
        <contactname>Contact Name</contactname>
        <contactphone>Contact Phone </contactphone>
        <contactemail>Contact email address</contactemail>
        <accountinfo>
            The URL for more information on this Account
        </accountinfo>
    </account>
</authRequests>
```

3. Must return the following HTTP status codes:

   200 – Success
   400 – Missing or invalid UUID
   500 – General Server errors

13.2.4 2.4 Active Service/Account URL

This URL is served by the portal, it's relative to the Dispatcher URL, and returns a list of Active Accounts/Services in the portal for the given ISY:

1. The URL must be of the form:
   <Dispatcher URL>/api/auth/active?uuid=<ISY's UUID>; e.g.
   http://my.portal.com/dispatcher/api/auth/active?uuid=00:11:22:33:44:55

2. Will return the payload as defined in **portal.xsd::authActive ;** e.g.:

```
<authActive>
    <account>
        <id>Unique ID for the account</id>
        <name>Name of the account</name>
        <description>Description for this account</description>
        <address/>
        <website>The actual website for the portal</website>
        <contactname>Contact Name</contactname>
        <contactphone>Contact Phone </contactphone>
        <contactemail>Contact email address</contactemail>
        <accountinfo>
            The URL for more information on this Account
        </accountinfo>
    </account>
</authActive>
```

3. Must return the following HTTP status codes:

   200 – Success
   400 – Missing or invalid UUID
   500 – General Server errors

13.2.5 Account/Service Approval URL

This URL is served by the portal, it's relative to the Dispatcher URL, and allows an ISY user to Approve a portal account/service request (see section **13.2.2 Proxy URL**)

1. The URL must be of the form:

<Dispatcher URL>/api/auth/approve?<Account ID>/<ISY UUID>/<Authorization Key>
http://my.portal.com/dispatcher/api/auth/approve/$superadmin/00:11:22:33:44:55/AABBCCDDEEFF00112233445566778899AABBCCDDEEFF00112233445566

2. The Key is a 32 byte hex number as String. This key has to be returned AS-IS back to ISY through the persistent socket connection from ISY to the portal

3. Must return the following HTTP status codes:

200 – Successful key exchange with ISY through socket
400 – Missing account, uuid or key; ISY does not exist in portal database
480 – ISY is not online (no persistent connection from ISY to the portal
500 – General serve error
XXX: If call to ISY through sockets returns a code other than 200, the same http code will be returned by this API. (Example: if POST /rest/portal/approve returns 403 Forbidden, then 403 will be returned here).

13.2.6 Account/Service Revocation URL

This URL is served by the portal, it's relative to the Dispatcher URL, and allows an ISY user to Revoke a portal account/service request (see section **13.2.2 Proxy URL**)

1. The URL must be of the form:

<Dispatcher URL>/api/auth/revoke?<Account ID>/<ISY UUID>/<Authorization Key>
http://my.portal.com/dispatcher/api/auth/revoke/$superadmin/00:11:22:33:44:55/AABBCCDDEEFF00112233445566778899AABBCCDDEEFF00112233445566

2. The Key is a 32 byte hex number as String. This key has to be returned AS-IS back to ISY through the persistent socket connection from ISY to the portal

3. Must return the following HTTP status codes:
   200 – Successful key exchange with ISY through socket
   400 – Missing account, uuid or key; ISY does not exist in portal database
   480 – ISY is not online (no persistent connection from ISY to the portal
   500 – General serve error
   XXX: If call to ISY through sockets returns a code other than 200, the same http code will be returned by this API. (Example: if POST /rest/portal/approve returns 403 Forbidden, then 403 will be returned here).

13.2.7 Subscription URL

*Portal must be registered to ISY (see section **13.3 Process Flow**)

Subscription URL is the URL to which ISY publishes all its events. Subscription URL should be provided in the ***<reportURL>*** element of ***Subscribe*** Web Service call by the Proxy Server. Subscription URL can be an entirely different URL than the Proxy URL and it must support HTTPS.

There can only be one subscription to the portal the SID (Subscription ID) for which is always ***0***.

Please review ISY's WSDK)for more information, section: 17 **Soap / Web Services (WSDK)**, specifically section for **/rest/subscriptions** interface information.

13.2.8 Authentication and Authorization

*Portal must be registered to ISY (see section **13.3 Process Flow**)

If the portal has been registered (see section 3), for ISY, the Portal is a preauthorized and authenticated client and thus ISY does not check credentials for requests coming from the portal. As such, it's of utmost importance to make sure the Proxy Server has a robust authentication, authorization, and security infrastructure.

Since the proxy is acting on behalf of a client, as such the proxy **must** trap the ***Authenticate*** Web Service (from clients) and process its own authentication and authorization.

Furthermore, the proxy must be aware of URLs which do not require any authentication:

1. /desc
2. /WEB/*

Note: If you are going to let the Admin Console communicate with ISY through the proxy and ***if and only if*** you wish the Admin Console to use a portal-defined ***URLbase*** to communicate with the proxy, then you must trap the ***/desc*** URL, and replace the ***<URLBase>*** element with the URL bases of your own choosing. Example:

Replace the following:

<***URLBase***>***http://192.168.0.129***</***URLBase***>

With:

<URLBase>https://your-portal-url/[any-path]+</URLBase>

13.2.9 Subscription

*Portal must be registered to ISY (see section **13.3 Process Flow**)

Since the clients are now going through the proxy server, as such the proxy server must now provide subscription services to the clients. Therefore, the proxy server must trap subscription requests and stop them from getting to ISY. The following Web Services must be trapped:

1. ***Subscribe***
2. ***IsSubscribed***

For definition of these services, please consult the WSDK documentation, section: 17 **Soap / Web Services (WSDK)**

13.3 Process Flow

With firmware releases 4.3.x and above, a security handshake has been introduced to ensure that

1. ISY is talking to the correct portal
2. The user has full control over authorization between the portal and ISY
3. Security negotiations are handled using random keys that live only for 5 minutes (or less)
4. If ISY does not consider the portal registered, all requests to ISY through the socket will return 401
5. Additional <isRegistered> element in /rest/whoami provides information about account/service registration status

The following process flow assumes that:

1. Portal Module is installed in ISY
2. The Portal has an active account for ISY/User combination

13.3.1 Putting It Together

The process flow is as follows:

1. Upon reboot

   a. ISY tries to retrieve a list of Proxy Servers from the Dispatcher URL
   b. For each Proxy URL ISY tries to make a connection thereto
   c. If successful, a session is established; **go to 2**
   d. If unsuccessful, try the next URL
   e. If all URLs fail, wait for 1 minute and **go back to a**

2. The Proxy Server accepts the connection from ISY and starts communicating with ISY using ***this socket*** here on out:
   a. Proxy Server retrieves ISY identification information by submitting an HTTP GET request to **/rest/whoami**
   b. If <isRegistered>true</isRegistered> go to step (c). Else
      i. ISY has not yet registered the portal. The portal must put this specific ISY in a list that is returned via Service/Account Request URL
      ii. The user will retrieve the list of requests through Admin Console or Dashboard
      iii. The user will approve the request in which case ISY will call Service/Account Approval URL.
      iv. The Portal validates the information and immediately calls ISY, through the portal socket, via /rest/portal/approve
   c. Proxy Server uses the MAC address in addition or combination with some other identifying information to create and maintain a mapping between the socket and ISY
   d. Proxy Server then subscribes to ISY while providing it a unique URL for the ***<reportURL>*** element of the ***Subscribe*** Web Service call.

   Important Note: You cannot use REUSE_SOCKET value for <reportURL> since it would cause ISY to use the same socket for both subscriptions as well as command/control

3. Proxy Server uses some algorithms based on ISY's Heartbeat Event in combination with other control information (such as TCP KeepAlive packets sent by ISY or periodic ***/rest/whoami*** requests) to make a determination as to whether or not ISY is still communicating with it:
   a. In case it has determined that ISY is no longer publishing events, Proxy Server uses ***IsSubscribed*** Web Service call using ***SID of 0***, to see whether or not it's still subscribed to ISY
   b. If it's not subscribed, ***repeats 2a – 2c***

c. If it cannot communicate with ISY at all then it closes the socket and removes all the mappings between the socket an ISY. Proxy Server shall then wait indefinitely for ISY to try and contact it again and, once contacted, ***repeats 2a – 2c***

4. Just as any other client, ISY publishes all events to the Subscription URL
   a. In case of connection/network failure, ISY will retry **3** times. In case all 3 attempts fail, ISY expires the subscription
   b. It's up to the Server to notice lack of events coming from ISY and re-subscribe (***3a)***
   c. The relationship between Subscription socket and Proxy socket is as follows:
      i. If ISY can no longer communicate with the Proxy, ISY expires the associated subscription (if any)
      ii. If ISY can no longer communicate with the Subscription Server, ISY only expires the subscription but the Proxy socket remains active (**4a**)

Once a socket session is established between ISY and the Proxy Server, all Web Services and REST interface calls can be issued ***as-is*** by the Proxy Server and through the Proxy socket.

The only limitation is that the commands must be issued synchronously: only one Web Service or REST interface call can be issued at any given time. i.e the Proxy Server has to wait for one command to finish before issuing the next. For best performance, it's advised that the Proxy Server uses a queuing system to queue commands and process them one at a time.

13.4 REST Interface

As mentioned before, the Proxy Server can use all ISY Web Services and REST interface calls, as-is (and without the Authorization headers) through the Proxy Server socket. In addition to ISY services, the Portal Integration Module provides the following additional REST interface(s):

13.4.1 /rest/whoami

* This URL is accessible through the portal socket as well as local network

This interface returns uniquely identifying attributes of the given ISY to which the Proxy Server is connected:

<iam>
<partner>*A string representing the partner ID*</partner>**
<type>*A string representing the type of ISY*</type>**
<product>*ISY's product ID which represents a model number*</product>**
<id>*Hexadecimal MAC address*</id>**
<SID>

Subscription ID if any … this element will not be there if ISY has no active subscription to the Portal. Otherwise, the value is always:
uuid:0
</SID>
<electricity>
<providerId>
A String representing the Utility provider ID if any
</providerId>
<enrollmentGroup>
An Integer representing the Utility enrollment group if any
</enrollmentGroup>
</electricity>
<isRegistered>
True|false
*True if and only if there's at least one active/registered account (see section **13.3 Process Flow**)*
** To accommodate older portal solutions, if ISY is booted up with a module already installed, then ISY will consider that portal registered.*
</isRegisterd>
</iam>

Example:

```
<iam>
    <partner>Portal-X-Partner</partner>
    <type>ISYv50</type>
    <product>1110</product>
    <id>0021b9030405</id>
    <SID>uuid:0</SID>
    <electricity>
        <providerId>SDGE</providerId>
        <enrollmentGroup>0</enrollmentGroup>
    </electricity>
    <isRegistered>true</isRegistered>
</iam>
```

13.4.2 /rest/portal/status

* This URL is accessible through the portal socket as well as local network

This interface returns the status of the portal socket and whether or not it has active registrations (defined in portal.xsd::PortalStatus). The following example shows 3 active accounts for ISY:

```
<PortalStatus isConnected="true" isRegistered="true">
    <accounts>
        <account status="registered">
            <id>$superadmin</id>
        </account>
        <account status="registered">
            <id>552ad3a2c42fba250962898b</id>
        </account>
        <account status="registered">
            <id>552ad396c42fba250962898a</id>
        </account>
    </accounts>
</PortalStatus>
```

13.4.3 /rest/portal-server/requests/<ISY's UUID>

* This URL is only accessible through ISY client

This interface calls Service/Account Request URL (see section **13.2.3 Service/Account Request URL**) and returns the contents (portal.xsd::authActive) or error. Example:
http://local-isy.ip.address/rest/portal/server/requests/00:21:b9:01:fb:ce

13.4.4 /rest/portal-server/active/<ISY's UUID>

* This URL is only accessible through ISY client

This interface calls Active Service/Account URL (see section **13.2.6 Account/Service Revocation URL**) and returns the contents (portal.xsd::authActive) or error. Example:
http://local-isy.ip.address/rest/portal/server/active/00:21:b9:01:fb:ce

13.4.5 /rest/portal-server/approve/<Account ID>

* This URL is only accessible through ISY client

This interface calls Account/Service Approval URL (see section **13.2.7 Subscription URL**) and returns success/failure status. ISY fills in UUID and random key.
Example
http://local-isy.ip.address/rest/portal/server/approve/0022335EABC8924

13.4.6 /rest/portal-server/revoke/<Account ID>

* This URL is only accessible through ISY client

This interface calls Account/Service Revocation URL (see section **13.2.6 Account/Service Revocation URL**) and returns success/failure status. ISY fills in the UUID and random key.

Example
http://local-isy.ip.address/rest/portal/server/approve/0022335EABC8924

13.4.7 /rest/portal/approve/<Account ID>/<ISY's UUID>/Key

* This URL is only accessible through the **portal socket**.

When ISY calls the portal through Account/Service Approval URL (see section **13.2.5 Account/Service Approval URL**), the portal must validate the Account ID and UUID and then immediately call this URL through the **portal socket** providing Account ID, UUID, and the Key that was passed in the initial call from ISY.

ISY will validate the information and returns:
- 200 – Successful
- 400 – Bad request

If successful, ISY considers the account registered, sets <isRegistered> element to true in /rest/whoami, and allows traffic to go through the portal socket.

13.4.8 /rest/portal-server/revoke/<Account ID>/ISY's UUID/Key

* This URL is only accessible through the **portal socket**.

When ISY calls the portal through Account/Service Revocation URL (see section **13.2.6 Account/Service Revocation URL**), the portal must validate the Account ID and UUID and then immediately call this URL through the **portal socket** providing the Account ID, UUID, and the key that was passed in the initial call from ISY.

ISY will validate the information and returns:

200 – Successful

400 – Bad request

If successful, ISY removes registration for the associated account. If there are no other accounts that are registered, then <isRegistered> will be false in /rest/whoami and ISY will block all requests (except portal security requests) through the portal socket.

13.4.9 Events

13.4.9.1Portal Events (control = "_23")

In case a portal module is installed, these events are published based on:

1. Portal socket connection status
2. Portal account registration status

action = "1" -> Status

node = null

<**eventInfo**>portal.xsd::PortalStatus</**eventInfo**>

14 ISY Energy Management System Developer's Manual

Web Services SDK and RESTG Interface, based on firmware 4.3.1

14.1 Introduction

ISY is an energy management system platform based on ISY framework. As such, all ISY interfaces, services, and events are applicable to ISY as well.

ISY adds support for energy management use cases as well as native support for Smart Grid standards and interfaces such as SEP (Smart Energy Profile).

SEP Profile is a set of instructions which can be sent by the utilities to either instruct the HAN (Home Area Network) to orchestrate certain scenarios to achieve load shedding/demand reduction or provides useful information to the consumer so that they could make informed decisions with respect to their energy utilization.

ISY can natively process and orchestrate the HAN based on SEP events/instructions. All SEP events have a Start Time and duration. Duration could be forever – or till a Stop event has been sent.

Although detailed discussion on SEP is outside the scope of this document, but understanding the following four instructions, as defined by SEP, are essential for developing applications for ISY:

14.2 DRLC (Demand Response Load Control)

DRLC is an instruction sent by the utility, or a DR service provider, to cause the HAN to change its operational characteristics based on the following set of attributes:

Criticality – the level of importance for this event

Duty Cycle – the percent of time in which a device must be cycled on/off. Usually applied to load controllers such as pool pumps and water heaters

Average Load Adjustment – the percentage by which adjustable loads should shed their loads. Usually applied to dimmers and variable load devices

Heating Setpoint – precise heating setpoint to which thermostats must be adjusted to. This is mutually exclusive of Heating Setpoint Offset

Cooling Setpoint – precise cooling setpoint to which thermostats must be adjusted to. This is mutually exclusive of Cooling Setpoint Offset

Heating Setpoint Offset – The offset by which the Heating Setpoint for thermostats must be decreased

Cooling Setpoint Offset – The offset by which the Cooling Setpoint for thermostats must be increased

Device Class – This attribute defines the classes of devices (such as pool pumps) that this event must be applied to.

Note that all DRLC events must be explicitly opted into by the customer.

14.3 Message

Message is an informational signal sent by the utility to notify the customer of some event. Messages come in two types:

i. Those requiring customer to confirm receipt
ii. Those that do not require the customer to confirm receipt

14.4 Price

Price is an informational signal sent by the utility to notify the customer of pricing information per tier. CPP (Critical Peak Price) is sent on Tier 5.

14.5 Meter

Meter is the information coming directly from an AMI/Smart Grid meter using SEP protocol. Meter provides such information as current consumption per tier or even current production (i.e. Solar Panels).

14.6 Common Messages/Signals

Theoretically, all SEP enabled Meters would be able to provide all the above messages/signals. ISY, when equipped with Zigbee SEP module, can natively communicate with the meter and process the information and take actions based on user defined use cases.

This said, however, since not all meters are yet equipped with Zigbee SEP modules, UDI has come up with Web Services version of the first 3 signals (DRLC, Message, and Price) and thus

allowing for the same signals to be sent using Web Services. In UDI Vernacular, we call these Web Services Broadband SEP.

Broadband SEP is an optional module which might need to be activated. If you do not have this module enabled, please contact sales@universal-devices.com with your UUID (Help | About) and request for this module to be activated.

The beauty of Broadband SEP is that you can use 3rd party tools, such as SOAP UI to consume SEP Services provided by ISY and generate or modify events at will.
As mentioned before, ISY is a full-fledged energy management system. In order to provide the greatest level of compatibility and flexibility, ISY supports SEP signal from either Web Services or directly from Zigbee meters.

ISY has the capability of receiving, scheduling, and running up to 10 events for each type of DRLC, Price, and Message events. Furthermore, all events are persisted in ISY's file system so that, in case of power failure, they can be restored and restarted/rescheduled.

In addition to full ISY Framework's programmability options available to the users, ISY provides the developer with simple configuration files which can be used in a majority of cases to make customer setup easier and more user friendly. For more information on configuration options for SEP events, please see section 3. ***DEIDRICH *** a few pages down *** Configuration ***

14.7 Getting Started

ISY is based on the same framework as ISY and therefore communications and event infrastructure follow the same paradigm. If you have not yet reviewed ISY's WSDK Developer's guide, please send an email to sales@universal-devices.com.

If you do not already have Broadband SEP (BSEP) module installed on ISY, please send an email to sales@universal-devices.com with your UUID (Help | About) and your desire to have Broadband SEP module activated.

Once you are successfully communicating with ISY and have BSEP installed, then:
1. Go to http://isy:port/desc (or http://your.isy.ip.address:port/desc)

   You will be presented with the description of services provided by ISY. In the <serviceList> element, look for **UDISEPWebServices** as the <serviceType>. What you are looking for is the URL for Web services binding. This URL is defined <SCPDURL> (see below):

```
<serviceList>
    <service>
        Default ISY Framework service descriptions
    </service>
    <service>
        <serviceType>UDISEPWebServices</serviceType>
        <serviceId>
            uuid:00:03:f4:03:65:96-UDISEPWebServices
        </serviceId>
        <SCPDURL>/sepServices.wsdl</SCPDURL>
        <controlURL>/sepServices</controlURL>
    </service>
</serviceList>
```

2. Now, all you need to do is point your SOAP client to: http://your.isy.ip.address:port/[value for SCPDURL] and import BSEP web services

3. Get ISY Configuration (GetISYConfig Web Service or /rest/config) and ensure that Broadband SEP module is installed (id = 21080). For more information on Configuration Resources and Modules, please consult ISY WSDK Developer's Guide, section: 17 **Soap / Web Services (WSDK)**

4. All BSEP web services are defined in the WSDL returned by step 2. These include all SEP event objects and configuration definitions

5. If you are using SOAP UI, you can immediately communicate and issue SEP events to ISY

Please note that all requests require the Authorization header. Furthermore, if you wish to receive SEP events, please do make sure that you have subscribed to ISY.

14.8 SEP Event Lifecycle

All SEP Events go through a specific life cycle the starting point of which is Expired or Done. These states are defined by the **eventState** attribute of all events and enumerated by **SEPEventState** both of which are defined in the WSDL.

All SEP Events have a Start Time (defined by the **startTime** element) and Duration (defined by **duration** element in minutes). Two special cases:

a. If Start Time is 0 or not provided, ISY considers the event's Start Time to be **now**

b. If Duration is 0, not provided, or 65535 (max UINT2), ISY considers the duration to be forever or till a Stop event for the same type has been issued

When events are received by ISY, the following checks are made to make sure the event is valid:

a. Whether or not it has the correct Enrollment Group (defined by **enrollmentGroup** element on all events) matching that already configured via the **EnrollmentGroup** element in **UserElectricityOptions** configuration (see section 3). If they do not match, the event is discarded as invalid and logged
b. Whether or not the Event ID (defined by the **eventId** element) is not already being used by other events in the system. If another event with the same Event ID is found, the event is discarded as invalid and logged
c. Whether or not the event is in the past. If so, event is discarded as invalid and logged
d. Whether or not there are any conflicts with existing events in the system
    i. With the exception of **Price** event, there can only be one event for any specific duration in time. If there are any conflicts, the event is considered to have a schedule conflict, discarded as invalid and logged
    ii. In the case of **Price** event, multiple events could be running in the same time interval as long as they have different Price Tiers (defined by **tier** element in **sepobjs.SEPPriceObject**

If the event is considered valid, then
a. It will go to the **Running** state if it should be started
b. It will go to the **Scheduled** state if it should be scheduled

When an event goes into **Running** state, ISY checks the User options for that event to see if any actions should be taken and, if so, it executes on those actions.

Events are periodically checked to see whether or not they should be expired in which case they will go to the **Done** state.

From a BSEP perspective, events are started using the **SEPStart[Event]** service and can be stopped at any time using the **SEPStop[Event]** service where **Event** could be any of the following:

a. DREvent
b. Price
c. Message

Every SEP causes ISY to publish the state of that event to all subscribers. Please note that the published events follow the same schema as those in the WSDL as well as **sepobjs.xsd** for requests and namely:

a. **sepobjs.xsd**.**SEPDRObject**
b. **sepobjs.xsd**.**SEPMessageObject**
c. **sepobjs.xsd**.**SEPPriceObject**

For more information on the type of events published by ISY, please see section 4. ***DEIDRICH *** down a few pages ISY SEP Events (control = "_12")***

All events are stored in the file system and therefore are reevaluated at startup and go through their normal life cycle.

14.9 Lifecycle of DR Events

Unlike straight forward nature of Price, Message, and Meter events, DR events a little different since they provide information as to how the HAN should respond. For instance, an event could target a specific Device Class such as Pool Pumps or a list of device classes. Or, Duty Cycle requires ISY to already know the total period during which Duty Cycle percent of the total time the device should be off and the rest on. As such, certain configuration options should be made to the operational environment of ISY.

Before an event can be acted upon, ISY goes through all the configured devices/nodes in the system and performs the following evaluation to filter out those nodes/devices that do not meet the criteria for the DR event:

Whether or not the node's Device Class (as defined by deviceClass in Node: see ISY WSDK Developer's Guide, 17 **Soap / Web Services (WSDK)**) is included in the event's Device Class. See next section for more information on Device Class.

14.9.1 Device Class

Device class defines the class of a device and is represented by the **deviceClass** element in **sepobjs.xsd**.**SEPDRObject**. Device classes are defined in **sepobjs.xsd**.**SEPDeviceClass** type as follows:

1 = HVAC / Thermostats
2 = Strip Heater
4 = Water Heater
8 = Pool Pump
16 = Smart Appliance
32 = Irrigation Pump
64 = Managed Load
128 = Simple
256 = Exterior Lighting
512 = Interior Lighting
1024 = Electric Vehicle
2048 = Generation System
4096 = Washer
8192 = Dryer
16384 = Oven
32768 = Refrigerator
65535 = ALL (0)

For multiple device classes, the values for each class are OR'ed with one another (e.g. 64 | 128). To set the Device Class for a node, please use ISY Web Service: **SetNodePowerInfo(deviceClass, wattage, dcPeriod)**.

14.9.2 Duty Cycle

Duty Cycle is the percentage of total period of time during which the device must remain off. For instance a Duty Cycle of 50% for a total period of 1 hour means that the device is turned Off for 30 minute and then turned on for 30 minutes.

Duty Cycle is defined by the **dutyCycle** element in **sepobjs.xsd**.**SEPDRObject** and the value is in percentage (0-100).

ISY uses the following rules to figure out whether or not a device can be put on Duty Cycle:
The device in question has a non-zero dcPeriod (defined by the **dcPeriod** attribute in the Node object: (see section: **8 ISY Developer's Manual**).

14.9.3 Opting In/Out

DR events must explicitly be acted upon by the user. In SEP vernacular, the actions taken by the user are called Opting in or Opting out.

In case, Auto Opt-in has been set, ISY automatically opts the user in as soon as the events arrive.

SEPDROpt service enables you to opt in/out of a specific DR event. In case of opting out, the event goes into the Done state and will not be reevaluated again. In case of opting in:

a. If the event is in the **Scheduled** state, it remains in the same state but an opt in event is sent out to ISY subscribers and the utility through the meter

b. If the event is in the **Running** state, it starts processing the event by adjusting HAN devices to match the attributes as defined by the event

14.9.4 Invalidating a DR Event

ISY automatically opts the user out of a DR event in case any changes to HAN devices invalidate the DR event's attributes.

For instance, if the event calls for a Duty Cycle of 50% applied to pool pumps, and if the pool pump was **on** prior to the event, and if the pool pump should now be in the **off** cycle, and if someone turns **on** the pool pump, then ISY invalidates the DR event and automatically opts the user out.

14.9.5 Revert to the State Prior to DR Event

Using the configuration options, you can configure ISY to go back to its state before the start of the DR Event. Please note that in the following two conditions, ISY does not honor the Revert request:

1. If the customer chooses to opt out during a running DR event
2. If the DR event is invalidated (see previous section)

14.10 Configuration

User configurations are simple means of letting ISY know what actions should be taken based on received SEP events.

Configurations are uploaded to ISY by posting to a specific URL in the following format:
/file/upload/[config_file_URI]?load=y

14.11 Base Electricity Configuration

Post sepobjs.xsd::**UserElectricityOptions** object to:
/file/upload/CONF/ELEC.CFG?load=y

14.12 Message Configuration

Post sepobjs.xsd::**SEPMessageUserOptions** object to:
/file/upload/CONF/EMMSO.CFG?load=y

14.13 Price Configurations

Post sepobjs.xsd::**SEPPriceUserOptions** object to:
/file/upload/CONF/EMPO.CFG?load=y

14.14 DRLC Configurations

Post sepobjs.xsd::**SEPDRUserOptions** object to:
/file/upload/CONF/EMDO.CFG?load=y

14.15 Meter Configurations

Post **SEPMeterUserOptions** object to:
/file/upload/CONF/EMMO.CFG?load=y

14.15.1 ISY SEP Events (control = "_12")

In addition to all the events published by ISY framework, ISY publishes SEP events using control = _17.

14.15.2 Network Status Changed (action = "1")

Status of Zigbee SEP Network.
node = null
eventInfo = **zigbee.xsd.ZigbeeNetwork**

14.15.3 Time Status Changed (action = "2")

Time from the Zigbee SEP meter changed.
node = null
eventInfo = **zigbee.ZigbeeSEPTime**

14.15.4 New Message (action = “3”)

A new Message event was received and validated.
node = null
eventInfo = **sepobjs.xsd**.**SEPMessageObject**

14.15.5 Scheduled Message (action = “31”)

A new Message event was received, validated, and scheduled for future execution.
node = null
eventInfo = **sepobjs.xsd**.**SEPMessageObject**

14.15.6 Message Stopped (action = “4”)

Message event was stopped and is no longer active
node = null
eventInfo = **sepobjs.xsd**.**SEPMessageObject**

14.15.7 New Price (action = “5”)

A new Price event was received and validated
node = null
eventInfo = **sepobjs.xsd**.**SEPPriceObject**

14.15.8 Scheduled Price (action = “51”)

A new Price event was received, validated, and scheduled for future execution
node = null
eventInfo = **sepobjs.xsd**.**SEPPriceObject**

14.15.9 Price Stopped (action = “6”)

Price event was stopped and is no longer active
node = null
eventInfo = **sepobjs.xsd**.**SEPPriceObject**

14.15.10 New DR Event (action = “7”)

A new DR event was received and validated
node = null
eventInfo = **sepobjs.xsd**.**SEPDRObject**

14.15.11 Scheduled DR Event (action = “71”)

A new DR event was received, validated, and scheduled for future execution
node = null
eventInfo = **sepobjs.xsd**.**SEPDRObject**

14.15.12 DR Event Stopped (action = “8”)

DR event was stopped and is no longer active
node = null
eventInfo = **sepobjs.xsd**.**SEPDRObject**

14.15.13 Metering Event (action = “9”)

Meter attribute value changed
node = null
eventInfo = **sepobjs.SEPMeteringReport**
Root element = none

14.15.14 Metering Format Event (action = “10”)

Meter format attribute values changed
node = null
eventInfo = **sepobjs.SEPMeterFormat**
Root element = **MeterFormat**

14.15.15 Fast Poll Mode (action = “110”)

ISY is in Fast Poll mode.

14.15.16 Normal Poll Mode (action = “111”)

ISY is in Normal Poll mode.

14.16 ISY Billing Events (control = "_22")

14.16.1 Cost/Usage Changed Event (action = "1")

Price, cost, and usage information changed.
node = null
eventInfo = **billobjs.xsd::CostUsageChangedEventInfo**
Root element = eventInfo

14.17 REST Interface

REST is an easy to use URL based command set which allows the developer to communicate and control ISY's Energy Management services.

Except for uploading configuration files, all REST commands use HTTP GET method.

14.17.1 Message

/rest/emeter/message
Returns a list of all messages (10)
Object: **sepobjs.SEPMessageObject**

/rest/emeter/message/reset
Clears all the cached and running events

/rest/emeter/message/query
Queries the meter for the last message

/rest/emeter/message/<event_id>
Returns the message with the given event_id
Object: **sepobjs.SEPMessageObject**

/rest/emeter/message/<event_id>/confirm
Sends a confirmation to for the message with the given event_id

/rest/emeter/message/log
Returns the log for messages (see section 6.0 for format of the log)

14.17.2 Price

/rest/emeter/price
Returns a list of all prices (10)
Object: **sepobjs.SEPPriceObject**

/rest/emeter/price/reset
Clears all the cached and running events

/rest/emeter/price/query
Queries the meter for the current price if any

/rest/emeter/price/query?offset=num1**&*maxListing*=**num2
Queries the meter for the scheduled prices if any:
offset – a number between -255 to 255 representing the number of minutes from the current time
maxListing – a number from 0 to 9 (0 returns means all)

/rest/emeter/price/<event_id>
Returns the price with the given event_id
Object: **sepobjs.SEPPriceObject**

/rest/emeter/price/log
Returns the log for prices (see section 6.0 for format of the log)

14.17.3 DRLC

/rest/emeter/drlc
Returns a list of all DRLC events (10)
Object: **sepobjs.sepo.SEPDRObject**

/rest/emeter/drlc/reset
Clears all the cached and running events

/rest/emeter/drlc/query?*offset*=num1**&*maxListing*=**num2
Queries the meter for the scheduled DRLC events if any:
offset – A number between -65535 to 65535 representing the number of minutes from the current time
maxListing – a number from 0 to 9 (0 returns means all)

/rest/emeter/drlc/<event_id>
Returns the DRLC event with the given event_id
Object: **sepobjs.SEPDRObject**

/rest/emeter/drlc/<event_id>/opt_in
Opts in to the DRLC event with the given event_id

/rest/emeter/drlc/<event_id>/opt_out
Opts out of the DRLC event with the given event_id

/rest/emeter/drlc/log
Returns the log for DRLC events (see section 6.0 for format of the log)

14.17.4 Zigbee SEP (Metering)

/rest/emeter
Summary of Zigbee network, module, DRLC, Price, Message, and default Metering attributes:
Object: **sepobjs.SEPSummary**
Root element: **AMIMeterSummary**

/rest/emeter/nework
Returns the current status of Zigbee Network.
Object: **zigbee.xsd.ZigbeeNetwork**
Root element: **ZBNetwork**

/rest/emeter/module
Returns the definition of currently installed Zigbee Module.
Object: **zigbee.xsd.ZigbeeModule**
Root element: **Module**

/rest/emeter/module/reset
Factory resets (to default) the currently installed Zigbee Module

/rest/emeter/time
Returns the module's current time and whether or not the module has been synchronized with the ESI/Meter.
Object: **sepobjs.xsd.ZigbeeSEPTime**
Root element: **Time**

/rest/emeter/time/query
Queries the ESI/Meter for current time and synchronizes the module

/rest/emeter/format
Returns the formatting information for measurements
Object: **sepobjs.xsd:SEPMeterFormat**
Root element: **MeterFormat**

/rest/emeter/format/query
Queries the meter formatting information

/rest/emeter/metering
Returns all the retrieved operational and reporting attributes from the meter
Object: **sepobjs.xsd:SEPMeteringReport**
Root element: **AMIMetering**

/rest/emeter/metering/log
Returns the log for meter events (see section 6.0 for format of the log)

/rest/emeter/metering/fastpoll?[mode]
Causes ISY to go into fast poll mode or back to normal mode:
mode={“on”,”off”}

/rest/emeter/query
Queries the meter for default attributes. The defaults may be different based on requirements from each utility provider

/rest/emeter/query/attr
Queries up to 5 specific attributes of the meter :
/rest/emeter/query/attr/***attribute_id_1***[/attribute_id_2[_5]]
Where ***attribute_id_x*** is defined in **sepobjs.SEPMeteringAttribute**

14.17.5 Billing

/rest/billing/today[/reset]
Returns the billing information for today **Billobjs.xsd::Billing**

/rest/billing/yesterday[/reset]
Returns the billing information for yesterday **Billobjs.xsd::Billing**

/rest/billing/cycle[/reset]
Returns the billing information for the current cycle **Billobjs.xsd::Billing**

/rest/billing/month/year
Returns the billing information for the given month/year cycle **Billobjs.xsd::Billing**

/rest/billing/report/today
Returns the hourly billing information for today **Billobjs.xsd::BillingReport**

/rest/billing/report/yesterday
Returns the hourly billing information for yesterday **Billobjs.xsd::BillingReport**

/rest/billing/report/month/year
Returns the daily billing information for the given month/year cycle **Billobjs.xsd::BillingReport**

14.18 Logs

Each event type has its own log which can be retrieved by using the respective REST interface.

All logs are tab delimited and end with a new line (\n) and follow the following format for the first 4 elements:

a. **Time Format:** M = Military Time, A = AM/PM

b. **Time of Event:** when the event was received or its state changed

c. **Type of Entry:** Log, Error, System Startup
   In case of Error/System Startup, the next entry is text
   In case of Log, the next entries depend on the type of event defined below

d. **Source of Event:** B = Broadband, M = Meter, A = Any

14.18.1 Message

/rest/emeter/message/log

/rest/emeter/message/log?reset=true
Clears the message log
In case of Error and System Startup, this entry is Text.
In case of Log, the following additional attributes are tab delimited:

a. **Event ID**
b. **Event State:** [Expired | Done | Scheduled | Running | Restored]
c. **Status:** [Confirmed | Unconfirmed | NA]
d. **Start Time**
e. **Duration:** in minutes
f. **Priority:** [Low | Medium | High | Critical | NA]
g. **Requires Confirmation:** [true | false]
h. **Message:** text

14.18.2 Price

/rest/emeter/price/log

/rest/emeter/price/log?reset=true
Clears the price log
In case of Error and System Startup, this entry is Text.
In case of Log, the following additional attributes are tab delimited:

a. **Event ID**
b. **Event State:** [Expired | Done | Scheduled | Running | Restored]
----- *If Expired/Done, the entry ends here. Otherwise:*
c. **Start Time**
d. **Duration:** in minutes
e. **Currency:** [USD | CAD | NA]
f. **Unit of Measure:** kWh
g. **Trailing Digits**
h. **Price**
i. **Number of Tiers**
j. **Tier**
k. **Label**

14.18.3 DRLC

/rest/emeter/drlc/log

/rest/emeter/drlc/log?reset=true
Clears the DRLC log
In case of Error and System Startup, this entry is Text.
In case of Log, the following additional attributes are tab delimited:

a. **Event ID**
b. **Event State:** [Expired | Done | Scheduled | Running | Restored]
c. **Opt Status: [**Unconfirmed | Opted In | Opted Out | NA]
----- If Expired/Done, then only Stop Reason attribute follows and the entry ends. Otherwise Stop Reason is removed and the entry continues with the Start Time
d. **Stop Reason: [**Completed | Canceled | Superseded | Opted Out | NA]
e. **Start Time**
f. **Duration:** in minutes
g. **Enrollment Group**
h. **Device Class:** All if all device classes otherwise the number
i. **Criticality:** [Unknown | Green | 1 | 2 | 3 | 4 | 5 | Emergency | Planned Outage | Service Disconnect | NA]
j. **Cooling Offset**
k. **Heating Offset**
l. **Cooling Setpoint**
m. **Heating Setpoint**
n. **Average Load Adjustment**
o. **Duty Cycle**

14.18.4 Meter

/rest/emeter/metering/log

/rest/emeter/metering/log?reset=true
Clears the Metering log
In case of Error and System Startup, this entry is Text.
In case of Log, the following additional attributes are tab delimited:

a. Attribute Name
For a list of attribute values, please peruse ***sepobjs.SEPMeteringAttribute***
b. Attribute Value
Numeric value

15 ISY 994 Z Series Energy Monitoring Developer's Manual

Supporting: Zigbee Brultech ECM1240, Zigbee Brultech GreenEye Monitor, and Zigbee UDI EM3

Web Services SDK and REST Interface, *Based on firmware 4.3.1*

15.1 Introduction

ISY994 Z Series incorporates sophisticated energy management capabilities to the base ISY platform supporting Zigbee Brultech ECM1240/GreenEye Monitor and UDI's EM3 3 Phase Energy Monitoring product. As such, all ISY interfaces, services, and events are applicable to 994Z as well.

ISY994 Z series comes equipped with an integrated high powered Zigbee radio operating on a Zigbee PRO stack. Utilizing the APIs, you can configure all parameters on Brultech ECM1240/GreenEye Monitor and EM3 wirelessly and through Zigbee.

Upon startup, ISY either establishes a PAN (as a Coordinator) or starts operating on the PAN that was already established prior to reboot. It's quite important to make sure that EM3 and ECM1240 are searching and joining the correct PAN and sending events to the correct end point. As such, there are two phases for the correct operation of the system:

1. Setup ISY for a specific PAN ID and channel mask that is known not to interfere with other RF devices such as Wifi systems.
2. ECM1240:
Setup so that ECM can search for the PAN ID configured in ISY, set source and destination endpoints, and ensure that ECM1240 is setup with the correct network and link keys (using encryption)
3. GreenEye Monitor
Consult GreenEye documentation to make sure GreenEye is configured for ISY Zigbee network parameters
4. EM3:
Setup so that EM3 can search for the PAN ID configured in ISY

Upon successful configuration, ECM1240/GreenEye Monitor and EM3 automatically scan and join the PAN and starts publishing energy events.

Depending on the product, different nodes are added to the device tree representing each channel. As with the rest of ISY platform, you can use the REST interface to get properties for each node.

15.2 Getting Started

ISY994 Z Series is based on the same framework as ISY and therefore communications and event infrastructure follow the same paradigm. If you have not yet reviewed ISY's WSDK Developer's guide, please send an email to sales@universal-devices.com.

If you do not already have Energy Monitoring Module installed on ISY, please send an email to sales@universal-devices.com with your UUID (Help | About) and your desire to have Energy Monitoring Module activated.

15.2.1 Configuring ISY

Setup Zigbee network as depicted in below.

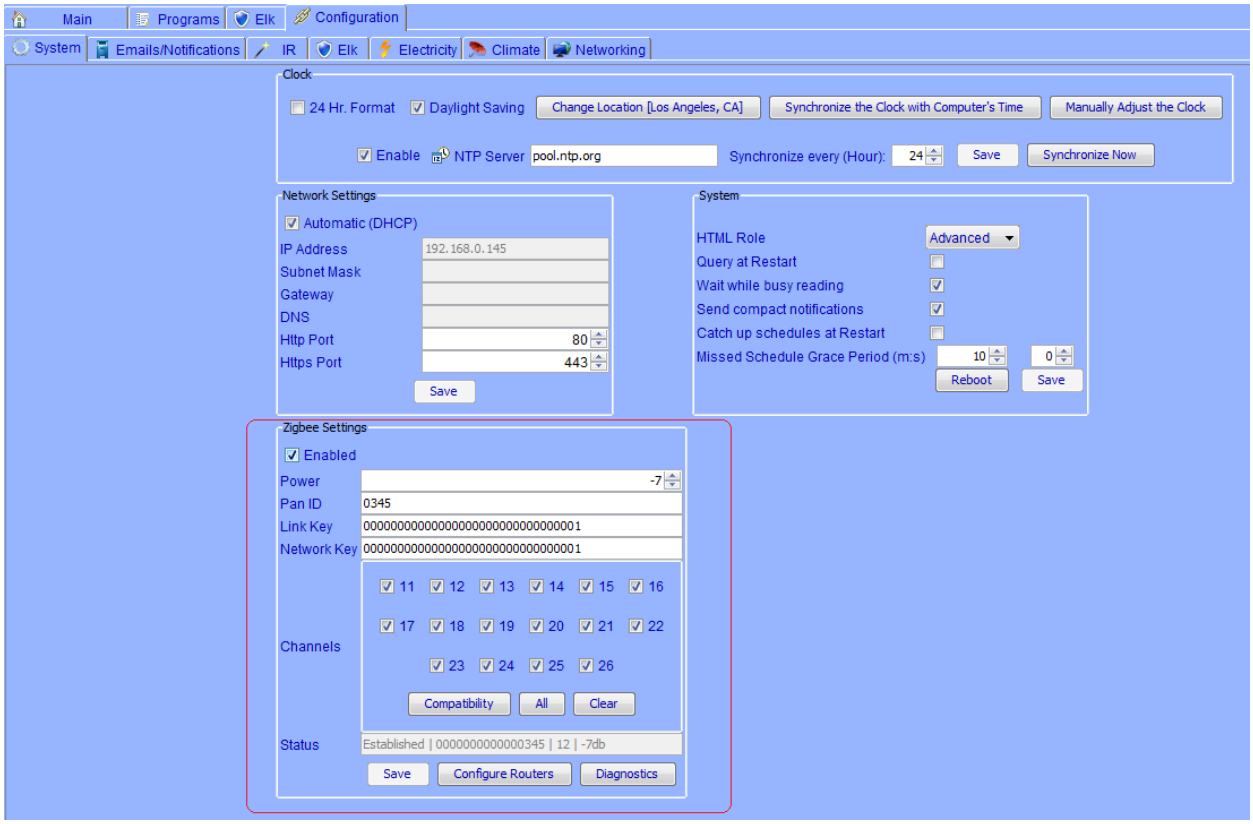


Figure 37: Setting up Zigbee Network

15.2.2 Configuring ECM1240

As mentioned before, ECM1240 needs to be configured to scan for and join ISY. Since ISY uses Zigbee PRO, it's important that the following parameters are set accurately:

1. Use Zigbee 2 Profile
   ATZS2
2. Enable Encryption
   ATEE1
3. Set Network Key
   ATKY1
4. Set PAN ID
   ATID[PANID] … see Figure **37**: Setting up Zigbee Network
5. Set Destination Endpoint
   ATDE2
6. Save and restart
   ATWR
   ATNR

At this point, ECM1240 should start scanning for ISY with the given PAN ID and join it if found.

15.2.3 Configuring GreenEye Monitor

GreenEye Monitor should already be setup for communications with ISY. Consult GreenEye documentation.

15.2.4 Configuring UDI EM3

UDI EM3 should automatically find ISY as long as the Network and Link Keys are set to 1 (see Figure **37**: Setting up Zigbee Network).

15.3 3. Nodes, Properties and Events

Just like any other device in ISY, Energy Monitoring devices are represented as Nodes for each channel. Each node may have different properties (and associated events) all of which are easily retrieved using the same REST command used for other nodes in ISY:
/rest/nodes/<node_id>

This said, unlike INSTEON devices – and in addition to device category/sub category – one has to inspect the ***<family>*** element in the node:

7 – UDI EM3: defined in 7_fam.xml
8 – ECM 1240: defined in 8_fam.xml

15.3.1 ECM 1240/GreenEye Monitor Nodes

ECM 1240 is represented by 7 nodes for 7 channels (See Figure **38**: ECM 1240 Nodes). The address for the main node ends with **1**.

For reference, the following table depicts the relationship between nodes, addresses, and properties:

Address <-> Channel	Supported Properties
1 ←→1	ST = Current Power TPW = Current Energy PPW = Polarized Power CV = Current Voltage CC = Current Current
2 ←→2	ST = Current Power TPW = Current Energy PPW = Polarized Power CC = Current Current
3 ←→3	ST = Current Power TPW = Current Energy
4 ←→4	ST = Current Power TPW = Current Energy
5 ←→5	ST = Current Power TPW = Current Energy
6 ←→6	ST = Current Power TPW = Current Energy
7 ←→7	ST = Current Power TPW = Current Energy

For GreenEye, there shall be 32 nodes for 32 channels.

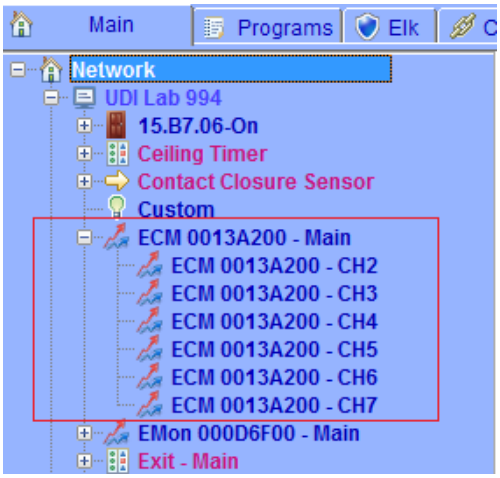


Figure 38: ECM 1240 Nodes

15.3.2 UDI EM3 Nodes

UDI EM3 is represented by 11 nodes for 5 channels (See **Figure 39: UDI EM3 Nodes**), 3 temperature sensors and 2 pulse counters. The address for the main node ends with **1**.

For reference, the following table depicts the relationship between nodes, addresses, and properties:

Address <-> Channel	Supported Properties
1 ←→Main	ST = Current Power TPW = Current Energy For all channels
5 ←→Channel 1	ST = Current Power TPW = Current Energy PF = Power Factor CV = Current Voltage CC = Current Current
6 ←→Channel 2	ST = Current Power TPW = Current Energy PF = Power Factor CV = Current Voltage CC = Current Current
7 ←→Channel 3	ST = Current Power TPW = Current Energy PF = Power Factor CV = Current Voltage CC = Current Current
8 ←→Channel 4	ST = Current Power TPW = Current Energy
9 ←→Channel 5	ST = Current Power TPW = Current Energy
40 ←→Local Temp.	ST
41 ←→Remote Temp1	ST
42 ←→Remote Temp2	ST
60 ←→Pulse Counter1	ST
61 ←→Pulse Counter2	ST

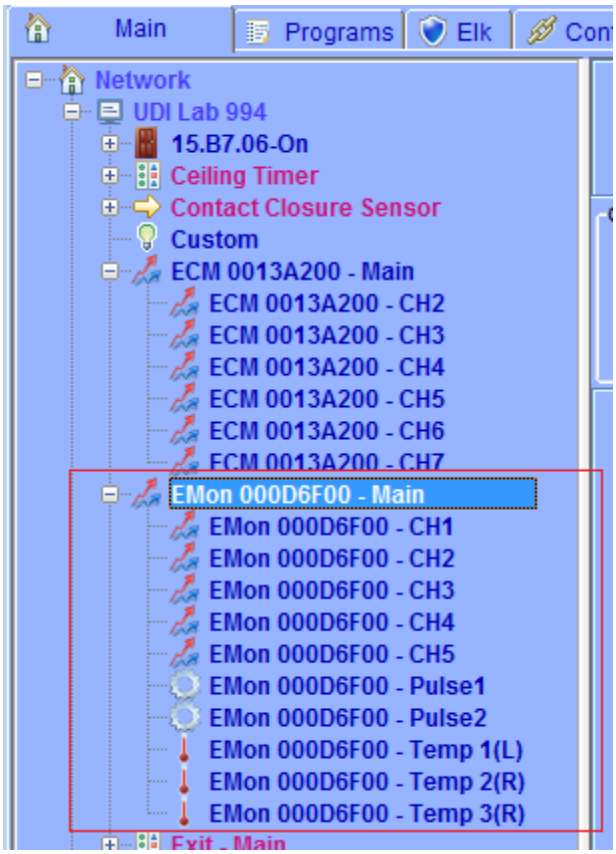


Figure 39: UDI EM3 Nodes

15.3.3 Events and Properties

The following events/controls/properties are defined for Energy Monitoring nodes. This said, not all nodes support all properties. One has to use /rest/nodes/<node_id> to inspect the supported controls:

TPW: Total Power (in kWh)
PPW: Polarized Power (in kWh)
PF: Power Factor
CC: Current Current (in Amps)
CV: Current Voltage (in Volts)
ST: Node dependent:
Energy Channel (in Watts)
Temp Sensor (in Degrees)
Pulse Counter (number of pulses)

15.3.4 Raw ECM140 Packet (control = _13 action = “7”)

node = null
<**eventInfo**>
[![CDATA]
Raw binary packet directly from Brultech]
</**eventInfo**>

15.4 REST Interface

REST is an easy to use URL based command set which allows the developer to communicate and control ISY’s Energy Management services including communications with ECM1240 and UDI EM3. Each ISY can support up to 32 Zigbee end points 16 of which can be energy monitors

15.4.1 Zigbee Network

/rest/zb
* Consult zigbee.xsd

15.4.2 REST Interface for ECM

Prefix: /rest/nodes/<nodeId>/cmd

/stopRT
Stops Real Time reporting for an ECM

/startRT
Starts Real Time reporting for an ECM

/reset
Resets accumulated values for an ECM

/cfg
Retrieves all the configuration information for the given ECM the XML for which is as follows:

<**EMonConfig**>
<**ct1** type="167" range="4"/>
<**ct2** type="167" range="4"/>
<**pt** type="131" range="6"/>
<**rtInterval**>real time interval (sec) </**rtInterval**>
<**dlInterval**>data logger interval (sec) </**dlInterval**>
<**firmware**>1026</**firmware**>
<**id**>unique id for the unit</**id**>
<**serial**>serial number for the unit</**serial**>
<**aux** constant="151" ***options***="62"> (see /setAux service)
<trim1>0</trim1>
<trim2>0</trim2>
<trim3>0</trim3>
<trim4>0</trim4>
<trim5>0</trim5>
<trim6>0</trim6>
</**aux**>
<**k1** h="142" l="141"/>
<**k2** h="142" l="141"/>
<**kv0**>212</**kv0**>
<**option**>0</**option**>
<**trigger**>200</**trigger**>
</**EMonConfig**>

/setCT?type=<T>&range=<R>
Sets CT configurations for a given channel (identified by the node). Only channels 1 and 2 are supported.
Where:
T = Types as defined by ECM (100 | 167)
R = Range as defined by ECM (3 | 4 | 5)

/setPT?type=<T>&range=<R>
Sets PT configurations on the given ECM.
Where:
T = Types as defined by ECM (?)
R = Range as defined by ECM (?)

/setInt?value=<V>
Sets Real Time interval (in seconds) on the given ECM.
Where:
V = The interval in seconds

/setTrig?value=<T>
Sets the Trigger value (in watts) on the given ECM.
Where:
T = The trigger value in Watts

/setAux?value=<O>
Sets the Aux options on the given ECM.
Where:
O = A byte bitmap as follows (Gain is X2):
Bit 0 = Aux 1 Gain
Bit 1 = Aux 2 Gain
Bit 2 = Aux 3 Gain
Bit 4 = Aux 4 Gain
Bit 5 = Aux 5 Gain
Bit 5 = Aux 5 Is used for Counting when set
Bit 6 = Aux 5 is DC bipoloar
Bit 7 = ?

Please note that these options are retrieved in the configuration in the **option** attribute of the **aux** element:
<**aux** constant="151" ***options***="62">

/toggPol
Toggles the polarity for the given channel (node) on the given ECM.

15.4.3 REST Interface for GreenEye Monitor

Prefix: /rest/nodes/<nodeId>/cmd

/stopRT
Stops Real Time reporting for an ECM

/startRT
Starts Real Time reporting for an ECM

/reset?type=<T>
Resets certain counters based on the type.
Where T:
1: Reset Pulse Counter 1
2: Reset Pulse Counter 2

3: Reset Pulse Counter 3
4: Reset Pulse Counter 4
5: Reset All Pulse Counters
6: Reset All Counters
7: Reset All Seconds Counters
8: Reset Seconds Counter for the Node in <nodeId>

/cfg
Not implemented.

/setCT?type=<T>&range=<R>
Not Implemented

/setPT?type=<T>&range=<R>
Sets PT configurations on the given ECM.
Where:
T = Types as defined by ECM (?)
R = Range as defined by ECM (?)

/setInt?value=<V>
Sets Real Time interval (in seconds) on the given ECM.
Where:
V = The interval in seconds

/setTrig?value=<T>
Not Implemented

/setAux?value=<O>
Not Implemented

/toggPol
Not Implemented

15.4.4 4.4 REST Interface for EM3

Prefix: /rest/nodes/<nodeId>/cmd

/stopRT
Stops Real Time reporting for an EM3

/startRT
Starts Real Time reporting for an EM3

/reset?type=<RT>*
Resets various parameters in EM3.
*<RT> can be a bitwise OR of the following:

1 = Reset accumulated values
2 = Reset configuration parameters
4 = Reset Zigbee
8 = Reset Pulse Count

/cfg
Retrieves all the configuration information for the given EM3 the XML for which is as follows:

<EM3Config debug="Boolean" realtime="Boolean" tempUnit="F|C" kyzMode="Boolean"**>**
<powerReportInterval>integer (seconds)</**powerReportInterval**>
<vaReportInterval>integer (seconds)</**vaReportInterval**>
<tempReportInterval>integer (seconds)</**tempReportInterval**>
<pulseReportInterval>integer (seconds)</**pulseReportInterval**>
<powerStorageInterval>integer (seconds)</**powerStorageInterval**>
<pulseStorageInterval>integer (seconds)</**pulseStorageInterval**>
</EM3Config>

/setCT?type=<T>&range=<R>
Currently not supported.

/setPT?type=<T>&range=<R>
Currently not supported.

/setInt?type=<T>&value=<V>
Sets Real Time interval (in seconds) for a various parameters, defined in T, on the given EM3.
Where:
T: could be any of the following:

- 1 = Power reporting interval
- 2 = Voltage/Current/Powerfactor (va) reporting interval
- 3 = Temperature reporting interval
- 4 = Pulse reporting interval
- 5 = Energy storage interval
- 6 = Pulse storage interval

V: The interval in seconds

/setTrig?value=<T>
Sets the Trigger value (in watts) on the given EM3.
Where:
T = The trigger value in Watts
/setOption?type=<OT>&value=<O>
Sets operating parameters for the EM3.
Where:
OT = is an option type which could be any of the following:

- 1 = Temp unit
  Where O could be:
  - 0 = F
  - 4 = C
- 2 = Debug
  Where O could be:
  - 0 = Debug Off
  - 1 = Debug On
- 3 = KYZ Mode
  Where O could be:
  - 0 = KYZ Mode Off
  - 1 = KYZ Mode On

Please note that these options are retrieved in the configuration in the **option** attribute of the **aux** element:
<**aux** constant="151" ***options***="62">

16 ISY 994 Z Series RCS Thermostat Support Scheduling Addendum

Supporting: TZ45

Web Services SDK, *Based on firmware 4.3.1*

16.1 Introduction

ISY994 Z Series incorporates sophisticated energy management capabilities to the base ISY platform supporting a variety of Zigbee devices including RCS Zigbee thermostats/load controllers. As such, all ISY interfaces, services, and events are applicable to 994Z as well.

ISY994 Z series comes equipped with an integrated high powered Zigbee radio operating on a Zigbee PRO stack. Utilizing the APIs, you can configure all parameters on your RCS Zigbee thermostats/load controllers wirelessly and through Zigbee.

Upon startup, ISY either establishes a PAN (as a Coordinator) or starts operating on the PAN that was already established prior to reboot. It's quite important to make sure that supported Zigbee devices are searching and joining the correct PAN and sending events to the correct end point. As such, there are two phases for the correct operation of the system:

1. Setup ISY for a specific PAN ID and channel mask that is known not to interfere with other RF devices such as WiFi systems.
2. Configure RCS Thermostat to join the configured network

This document describes the web services method through which an ISY WS client can get and set thermostat's schedules.

Note: Please do ensure that your thermostat has firmware version .47 or above. If not, please contact RCS for remedy.

16.2 Getting Started

ISY994 Z Series is based on the same framework as ISY and therefore communications and event infrastructure follow the same paradigm. If you have not yet reviewed ISY's WSDK Developer's guide, please send an email to sales@universal-devices.com.

If you do not already have RCS Zigbee Thermostat Support on ISY, please send an email to sales@universal-devices.com with your UUID (Help | About) and your desire to have it activated.

16.2.1 Configuring ISY

Setup Zigbee network as depicted below.

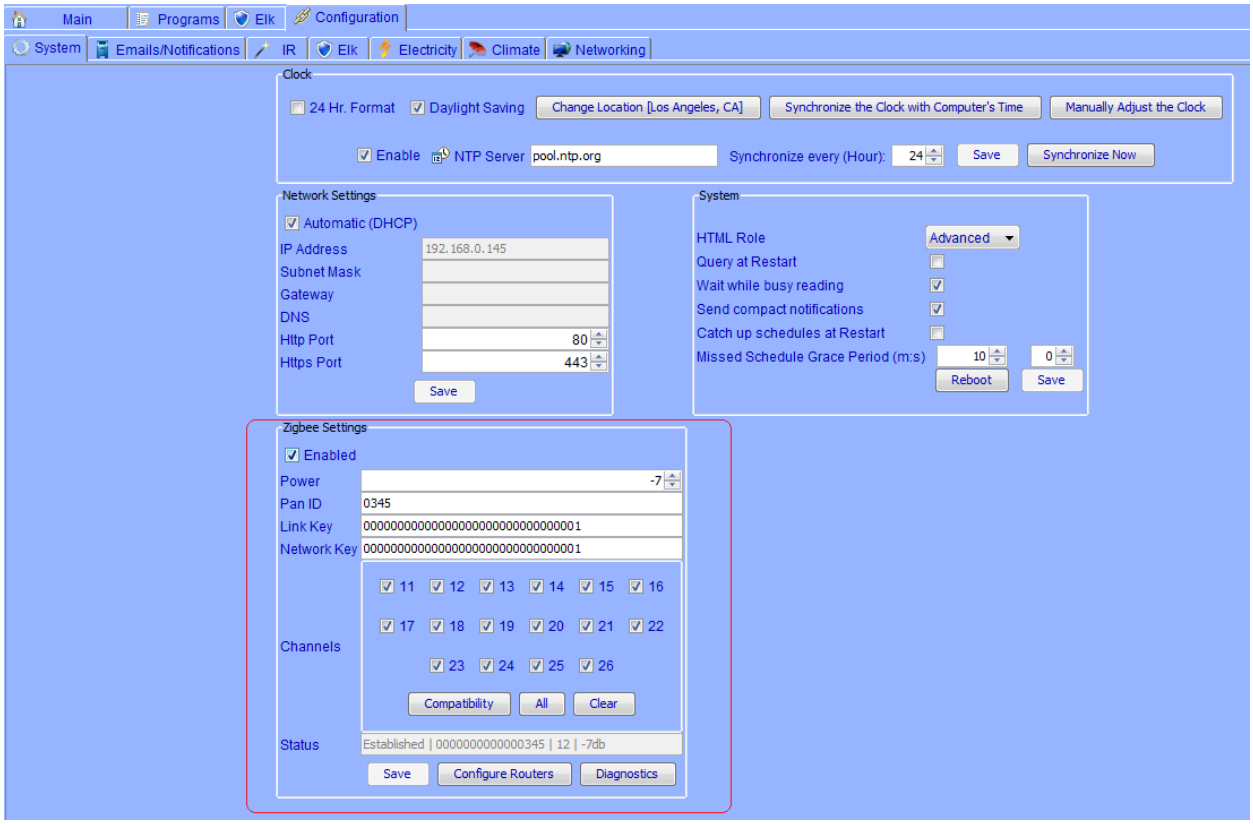


Figure 40: Setting up Zigbee Network

16.2.2 Configuring RCS Zigbee Thermostat

1. Click on the Menu button
2. Press and hold the two middle buttons between Menu and Hold/Away
3. Scroll down to Zigbee Install
4. Add your Thermostat to the network

16.3 Getting/Querying Schedules

To get if cached on ISY (will query if not cached):

```
<command>GETPROPS</command>
```

To query them use:

```
<command>QRYPROPS</command>
```

```
<s:Envelope>
    <s:Body>
        <u:DeviceSpecific xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
            <command>GETPROPS</command>
            <node>0013A200408D1120</node>
            <p1>SCHED</p1> <!-- Property category -->
            <p3>SCHED</p3> <!-- Property Name -->
        </u:DeviceSpecific>
    </s:Body>
</s:Envelope>
```

Returns:

mode – Predefined settings for a given schedule mode (currently only have settings for ‘away’)
day – Settings

For <day> entries, "0" means COOL and/or HEAT attribute shall not change that setpoint at that time

```
<schedule>
    <mode ID="away" COOL="83" HEAT="67"/>
    <day ID="sun">
        <e time="07:15" heat="0" cool="87"/>
        <e time="09:15" heat="68" cool="0"/>
        <e time="16:25" heat="0" cool="0"/>
        <e time="22:25" heat="72" cool="90"/>
    </day>
    <day ID="mon">
        <e TIME="06:05" COOL="78" HEAT="69"/>
        <e TIME="08:00" COOL="81" HEAT="66"/>
        <e TIME="16:00" COOL="78" HEAT="70"/>
        <e TIME="22:00" COOL="81" HEAT="66"/>
    </day>
```

```
    <day ID="tue">
        <e TIME="06:10" COOL="78" HEAT="69"/>
        <e TIME="08:00" COOL="81" HEAT="66"/>
        <e TIME="16:00" COOL="78" HEAT="70"/>
        <e TIME="22:00" COOL="81" HEAT="66"/>
    </day>
    <day ID="wed">
        <e TIME="06:15" COOL="78" HEAT="69"/>
        <e TIME="08:00" COOL="81" HEAT="66"/>
        <e TIME="16:00" COOL="78" HEAT="70"/>
        <e TIME="22:00" COOL="81" HEAT="66"/>
    </day>
    <day ID="thu">
        <e TIME="06:05" COOL="67" HEAT="56"/>
        <e TIME="08:05" COOL="66" HEAT="57"/>
        <e TIME="16:05" COOL="65" HEAT="58"/>
        <e TIME="22:05" COOL="64" HEAT="59"/>
    </day>
    <day ID="fri">
        <e TIME="06:20" COOL="79" HEAT="69"/>
        <e TIME="08:20" COOL="78" HEAT="68"/>
        <e TIME="16:20" COOL="77" HEAT="67"/>
        <e TIME="22:20" COOL="76" HEAT="66"/>
    </day>
    <day ID="sat">
        <e TIME="06:25" COOL="78" HEAT="70"/>
        <e TIME="08:25" COOL="78" HEAT="70"/>
        <e TIME="16:25" COOL="78" HEAT="70"/>
        <e TIME="22:25" COOL="81" HEAT="66"/>
    </day>
</schedule>
```

16.4 4. Setting Schedules

For <day> entries, specify "0" for COOL and/or HEAT attribute to not change that setpoint at that time

```
<s:Envelope>
    <s:Body>
    <u:DeviceSpecific xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
        <command>SETPROP</command>
        <node>0013A200408D1120</node>
        <p1>SCHED</p1> <!-- Property category -->
        <p3>SCHED</p3> <!-- Property Name -->
        <CDATA>
            <schedule>
                <mode id="away" heat="67" cool="83"/>
                <day id="sun">
                    <e time="07:15" heat="0" cool="87"/>
                    <e time="09:15" heat="68" cool="0"/>
                    <e time="16:25" heat="0" cool="0"/>
                    <e time="22:25" heat="72" cool="90"/>
                </day>
                <day id="mon">
                    <e time="06:05" heat="69" cool="78"/>
                    <e time="08:00" heat="66" cool="81"/>
                    <e time="16:00" heat="70" cool="78"/>
                    <e time="22:00" heat="66" cool="81"/>
                </day>
                <day id="tue">
                    <e time="06:10" heat="69" cool="78"/>
                    <e time="08:00" heat="66" cool="81"/>
                    <e time="16:00" heat="70" cool="78"/>
                    <e time="22:00" heat="66" cool="81"/>
                </day>
                <day id="wed">
                    <e time="06:15" heat="67" cool="78"/>
                    <e time="08:00" heat="66" cool="81"/>
                    <e time="16:00" heat="70" cool="78"/>
                    <e time="22:00" heat="66" cool="81"/>
                </day>
                <day id="thu">
                    <e time="06:05" heat="56" cool="67"/>
```

```
                    <e time="08:05" heat="57" cool="66"/>
                    <e time="16:05" heat="58" cool="65"/>
                    <e time="22:05" heat="59" cool="64"/>
                </day>
                <day id="fri">
                    <e time="06:20" heat="69" cool="79"/>
                    <e time="08:20" heat="68" cool="78"/>
                    <e time="16:20" heat="67" cool="77"/>
                    <e time="22:20" heat="66" cool="76"/>
                </day>
                <day id="sat">
                    <e time="06:25" heat="70" cool="78"/>
                    <e time="08:25" heat="70" cool="78"/>
                    <e time="16:25" heat="70" cool="78"/>
                    <e time="22:25" heat="66" cool="81"/>
                </day>
            </schedule>
        </CDATA>
    </u:DeviceSpecific>
    </s:Body>
</s:Envelope>
```

16.5 5. Events

Schedule events are identified by CLISMD Controls the actions for which are listed below:

```
<control>
    <name>CLISMD</name>
        <label>Schedule Mode</label>
        <readOnly>false</readOnly>
        <isQueryAble>true</isQueryAble>
        <isNumeric>false</isNumeric>
        <actions>
            <action>
                <name>0</name>
                <label>Hold</label>
            </action>
            <action>
                <name>1</name>
                <label>Run</label>
            </action>
            <action>
                <name>2</name>
                <label>Away</label>
            </action>
        </actions>
    </control>
```

17 Soap / Web Services (WSDK)

17.1 Soap Error Codes

Number	Error	Message
200	SOAP OK	Request Succeeded
401	Invalid Action	The HTTP Action header (SOAPAction) is invalid
402	Invalid Arguments	
403	Forbidden	
500	Internal Error	
501	SOAP Action Failed	Could not process the request SOAP message
600	Invalid Argument Value	Was expecting a different value for the given argument
601	Argument Value Out of Range	
602	Not Implemented	Returned when given SOAP Request is not implemented
603	Out of Memory	ISY is out of memory and might require reboot
604	Human Intervention Required	
605	String Argument Is Too Long	
609	Not Encrypted	Not used in 2.6.1 and above
612	Invalid Session	Not used in 2.6.1 and above
701	Not Authorized	Credentials are either wrong or not provided
711	Signature Failed	Not used in 2.6.1 and above
712	Signature Missing	Not used in 2.6.1 and above
714	Invalid Sequence	Not used in 2.6.1 and above
715	Invalid Control URL	Control URL is the location where SOAP messages are Posted to
721	Algorithm Not Supported	Not used in 2.6.1 and above
781	No Such Session	The requested session does not exist
801	Exhausted Secure Sessions	No more sessions are available. In some cases, this may require reboot
802	Deivce Error	Could not communicate with a device
803	Is In Linking Mode	When ISY is in Linking mode, most other operations fail returning this code
810	Subscription: Incompatible Headers	Subscriptions require headers as defined in the developers guide
811	Subscirption: Missing Callback	Need CALLBACK: URL in the HTTP Header. <URL> could be <REUSE_SOCKET>
812	Invalid NT	Not used in 2.6.1 and above
813	Subscription: SID Not Found	The Subscription for the given SID does not exist
814	Subscription: Invalid Callback URL	If <REUSE_SOCKET> is not used, the URL is pointing to a destination that is not reachable
815	Subscription: MAX Subscribers Reached	You can no longer subscribe
816	Disclaimer not available	You may ignore this error
817	Subscription: Already subscribed	You are already subscribed and there's no need to subscribe again

17.2 WSDL Core Web Services Documentation

 Web Services Flag, Objects, Parameters, Messages, and Bindings for ISY

UDI Binding binds the concrete Port (UDI Services) to the Abstract Port Type (UDIServices_PortType)

Port type UDIServices_PortType

17.2.1 AddDDNSHost

Description	Registers and DDNS host with the associated IP
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	AddDDNSHostRequest (soap:body, use = literal)
	AddDDNSHost type *AddDDNSHost* • host type *string* • ip type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* • status type *string* • info - optional; type *string*

17.2.2 AddFolder

Description	Adds a folder
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	AddFolderRequest (soap:body, use = literal)
	AddFolder type *AddFolder* • id type *string* • name type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* • status type *string* • info - optional; type *string*

17.2.3 AddGroup

Description	Adds a scene/group
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	AddGroupRequest (soap:body, use = literal)
	AddGroup type *AddGroup* • id type *string* • name type *string* • flag type *NodeTypeFlag* - type *byte* with restriction - enum { '1', '4', '8' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* • status type *string* • info - optional; type *string*

17.2.4 AddNode

Description	Adds a predefined node for a device with a given address
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	AddNodeRequest (soap:body, use = literal)
	AddNode type *AddNode* • id type *string* • name type *string* • type type *string* • flag type *NodeOperationsFlag* - type *byte* with restriction - enum { '1', '2', '3', '4' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* • status type *string* • info - optional; type *string*

17.2.5 ClearLastError

Description	Clears the list of recent errors in ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ClearLastErrorRequest (soap:body, use = literal)
	ClearLastError type *ClearLastError*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* • status type *string* • info - optional; type *string*

17.2.6 GetCurrentSystemStatus

Description	Sends the current system status (whole) to the given subscriber
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetCurrentSystemStatusRequest (soap:body, use = literal)
	GetCurrentSystemStatus type *GetCurrentSystemStatus* • SID type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* • status type *string* • info - optional; type *string*

17.2.7 GetDDNSHost

Description	Returns the currently configured DDNS host
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetDDNSHostRequest (soap:body, use = literal)
	GetDDNSHost type *GetDDNSHost*
Output	GetDDNSHostResponse (soap:body, use = literal)
	DDNSHost type *DDNSHost* DDNS host object. The name is mandatory. • name type *string* • ip - optional; type *string*

17.2.8 GetDebugLevel

Description	Gets the debug options and current level
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetDebugLevelRequest (soap:body, use = literal)
	GetDebugLevel type *GetDebugLevel*
Output	GetDebugLevelResponse (soap:body, use = literal)

	DBG type *DBG*
	Debug structure
	• options type *DebugOptions*
	  ○ The possible debug levels that can be chosen
	  ▪ option - unbounded; type *string*
	• current type *string*

17.2.9 GetISYConfig

Description	Returns the current configuration of ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetISYConfigRequest (soap:body, use = literal)
	GetISYConfig type *GetISYConfig*
Output	GetISYConfigResponse (soap:body, use = literal)
	configuration type *configuration* ▪ deviceSpecs type *DeviceSpecifications* Describes the characteristics of the ISY hardware/firmware and some info about the underlying protocol ▪ make - optional; type *string* ▪ manufacturerURL - optional; type *string* ▪ model type *string* ▪ icon - optional; type *string* ▪ archive type *string* ▪ chart type *string* ▪ queryOnInit type *boolean* Whether or not ISY queries all devices upon reboot ▪ oneNodeAtATime type *boolean* Whether or not nodes are queried one at a time ▪ upnpSpecs type *UPnPSpecifications* ▪ upnpDevice type *UPnPInfo* Provides pertinent information about how ISY will interact with the UPnP subsystem ▪ utype type *string* The UPnP device/service type which is advertised and responded to ▪ version type *string*

- upnpService type *UPnPInfo*
  - utype type *string*

    The UPnP device/service type which is advertised and responded to

  - version type *string*
- controls - optional; type *Controls*
  - control - optional, unbounded; type *Control*
    - name type *ControlType*
    - label - optional; type *string*
    - readOnly - optional; type *boolean*

      default is true

    - isQueryAble - optional; type *boolean*

      default is false

    - isNumeric - optional; type *boolean*
    - numericUnit - optional; type *string*
    - min - optional; type *integer*
    - max - optional; type *integer*
    - isGlobal - optional; type *boolean*

      Applies to ISY as a whole

    - isGUI - optional; type *boolean*

      It has no correspomding value on the device side

    - actions - optional; type *Actions*
      - action - optional, unbounded; type *Action*
        - name type *string*
        - label type *string*
        - description - optional; type *string*
        - readOnly - optional; type *boolean*
- driver_timestamp type *string*
- app type *string*
- app_version type *string*
- platform type *string*
- build_timestamp type *string*
- baseDriver - optional; type *ProductDriverType*

  Product Drivers are tied into functionality for specific Protocol/Configuration

  - type type *string*

The type/name of the driver. e.g. INSTEON

- version type *string*

  ISY version for the driver

- root type *DeviceRoot*

  Defines the root Group for ISY.

  - id type *string*
  - name type *string*
- product type *Product*

  Describes the type of ISY for which this is the configuration

  - id type *string*
  - desc - optional; type *string*
- features - optional; type *Features*

  A list of ISY installed features/services

  - feature - optional, unbounded; type *Feature*

    An ISY feature or service.

    - id type *int*
    - desc - optional; type *string*
    - isInstalled - optional; type *boolean*
    - isAvailable - optional; type *boolean*
- triggers type *boolean*

  Whether or not Triggers are activated

- maxTriggers - optional; type *integer*

  Maximum number of triggers supported for this platform. If not present, assume 1024. Examples: 1024, 2048

- security - type *string* with restriction - enum { 'SSL', 'UPNP' }
- isDefaultCert - optional; type *boolean*

  Whether or not we are using the default SSL certificate which was shipped originally with ISY

- internetAccessURL - optional; type *string*

  The remote/extenal/internet URL by which one can access this system

	▪ secsys - optional; type *SecuritySystemType* Defines the type of security system(s) configured for ISY ○ type type *SupportedSecuritySystems* - type *string* with restriction - enum { 'ELK' } Supported Security Systems ○ version type *string* ISY version for the driver If a security system has been installed/configured, include the type/version here

17.2.10 GetLastError

Description	Gets the list of recent errors in ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetLastErrorRequest (soap:body, use = literal)
	GetLastError type *GetLastError*
Output	GetLastErrorResponse (soap:body, use = literal)
	LastError type *LastError* A list of DriverError(s) ▪ error - unbounded; type *DriverError* Driver error definitions ▪ code type *int* ▪ param1 - optional; type *string* ▪ param2 - optional; type *string*

17.2.11 GetNodesConfig

Description	Returns the configuration of nodes, groups, scenes, and any relationship thereto
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetNodesConfigRequest (soap:body, use = literal)
	GetNodesConfig type *GetNodesConfig*
Output	GetNodesConfigResponse (soap:body, use = literal)https://www.universal-devices.com/developers/wsdk/latest/docs/udiws30-all.html - src.id124638
	nodes type *nodes* The configuration of all the des already set in ISY ▪ root - optional; type *string* The name of ISY network. If blank, then it may be construed as the Network node. ▪ folder - optional, unbounded; type *folder* - extension of abstract type *UDNode* ▪ address type *string* The unique address for a node. This address is the one to be used in subsequent calls to ISY to impact a node/group/scene change ▪ name type *string* The user defined name for a device. ▪ family - optional; type *NodeFamilyID* This element defines the family of a device. Families are basically the protocol/manufacturer level categories for a device. For instance, RCS is a family and so is INSTEON or UPB. Using this element allows supporting multiple protocols using the same system. In case of a device node, each family may have different device types (such as thermostats vs. load controllers). Family definitions maybe found in [id]_fam.xml. e.g. 1_fam.xml defines INSTEON family whereas 3_fam.xml defines RCS ▪ parent - optional; type *NodeParent*

- type type *NodeHierarchyFlag* - type *byte* with restriction - enum { '0', '1', '2', '3' }

  The hierarchy type for the parent

- The address of the parent if any

- flag - required; type *short*

  A bit mask: 0x01 -- Node is initialized (internal) 0x02 -- Node is going to be crawled (internal) 0x04 -- This is a group node 0x08 -- This is the root node for ISY, i.e. My Lighting 0x10 -- Device Communications Error 0x20 -- Brand new node 0x40 -- Node shall be deleted 0x80 -- Node is device root

- Signifies a folder. Note that folders do not have ELK-IDs
- node - optional, unbounded; type *node* - extension of abstract type *UDNode*
  - address type *string*

    The unique address for a node. This address is the one to be used in subsequent calls to ISY to impact a node/group/scene change

  - name type *string*

    The user defined name for a device.

  - family - optional; type *NodeFamilyID*

    This element defines the family of a device. Families are basically the protocol/manufacturer level categories for a device. For instance, RCS is a family and so is INSTEON or UPB. Using this element allows supporting multiple protocols using the same system. In case of a device node, each family may have different device types (such as thermostats vs. load controllers). Family definitions maybe found in [id]_fam.xml. e.g. 1_fam.xml defines INSTEON family whereas 3_fam.xml defines RCS

  - parent - optional; type *NodeParent*
    - type type *NodeHierarchyFlag* - type *byte* with restriction - enum { '0', '1', '2', '3' }

      The hierarchy type for the parent

  - The address of the parent if any
- flag - required; type *short*

  A bit mask: 0x01 -- Node is initialized (internal) 0x02 -- Node is going to be crawled (internal) 0x04 -- This is a group node 0x08 -- This is the root node for ISY, i.e. My Lighting 0x10 -- Device

Communications Error 0x20 -- Brand new node 0x40 -- Node shall be deleted 0x80 -- Node is device root

- type type *string*

  The type of device. For INSTEON: device-cat.device-subcat.version.reserved (4 bytes) Please note that the if family element is present, then the device type is a category/sub category within the family. In most cases the device cat/subcat for all families are the same

- enabled - optional; type *boolean*

  Whether or not the node is enabled (plugged in). Note: this feature only works on 99 Series

- deviceClass - optional; type *DeviceClass*

  Type of device which may give a hint as to how to duty cycle

- wattage - optional; type *int*

  Approximate wattage for this device in watts

- dcPeriod - optional; type *int*

  Duty cycle period in minutes

- pnode - optional; type *string*

  The address of the primary node for the device partially represented by this node. If this node is the primary node then pnode will equal address. A device may be represented by one or more nodes. One of these nodes is designated the primary node and is used to help group the set of nodes for a device. Note: UPB Mandatory/INSTEON Optional.

- sgid - optional; type *string*

  The ID identifying the subgroup of end-points for the device. Each node representing a subset of the end-points for a device is identified by a unique subgroup ID. This is similar in function to the last digit in an Insteon Node Address. Note: UPB Mandatory/INSTEON Optional.

- qry - optional; type *Empty*

Whether or not a node is queryable. If this tag is not present, then node is not queryable. A tag was used for future extensibility. Note: Currently UPB Only.

- ctl - optional; type *Empty*

  Whether or not a node may be used as a controller. If this tag is not present, then node is may not be used as a controller. A tag was used for future extensibility. Note: Currently UPB Only.

- tx - optional; type *unsignedInt*

  Bitmask indicating the controller endpoints represented by this node. A tag was used for future extensibility. Note: Currently UPB Only.

- rx - optional; type *unsignedInt*

  Bitmask indicating the responder endpoints represented by this node. A tag was used for future extensibility. Note: Currently UPB Only.

- rsp - optional; type *ResponderType*
  - type - required; type *ResponderTypes*

  If present, then this node may be used as a scene responder. The type of responder is defined in the type attribute. Note: Currently, UPB Only (see upb.xsd)

- ELK_ID - optional; type *string*

  ELK unique identifier for each node/scene/group

- group - optional, unbounded; type *group* - extension of abstract type *UDNode*
  - address type *string*

    The unique address for a node. This address is the one to be used in subsequent calls to ISY to impact a node/group/scene change

  - name type *string*

    The user defined name for a device.

  - family - optional; type *NodeFamilyID*

    This element defines the family of a device. Families are basically the protocol/manufacturer level categories for a device. For instance, RCS is a family and so is INSTEON or UPB. Using this element allows supporting multiple

protocols using the same system. In case of a device node, each family may have different device types (such as thermostats vs. load controllers). Family definitions maybe found in [id]_fam.xml. e.g. 1_fam.xml defines INSTEON family whereas 3_fam.xml defines RCS

- parent - optional; type *NodeParent*
  - type type *NodeHierarchyFlag* - type *byte* with restriction - enum { '0', '1', '2', '3' }

    The hierarchy type for the parent

  - The address of the parent if any
- flag - required; type *short*

  A bit mask: 0x01 -- Node is initialized (internal) 0x02 -- Node is going to be crawled (internal) 0x04 -- This is a group node 0x08 -- This is the root node for ISY, i.e. My Lighting 0x10 -- Device Communications Error 0x20 -- Brand new node 0x40 -- Node shall be deleted 0x80 -- Node is device root

- deviceGroup - optional; type *string*

  The underlying group/scene number associated with this scene. Normally, not used except for programs made for debugging

- ELK_ID - optional; type *string*

  ELK unique identifier for each node/scene/group

- members - optional; type *UDLinkMembers*
  - link - unbounded; type *UDLink*
    - type type *short*

      0x00 -- Node is neither a controller or responder for the enclosing scene. i.e. devices under the My Lighting node 0x10 -- Node is a controller for the enclosing scene 0x20 -- Node is a responder for the enclosing scene

  - The unique address for a node.
- The address and the type (controller vs. responder) of devices participating in a group/scene

17.2.12 GetSMTPConfig

Description	Returns the SMTP Options configured in ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetSMTPConfigRequest (soap:body, use = literal)
	GetSMTPConfig type *GetSMTPConfig*
Output	GetSMTPConfigResponse (soap:body, use = literal)
	SMTPConfig type *SMTPConfig* SMTP Configuration UseDefaultSMTP = Whether or not we should use UDI's default SMTP Server SMTPServer = SMTP Server SMTPPort = SMTP Port SMTPUID = SMTP User ID SMTPPWD = SMTP Password SMTPFrom = SMTP From email address SMTPTimeout = SMTP Timeout to wait for each operation UseTLS = Whether or not we should use TLS ▪ UseDefaultSMTP - optional; type *boolean* ▪ SMTPServer type *string* ▪ SMTPPort type *int* ▪ SMTPUID - optional; type *string* ▪ SMTPPWD - optional; type *string* ▪ SMTPFrom - optional; type *string* ▪ SMTPTimeout - optional; type *int* ▪ UseTLS - optional; type *Boolean*

17.2.13 GetSceneProfiles

Description	Gets the profile attributes for responders. Set the controller element to scene's address and: In case this is an ISY scene, set the node element to "none." In case this is a controller for a scene, set the node element to the controller's address
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetSceneProfilesRequest (soap:body, use = literal)
	GetSceneProfiles type *GetSceneProfiles* ▪ node type *string* ▪ controller type *string*
Output	GetSceneProfilesResponse (soap:body, use = literal)
	SceneProfiles type *SceneProfiles* The structure which represents the profiles of responders for a scene ▪ SP - optional, unbounded; type *SceneProfile* The structure which represents the profile of responders for a scene OL = on level RR = ramp rate Climate will be added in release 3 ▪ node type *string* ▪ OL - optional; type *double* ▪ RR - optional; type *double* ▪ CLISPC - optional; type *double* ▪ CLISPH - optional; type *double* ▪ CLIFS - optional; type *int* ▪ CLIMD - optional; type *int*

17.2.14 GetStartupTime

Description	Returns a timestamp of when ISY was last started
Style	document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetStartupTimeRequest (soap:body, use = literal)
	GetStartupTime type *GetStartupTime*
Output	TimeResponseMessage (soap:body, use = literal)
	UDITimeResponse type *UDITimeResponse* ○ val - required; type *dateTime* Timestamp in the form of YYYYMMDD HH:MM:SS

17.2.15 GetSystemDateTime

Description	Returns the current state of ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetDateTimeRequest (soap:body, use = literal)
	GetSystemTime type *GetSystemTime*
Output	GetDateTimeResponse (soap:body, use = literal)
	DT type *DT* ISY's DateTime Structure: NTP = time in NTP format TMZOffset = Time zone Offset from GMT (in seconds) DST = Daylight Saving Time (true/false) Lat = Latitude (in degrees) Long = Longitude (in degrees) Sunrise = Sunrise time in NTP format (response only) Sunset = Sunset time in NTP format (response only) IsMilitary = Whether or not we should use 24 hour format ▪ NTP type *long* ▪ TMZOffset type *int* ▪ DST type *boolean* ▪ Lat type *float* ▪ Long type *float* ▪ Sunrise - optional; type *long* ▪ Sunset - optional; type *long* ▪ IsMilitary - optional; type *boolean*

17.2.16 GetSystemOptions

Description	Returns the options by which ISY is configured
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetSystemOptionsRequest (soap:body, use = literal)
	GetSystemOptions type *GetSystemOptions*
Output	GetSystemOptionsResponse (soap:body, use = literal)
	SystemOptions type *SystemOptions* ISY's System Options (configuration of how ISY works) MailTo = Comma delimited list of notification recipients CompactEmail = Whether or not to send short notifications QueryOnInit = Whether or not to query all devices at Restart PCacthUp = Whether or not programs should catch up upon startup PGracePeriod = The amount of time (in seconds) after which a schedule is still considered met NTPHost = NTP host name NTPActive = Whether or not ISY has been able to communicate with the NTP host NTPEnabled = Whether or not NTP checking is enabled NTPInterval = The amount of time (in seconds) between each NTP check WaitBusyReading = Whether or not ISY has to wait for "silence" before sending the next packet HTMLRole = The role of the HTML user ▪ MailTo type *string* ▪ CompactEmail type *boolean* ▪ QueryOnInit type *boolean* ▪ PCatchUp type *boolean* ▪ PGracePeriod type *int* ▪ NTPHost type *string* ▪ NTPActive type *boolean* ▪ NTPEnabled type *boolean* ▪ NTPInterval type *int* ▪ WaitBusyReading type *boolean* ▪ HTMLRole type *HTMLRoleFlag* - type *integer* with restriction - enum { '1', '2', '3', '4' }

17.2.17 GetSystemStatus

Description	Returns the current state of ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetSystemStatusRequest (soap:body, use = literal)
	GetSystemStatus type *GetSystemStatus*
Output	GetSystemStatusResponse (soap:body, use = literal)
	SysStat type *SysStat* ISY's System Status: NumRemainingOps = an estimate of how many operations are left to complete DelayBetweenOps (in miliseconds) = an estimate of any delay between each operation Note: you may assume 1 second per each operation ▪ Stat type *ISYStatusFlag* - type *byte* with restriction - enum { '0', '1', '2', '3', '10' } ▪ NumRemainingOps type *long* ▪ DelayBetweenOps type *long* ▪ InSafeMode type *Boolean*

17.2.18 GetVariable

Description	Retrieves a variable
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetVariableRequest (soap:body, use = literal)
	GetVariable type *GetVariable* Parameter for retrieving the value of a specific variable ▪ type type *UDIVariableType* - type *string* with restriction - enum { '1', '2' } ▪ id type *string*
Output	GetVariableResponse (soap:body, use = literal)
	var type *var* Defines a variable ▪ init type *string* Initial value of a variable ▪ val type *string* ▪ ts type *string* The last updated time stamp in YYYYMMDD HH:MM:SS ○ type - required; type *UDIVariableType* - type *string* with restriction - enum { '1', '2' } ○ id - required; type *string* The id of the variable

17.2.19 GetVariables

Description	Retrieves all variables of a certain type
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetVariablesRequest (soap:body, use = literal)
	GetVariables type *GetVariables* Parameter for retrieving all variables of a certain type ▪ type type *UDIVariableType* - type *string* with restriction - enum { '1', '2' }
Output	GetVariablesResponse (soap:body, use = literal)
	vars type *vars* list of variables ▪ var - optional, unbounded; type *Variable* Defines a variable ▪ init type *string* Initial value of a variable ▪ val type *string* ▪ ts type *string* The last updated time stamp in YYYYMMDD HH:MM:SS ○ type - required; type *UDIVariableType* - type *string* with restriction - enum { '1', '2' } ○ id - required; type *string* The id of the variable

17.2.20 InternetAccess

Description	Enables/Disables port forwarding to ISY on UPnP enabled routers. This process might take a long time and, as such, one must take caution with TCP connect timeouts
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	InternetAccessRequest (soap:body, use = literal)
	InternetAccess type *InternetAccess* ▪ flag type *InternetAccessFlag* - type *byte* with restriction - enum { '24', '38' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.21 IsDDNSHostAvail

Description	Whether or not a DDNS Host is available
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	IsDDNSHostAvailRequest (soap:body, use = literal)
	IsDDNSHostAvail type *IsDDNSHostAvail* ▪ host type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.22 IsSubscribed

Description	Whether or not the client still holds a subscription to ISY; 0 - Not subscribed 1 – Subscribed
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	IsSubscribedRequest (soap:body, use = literal)
	IsSubscribed type *IsSubscribed* ▪ SID type *string*
Output	IntResponseMessage (soap:body, use = literal)
	UDIIntResponse type *UDIIntResponse* ○ val - required; type *int* Return value of type integer. Currently used for IsSubscribed

17.2.23 MoveNode

Description	Moves a node into a group/scene
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	MoveNodeRequest (soap:body, use = literal)
	MoveNode type *MoveNode* ▪ node type *string* ▪ group type *string* ▪ flag type *LinkModeFlag* - type *byte* with restriction - enum { '16', '32' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.24 Query

Description	Queries a node, a scene, or even the whole network
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryRequest (soap:body, use = literal)
	QueryAll type *QueryAll* ▪ node type *string* ▪ flag type *NodeTypeFlag* - type *byte* with restriction - enum { '1', '4', '8' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.25 Reboot

Description	Reboots the system
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RebootRequest (soap:body, use = literal)
	Reboot type *Reboot*
Output	DefaultResponseMessage (soap:body, use = literal)https://www.universal-devices.com/developers/wsdk/latest/docs/udiws30-all.html - src.id125438
	DIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.26 RemoveDDNSHost

Description	Removes the registered DDNS host. Must use GetDDNSHost
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RemoveDDNSHostRequest (soap:body, use = literal)
	RemoveDDNSHost type *RemoveDDNSHost* DDNS host object. The name is mandatory. ▪ name type *string* ▪ ip - optional; type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.27 RemoveFolder

Description	Removes a folder (permanently) from configuration
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RemoveFolderRequest (soap:body, use = literal)
	RemoveFolder type *RemoveFolder* ▪ id type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.28 RemoveFromGroup

Description	Removes a Node from a Group
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RemoveFromGroupRequest (soap:body, use = literal)
	RemoveFromGroup type *RemoveFromGroup* ▪ node type *string* ▪ group type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.29 RemoveGroup

Description	Removes a group (permanently) from configuration
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RemoveGroupRequest (soap:body, use = literal)
	RemoveGroup type *RemoveGroup* ▪ id type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.30 RemoveModem

Description	Replaces the modem attached to ISY and reconfigures devices if necessary
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RemoveModemRequest (soap:body, use = literal)
	RemoveModem type *RemoveModem*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.31 RemoveNode

Description	Removes a node (permanently) from configuration
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RemoveNodeRequest (soap:body, use = literal)
	RemoveNode type *RemoveNode* ▪ id type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	DIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.32 RenameFolder

Description	Renames a folder in configuration
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RenameFolderRequest (soap:body, use = literal)
	enameFolder type *RenameFolder* ▪ id type *string* ▪ name type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.33 RenameGroup

Description	Renames a group in configuration
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RenameGroupRequest (soap:body, use = literal)
	RenameGroup type *RenameGroup* ▪ id type *string* ▪ name type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.34 RenameNetwork

Description	Renames the Network name
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RenameNetworkRequest (soap:body, use = literal)
	RenameNetwork type *RenameNetwork* ▪ name type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.35 RenameNode

Description	Renames a node in configuration
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RenameNodeRequest (soap:body, use = literal)
	RenameNode type *RenameNode* ▪ id type *string* ▪ name type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.36 ReplaceDevice

Description	Replaces one device with another (swap)
Style	document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ReplaceDeviceRequest (soap:body, use = literal)
	ReplaceDevice type *ReplaceDevice* ▪ node type *string* ▪ NewNode type *string* ▪ firmware type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.37 ReplaceModem

Description	Replaces the modem attached to ISY and reconfigures devices if necessary
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ReplaceModemRequest (soap:body, use = literal)
	ReplaceModem type *ReplaceModem*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.38 RestoreDevice

Description	Restores a specific device from the configuration in ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RestoreDeviceRequest (soap:body, use = literal)
	RestoreDeviceFromNode type *RestoreDeviceFromNode* ▪ node type *string* ▪ flag type *RestoreDevicesFlag* - type *byte* with restriction - enum { '0', '1' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.39 RestoreDevices

Description	Restores devices from the configuration in ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RestoreDevicesRequest (soap:body, use = literal)
	RestoreDevicesFromNodes type *RestoreDevicesFromNodes* ▪ flag type *RestoreDevicesFlag* - type *byte* with restriction - enum { '0', '1' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.40 SecuritySystemAction

Description	Arm/disarm a security system
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SecuritySystemActionRequest (soap:body, use = literal)
	SecuritySystemRequest type *SecuritySystemRequest* Provide the command to be processed in the SecAction tag for example, see ELKActionCodes Provide the security code in the code tag ▪ SecAction type *string* ▪ code type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.41 SendHeartbeat

Description	Immediately sends a heartbeat message to all subscribers
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SendHeartbeatRequest (soap:body, use = literal)
	SendHeartbeat type *SendHeartbeat*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.42 SendTestEmail

Description	Sends a test email. This is used to check SMTP settings.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SendTestEmailRequest (soap:body, use = literal)
	SendTestEmail type *SendTestEmail* ▪ id type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.43 SetBatchMode

Description	Causes the system to go to/exit batch mode
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetBatchModeRequest (soap:body, use = literal)
	SetBatchMode type *SetBatchMode* ▪ flag type *BatchModeFlag* - type *integer* with restriction - enum { '0', '1' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.44 SetBatteryDeviceWriteMode

Description	Enables/disables writing to battery devices automatically
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetBatteryDeviceWriteModeRequest (soap:body, use = literal)
	SetBatteryDeviceWriteMode type *SetBatteryDeviceWriteMode* ▪ flag type *BatteryDeviceWriteFlag* - type *integer* with restriction - enum { '0', '1' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.45 SetDebugLevel

Description	Sets the debug level (from 1 to 3)
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetDebugLevelRequest (soap:body, use = literal)
	SetDebugLevel type *SetDebugLevel* ▪ option type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.46 SetLinkingMode

Description	Changes the way ISY discovers nodes/devices ISY as a controller or ISY as a responder
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetLinkingModeRequest (soap:body, use = literal)
	SetDeviceLinkMode type *SetDeviceLinkMode* ▪ flag type *LinkModeFlag* - type *byte* with restriction - enum { '16', '32' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.47 SetNTPOptions

Description	Sets NTP Options
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetNTPOptionsRequest (soap:body, use = literal)
	SetNTPOptions type *SetNTPOptions* NTPHost = NTP host name NTPEnabled = Whether or not NTP checking is enabled NTPInterval = The amount of time (in seconds) between each NTP check ▪ NTPHost type *string* ▪ NTPEnabled type *boolean* ▪ NTPInterval type *int*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.48 SetNodeEnabled

Description	Enables or disables a node
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetNodeEnabledRequest (soap:body, use = literal)
	SetNodeEnabled type *SetNodeEnabled* ▪ node type *string* ▪ flag type *SetNodeEnabledFlag* - type *byte* with restriction - enum { '0', '1' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.49 SetNodePowerInfo

Description	Sets the Power characteristics for a node
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetNodePowerInfoRequest (soap:body, use = literal)
	SetNodePowerInfo type *SetNodePowerInfo* Only available on Orchestrator series ▪ node type *string* ▪ deviceClass type *DeviceClass* ▪ wattage - optional; type *unsignedInt* ▪ dcPeriod - optional; type *unsignedInt* The total amount of time, in minutes, which a duty cycle event uses to calculate the amount of time the device should be on/off.
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.50 SetNotificationsOptions

Description	Sets Notifications Options
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetNotificationsOptionsRequest (soap:body, use = literal)
	SetNotOptions type *SetNotOptions* MailTo = Comma delimited list of notification recipients CompactEmail = Whether or not to send short notifications ▪ MailTo type *string* ▪ CompactEmail type *Boolean*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.51 SetParent

Description	Sets a parent for a node
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetParentRequest (soap:body, use = literal)
	SetParent type *SetParent* ▪ node type *string* ▪ nodeType type *NodeHierarchyFlag* - type *byte* with restriction - enum { '0', '1', '2', '3' } ▪ parent - optional; type *string* ▪ parentType - optional; type *NodeHierarchyFlag* - type *byte* with restriction - enum { '0', '1', '2', '3' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.52 SetProgramOptions

Description	Sets Program Options
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetProgramOptionsRequest (soap:body, use = literal)
	SetProgramOptions type *SetProgramOptions* PCacthUp = Whether or not programs should catch up upon startup PGracePeriod = The amount of time (in seconds) after which a schedule is still considered met HTMLRole = The role for HTML user ▪ PCatchUp type *boolean* ▪ PGracePeriod type *int* ▪ HTMLRole type *HTMLRoleFlag* - type *integer* with restriction - enum { '1', '2', '3', '4' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.53 SetSMTPConfig

Description	Sets SMTP Configuration Parameters
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetSMTPConfigRequest (soap:body, use = literal)
	SetSMTPConfig type *SetSMTPConfig* SMTP Configuration UseDefaultSMTP = Whether or not we should use UDI's default SMTP Server SMTPServer = SMTP Server SMTPPort = SMTP Port SMTPUID = SMTP User ID SMTPPWD = SMTP Password SMTPFrom = SMTP From email address SMTPTimeout = SMTP Timeout to wait for each operation UseTLS = Whether or not we should use TLS ▪ UseDefaultSMTP - optional; type *boolean* ▪ SMTPServer type *string* ▪ SMTPPort type *int* ▪ SMTPUID - optional; type *string* ▪ SMTPPWD - optional; type *string* ▪ SMTPFrom - optional; type *string* ▪ SMTPTimeout - optional; type *int* ▪ UseTLS - optional; type *Boolean*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.54 SetSceneProfile

Description	Sets the profile attribute for responders. Set the controller element to scene's address and: In case this is an ISY scene, set the node element to "none" In case this is a controller for a scene, set the node element to the responder's address
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetSceneProfileRequest (soap:body, use = literal)
	SetSceneProfile type *SetSceneProfile* • node type *string* • controller type *string* • control type *ControlType* • action type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.55 SetSystemDateTime

Description	Returns the current state of ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetDateTimeRequest (soap:body, use = literal)
	SetSystemTime type *SetSystemTime* ISY's DateTime Structure: NTP = time in NTP format TMZOffset = Time zone Offset from GMT (in seconds) DST = Daylight Saving Time (true/false) Lat = Latitude (in degrees) Long = Longitude (in degrees) Sunrise = Sunrise time in NTP format (response only) Sunset = Sunset time in NTP format (response only) IsMilitary = Whether or not we should use 24 hour format ▪ NTP type *long* ▪ TMZOffset type *int* ▪ DST type *boolean* ▪ Lat type *float* ▪ Long type *float* ▪ Sunrise - optional; type *long* ▪ Sunset - optional; type *long* ▪ IsMilitary - optional; type *boolean*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.56 SetUserCredentials

Description	Changes the userid and password for a user
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetUserCredentialsRequest (soap:body, use = literal)
	SetUserCredentials type *SetUserCredentials* ▪ name type *string* ▪ password type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.57 SetVariable

Description	Sets a variable to the desired value
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetVariableRequest (soap:body, use = literal)
	SetVariable type *SetVariable* Parameter for setting a variable to the desired value ▪ type type *UDIVariableType* - type *string* with restriction - enum { '1', '2' } ▪ id type *string* ▪ val type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.58 StartNodesDiscovery

Description	Puts ISY in discovery (linking) mode Optionally, provide the type of device to look for
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	StartNodesDiscoveryRequest (soap:body, use = literal)
	DiscoverNodes type *DiscoverNodes* Optionally, provide the type of node to be discovered ▪ ntype - optional; type *string* Whether or not ▪ flag - optional; type *byte* None ▪ family - optional; type *string* Optional family ... from family.xsd.
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.59 StopNodesDiscovery

Description	The flag decides the operations (reset, crawl, spider) to be performed after device(s) are discovered
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	StopNodesDiscoveryRequest (soap:body, use = literal)
	CancelNodesDiscovery type *CancelNodesDiscovery* ▪ flag type *NodeOperationsFlag* - type *byte* with restriction - enum { '1', '2', '3', '4' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.60 Subscribe

Description	reportURL: To reuse the socket, use REUSE_SOCKET as the value duration: use infinite
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SubscribeRequest (soap:body, use = literal)
	Subscribe type *Subscribe* ▪ reportURL type *string* ▪ duration type *string* ▪ SID - optional; type *string*
Output	SubscribeResponse (soap:body, use = literal)
	SubscriptionResponse type *Subscription* ▪ SID type *string* ▪ duration type *string*

17.2.61 SynchWithNTS

Description	Adjusts the clock to NTS
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SynchWithNTSRequest (soap:body, use = literal)
	SynchWithNTS type *SynchWithNTS*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.62 UDIService

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	UDIServiceRequest (soap:body, use = literal)
	UDIService type *UDIService* In the case of X10: 1. User "X10" as the control 2. action may be null 3. use house code and unit code in node. The format is: HouseCodeUnitCode (UnitCode may be null) ▪ control type *ControlType* ▪ action - optional; type *string* ▪ node type *string* ▪ flag type *NodeTypeFlag* - type *byte* with restriction - enum { '1', '4', '8' }
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.63 Unsubscribe

Description	Unsubscribe from ISY
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	UnsubscribeRequest (soap:body, use = literal)
	Unsubscribe type *Unsubscribe* ▪ SID type *string* ▪ flag - optional; type *int* flag is optional: 0 (or if not supplied): Return error if not currently subscribed 1 : Do not return an error if not currently subscribed
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.2.64 WriteDeviceUpdates

Description	Causes device updates to be written to the given node
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	WriteDeviceUpdatesRequest (soap:body, use = literal)
	WriteDeviceUpdates type *WriteDeviceUpdates* ▪ node type *string*
Output	DefaultResponseMessage (soap:body, use = literal)
	UDIDefaultResponse type *UDIDefaultResponse* ▪ status type *string* ▪ info - optional; type *string*

17.3 WSDL ELK Core Web Services Documentation

Target Namespace: http://www.universal-devices.com/wsdk/isy/elk/1.0

Port UDIELKWebServices_Port Port typeSource code

Location: http://192.168.0.195:8080/services/elk

Protocol: SOAP

Default style: document

Transport protocol: SOAP over HTTP

17.3.1 ArmArea

Description	Arms a security area
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ArmAreaRequest (soap:body, use = literal)
	ArmArea type *ArmAreaElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.2 AudioCmd

Description	Sends a command to an attached A/V equipment.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	AudioCmdRequest (soap:body, use = literal)
	AudioCmd type *AudioCmdElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.3 BypassArea

Description	Bypasses an area
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	BypassAreaRequest (soap:body, use = literal)
	BypassArea type *BypassAreaElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.4 BypassZoneToggle

Description	Toggles the bypass status of a zone
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	BypassZoneToggleRequest (soap:body, use = literal)
	BypassZoneToggle type *BypassZoneToggleElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.5 DisarmArea

Description	Disarms a security area
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	DisarmAreaRequest (soap:body, use = literal)
	DisarmArea type *DisarmAreaElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.6 DisplayTextOnAreaKeypads

Description	Displays the given text on an area keypad.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	DisplayTextOnAreaKeypadsRequest (soap:body, use = literal)
	DisplayTextOnAreaKeypads type *DisplayTextOnAreaKeypadsElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.7 GetAllStatus

Description	Retrieves all system settings. Returns results synchronously
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetAllStatusRequest (soap:body, use = literal)
	GetAllStatus type *GetAllStatusElement*
Output	GetAllStatusResponse (soap:body, use = literal)
	status type *status*

17.3.8 GetAreaStatus

Description	Retrieves the status for an area or all areas. Returns results synchronously.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetAreaStatusRequest (soap:body, use = literal)
	GetAreaStatus type *GetAreaStatusElement*
Output	AreaStatusResponse (soap:body, use = literal)
	status type *AreaResponseTypeElement*

17.3.9 GetConfig

Description	Retrieves ELK/ISY configuration parameters.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetConfigRequest (soap:body, use = literal)
	GetConfig type *GetConfigElement*
Output	GetConfigResponse (soap:body, use = literal)
	ELK-Config type *ELK-Config*

17.3.10 GetConnected

Description	Returns whether or not ISY/ELK are connected to one another.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetConnectedRequest (soap:body, use = literal)
	GetConnected type *GetConnectedElement*
Output	SystemStatusResponse (soap:body, use = literal)
	status type *SystemResponseElement*

17.3.11 GetEnabled

Description	Returns whether or not ELK Module is enabled.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetEnabledRequest (soap:body, use = literal)
	GetEnabled type *GetEnabledElement*
Output	SystemStatusResponse (soap:body, use = literal)
	status type *SystemResponseElement*

17.3.12 GetKeypadStatus

Description	Returns the status of a keypad or all keypads. Results returned synchronously.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetKeypadStatusRequest (soap:body, use = literal)
	GetKeypadStatus type *GetKeypadStatusElement*
Output	KeypadStatusResponse (soap:body, use = literal)
	status type *KeypadResponseElement*

17.3.13 GetOutputStatus

Description	Returns the status of an output or all outputs. Results returned synchronously.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetOutputStatusRequest (soap:body, use = literal)
	GetOutputStatus type *GetOutputStatusElement*
Output	OutputStatusResponse (soap:body, use = literal)
	status type *OutputResponseElement*

17.3.14 GetSystemStatus

Description	Retrieves the system status. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetSystemStatusRequest (soap:body, use = literal)
	GetSystemStatus type *GetSystemStatusElement*
Output	SystemStatusResponse (soap:body, use = literal)
	status type *SystemResponseElement*

17.3.15 GetThermostatStatus

Description	Returns the status of a thermostat or all thermostats. Results returned synchronously.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetThermostatStatusRequest (soap:body, use = literal)
	GetThermostatStatus type *GetThermostatStatusElement*
Output	ThermostatStatusResponse (soap:body, use = literal)
	status type *ThermostatResponseElement*

17.3.16 GetTopology

Description	Retrieves ELK configuration. Returns result synchronously.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetTopologyRequest (soap:body, use = literal)
	GetTopology type *GetTopologyElement*
Output	GetTopologyResponse (soap:body, use = literal)
	topology type *topology*

17.3.17 GetZoneStatus

Description	Returns the status of a given zone or all zones. Results are returned synchronously.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	GetZoneStatusRequest (soap:body, use = literal)
	GetZoneStatus type *GetZoneStatusElement*
Output	ZoneStatusResponse (soap:body, use = literal)
	status type *ZoneResponseElement*

17.3.18 KeypadPressFuncKey

Description	Simulates pressing a function key on a keypad.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	KeypadPressFuncKeyRequest (soap:body, use = literal)
	KeypadPressFuncKey type *KeypadPressFuncKeyElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.19 QueryAll

Description	Queries all system settings. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryAllRequest (soap:body, use = literal)
	QueryAll type *QueryAllElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.20 QueryAllZoneStatus

Description	Queries the status of all zones. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryAllZoneStatusRequest (soap:body, use = literal)
	QueryAllZoneStatus type *QueryAllZoneStatusElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.21 QueryAreaArmStatus

Description	Queries the Arm status of an area. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryAreaArmStatusRequest (soap:body, use = literal)
	QueryAreaArmStatus type *QueryAreaArmStatusElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.22 QueryKeypadTemperature

Description	Queries the temperature sensor on a keypad. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryKeypadTemperatureRequest (soap:body, use = literal)
	QueryKeypadTemperature type *QueryKeypadTemperatureElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.23 QueryOutputs

Description	Queries an output or all outputs. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryOutputsRequest (soap:body, use = literal)
	QueryOutputs type *QueryOutputsElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.24 QueryThermostat

Description	Queries the status of a thermostat or all thermostats. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryThermostatRequest (soap:body, use = literal)
	QueryThermostat type *QueryThermostatElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.25 QueryZoneStatus

Description	Queries the status of a given zone or all zones. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryZoneStatusRequest (soap:body, use = literal)
	QueryZoneStatus type *QueryZoneStatusElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.26 QueryZoneTemperature

Description	Queries the thermostats in a zone or all zones. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryZoneTemperatureRequest (soap:body, use = literal)
	QueryZoneTemperature type *QueryZoneTemperatureElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.27 QueryZoneVoltage

Description	Queries the voltages in a zone or all zones. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	QueryZoneVoltageRequest (soap:body, use = literal)
	QueryZoneVoltage type *QueryZoneVoltageElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.28 RefreshTopology

Description	Queries ELK configuration. Results are published as events.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	RefreshTopologyRequest (soap:body, use = literal)
	RefreshTopology type *RefreshTopologyElement*
Output	GetTopologyResponse (soap:body, use = literal)
	topology type *topology*

17.3.29 SetOutputOff

Description	Turns off an output.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetOutputOffRequest (soap:body, use = literal)
	SetOutputOff type *SetOutputOffElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.30 SetOutputOn

Description	Turns on an output.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetOutputOnRequest (soap:body, use = literal)
	SetOutputOn type *SetOutputOnElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.31 SpeakPhrase

Description	Causes the system to speak/vocalize the given phrase.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SpeakPhraseRequest (soap:body, use = literal)
	SpeakPhrase type *SpeakPhraseElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.32 SpeakWord

Description	Causes the system to speak/vocalize the given word.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SpeakWordRequest (soap:body, use = literal)
	SpeakWord type *SpeakWordElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.33 ThermostatCmd

Description	Sends a command to a thermostat.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ThermostatCmdRequest (soap:body, use = literal)
	ThermostatCmd type *ThermostatCmdElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.34 TriggerZone

Description	Momentarily triggers a zone to the physical state of Open. An error will occur if the zone is defined as normally open, or is currently open.
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	TriggerZoneRequest (soap:body, use = literal)
	TriggerZone type *TriggerZoneElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.3.35 UnbypassArea

Description	Releases the bypass of an area
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	UnbypassAreaRequest (soap:body, use = literal)
	UnbypassArea type *UnbypassAreaElement*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4 WSDL SEP/SmartGrid Core Web Services Documentation

Copyright 2007/2012 Universal Devices, Inc. All Rights Reserved Web Services Flag, Objects, Parameters, Messages, and Bindings for ISY WebService Port/Binding for ISY

Copyright 2007-2012 Universal Devices, Inc. All Rights Reserved Web Services Flag, Objects, Parameters, Messages, and Bindings for SEP Services

Target Namespace: http://www.universal-devices.com/wsdk/isy/sep/1.0

Port UDISEPServices_Port Port typeSource code

Location: http://192.168.0.195:8080/sepServices

Protocol: SOAP

Default style: document

Transport protocol: SOAP over HTTP

17.4.1 SEPCancelAllDREvents

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	EPCancelAllDREventsRequest (soap:body, use = literal)
	SEPCancelAllDREvents type *SEPCancelAllDREvents*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.2 SEPCancelAllMessageEvents

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPCancelAllMessageEventsRequest (soap:body, use = literal)
	SEPCancelAllMessageEvents type *SEPCancelAllMessageEvents*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.3 SEPCancelAllPriceEvents

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPCancelAllPriceEventsRequest (soap:body, use = literal)
	SEPCancelAllPriceEvents type *SEPCancelAllPriceEvents*
Output	DefaultResponseMessage (soap:body, use = literal)
	esponse type *UDIDefaultRespnse*

17.4.4 SEPConfirmMessage

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPConfirmMessageRequest (soap:body, use = literal)
	SEPConfirmMessage type *SEPConfirmMessage*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.5 SEPDROpt

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPDROptRequest (soap:body, use = literal)
	SEPDROpt type *SEPDROpt*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.6 SEPStartDREvent

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPStartDREventRequest (soap:body, use = literal)
	SEPStartDREvent type *SEPStartDREvent*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.7 SEPStartMessage

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPStartMessageRequest (soap:body, use = literal)
	SEPStartMessage type *SEPStartMessage*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.8 SEPStartPrice

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPStartPriceRequest (soap:body, use = literal)
	SEPStartPrice type *SEPStartPrice*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.9 SEPStopDREvent

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPStopDREventRequest (soap:body, use = literal)
	SEPStopDREvent type *SEPStopDREvent*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.10 SEPStopMessage

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPStopMessageRequest (soap:body, use = literal)
	SEPStopMessage type *SEPStopMessage*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.4.11 SEPStopPrice

Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SEPStopPriceRequest (soap:body, use = literal)
	SEPStopPrie type *SEPStopPrice*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5 WSDL Z-Wave Core Web Services Documentation

Target Namespace: http://www.universal-devices.com/wsdk/isy/zwave/1.0

Port UDIZWaveWebServices_Port Port typeSource code

Location: http://192.168.0.195:8080/zwaveServices

Protocol: unknown

17.5.1 ExcludeDevice

Description	Remove a device from the Z-Wave network
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ExcludeDeviceRequest (soap:body, use = literal)
	ExcludeDevice type *ExcludeDevice*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.2 FactoryResetDongle

Description	Factory reset the Z-Wave dongle force : [true|false] : True=Force factory reset, False=Factory reset if dongle is not part of existing Z-Wave network
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	FactoryResetDongleRequest (soap:body, use = literal)
	FactoryResetDongle type *FactoryResetDongle*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.3 IncludeDevice

Description	Add a device into the Z-Wave network power : [true|false] : True=High Power, False=Normal Power (default) nwi : [true|false] : True=Network wide inclusion, False=Standard Inclusion
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	IncludeDeviceRequest (soap:body, use = literal)
	IncludeDevice type *IncludeDevice*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.4 ReplicateSendPrimary

Description	As Primary Controller, replicate to another controller and make it the new Primary Controller
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	ReplicateSendPrimaryRequest (soap:body, use = literal)
	ReplicateSendPrimary type *ReplicateSendPrimary*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.5 SetActiveAntenna

Description	Set Active Antenna 0 : Internal Antenna 1 : External Antenna
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SetActiveAntennaRequest (soap:body, use = literal)
	SetActiveAntenna type *SetActiveAntenna*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.6 StartLearnMode

Description	Go into Z-Wave learn mode (to replicate, be added/removed from Z-Wave network, etc.)
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	StartLearnModeRequest (soap:body, use = literal)
	StartLearnMode type *StartLearnMode*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.7 StopIncludeExclude

Description	Cancel include/exclude/replication
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	StopIncludeExcludeRequest (soap:body, use = literal)
	StopIncludeExclude type *StopIncludeExclude*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.8 Sync

Description	Synchronize ISY with info on Z-Wave dongle for the given node, specify either id or uid. If neither is specified then sync all new and deleted nodes. id : The ISY node address uid : The Z-Wave unit ID of the device
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SyncRequest (soap:body, use = literal)
	Sync type *Sync*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

17.5.9 SyncFull

Description	Synchronize ISY with info on Z-Wave dongle for the given node, specify either id or uid. If neither is specified then sync all nodes. id : The ISY node address uid : The Z-Wave unit ID of the device
Style	Document
Operation type	*Request-response.* The endpoint receives a message, and sends a correlated message.
Input	SyncFullRequest (soap:body, use = literal)
	SyncFull type *SyncFull*
Output	DefaultResponseMessage (soap:body, use = literal)
	response type *UDIDefaultRespnse*

18 Table of Figures

Figure 1: Open Polyglot 20
Figure 2: Login to Polyglot 20
Figure 3: Register New NodeServer 21
Figure 4: Delete NodeServer 22
Figure 5: Delete NodeServer Disconnected 22
Figure 6: Polyglot Real-time Log File 23
Figure 7: NodeServer Connected 24
Figure 8: Node Details 25
Figure 9: NodeServer Real-time Log File 26
Figure 10: Raspian Stretch Lite Download 29
Figure 11: Etcher Program 30
Figure 12: BOOT Drive with SSH File Added 31
Figure 13: SecureCRT SSH Raspberry PI Session 32
Figure 14: Polyglot Script Completion 33
Figure 15: Polyglot Installation Status 34
Figure 16: Polyglot Login Screen 34
Figure 17: Polyglot Settings Status 35
Figure 18: Dashboard Screen – No NodeServers 36
Figure 19: Dashboard Screen – With ISY Portal NodeServer 36
Figure 20: NodeServer Store Screen 37
Figure 21: Register New NodeServer Screen 38
Figure 22: NodeServer Status 39
Figure 23: NodeServer Shown in ISY Network Tree 39
Figure 24: Polyglot Settings 46
Figure 25: Add Node Server 47
Figure 26: Managing Node Servers – Phillips Hue 48
Figure 27: New Java App 129
Figure 28: Name Your Project 130
Figure 29: Add Project Library 131
Figure 30: Template Class 132
Figure 31: New Package 137
Figure 32: New Parsed Services 138
Figure 33: Insert Code Web Services 140
Figure 34: TCPMon 144
Figure 35: Class Diagram 147
Figure 36: X10 Commands 178
Figure 37: Setting up Zigbee Network 389
Figure 38: ECM 1240 Nodes 391
Figure 39: UDI EM3 Nodes 393
Figure 40: Setting up Zigbee Network 401

19 Bibliography

Gutwin Tom https://wiki.universal-devices.com/index.php?title=ISY_Developers:Java_ISY_REST_Requester_Example [Online] // https://wiki.universal-devices.com.
James Milne https://github.com/Einstein42/udi-polyglotv2/wiki/Creating-a-NodeServer [Online] // https://github.com.
Universal Devices Forums [Online] // Universal Devices Forum. - https://forum.universal-devices.com/.
Universal Devices http://www.universal-devices.com/developers/wsdk/5.0.0/ISY-WS-SDK-Node-Server.pdf [Online] // http://www.universal-devices.com.
Universal Devices https://github.com/UniversalDevicesInc/Polyglot [Report].
Universal Devices https://ud-polyglot.readthedocs.io/en/development/ [Online] // https://ud-polyglot.readthedocs.io.
Universal Devices https://wiki.universal-devices.com/index.php?title=ISY_Developers:API:REST_Interface [Online] // https://wiki.universal-devices.com.
Universal Devices https://wiki.universal-devices.com/index.php?title=ISY_Developers:Java_Web_Services_Tutorial [Online] // https://wiki.universal-devices.com.
Universal Devices https://wiki.universal-devices.com/index.php?title=Node_Hints_Documentation [Online] // https://wiki.universal-devices.com.
Universal Devices ISY-99i & 994i Series User Guide [Book]. - [s.l.] : Universal Devices. - Vols. https://www.universal-devices.com/docs/production/ISY%20User%20Guide%20v3.3.10%20a2.pdf.
Universal Devices Main Page [Online] // Universal Devices Forum. - https://wiki.universal-devices.com/index.php?title=Main_Page.