An Investigation of Parameter Relationships in a High-Speed Digital Multimedia Environment

Submitted in fulfilment
of the requirements of the degree
Doctor of Philosophy
of Rhodes University

Nyasha Chigwamba

December 2013

Abstract

With the rapid adoption of multimedia network technologies, a number of companies and standards bodies are introducing technologies that enhance user experience in networked multimedia environments. These technologies focus on device discovery, connection management, control, and monitoring. This study focused on control and monitoring. Multimedia networks make it possible for devices that are part of the same network to reside in different physical locations. These devices contain parameters that are used to control particular features, such as speaker volume, bass, amplifier gain, and video resolution. It is often necessary for changes in one parameter to affect other parameters, such as a synchronised change between volume and bass parameters, or collective control of multiple parameters. Thus, relationships are required between the parameters. In addition, some devices contain parameters, such as voltage, temperature, and audio level, that require constant monitoring to enable corrective action when thresholds are exceeded. Therefore, a mechanism for monitoring networked devices is required. This thesis proposes relationships that are essential for the proper functioning of a multimedia network and that should, therefore, be incorporated in standard form into a protocol, such that all devices can depend on them. Implementation mechanisms for these relationships were created.

Parameter grouping and monitoring capabilities within mixing console implementations and existing control protocols were reviewed. A number of requirements for parameter grouping and monitoring were derived from this review. These requirements include a formal classification of relationship types, the ability to create relationships between parameters with different underlying value units, the ability to create relationships between parameters residing on different devices on a network, and the use of an event-driven mechanism for parameter monitoring. These requirements were the criteria used to govern the implementation mechanisms that were created as part of this study.

Parameter grouping and monitoring mechanisms were implemented for the XFN protocol. The mechanisms implemented fulfil the requirements derived from the review of capabilities of mixing consoles and existing control protocols. The formal classification of relationship types was implemented within XFN parameters using lists that keep track of the relationships between each XFN parameter and other XFN parameters that reside on the same device or on other devices on the network. A common value unit, known as the global unit, was defined for use as the value

format within value update messages between XFN parameters that have relationships. Mapping tables were used to translate the global unit values to application-specific (universal) units, such as decibels (dB). A mechanism for bulk parameter retrieval within the XFN protocol was augmented to produce an event-driven mechanism for parameter monitoring. These implementation mechanisms were applied to an XFN-protocol-compliant graphical control application to demonstrate their usage within an end user context.

At the time of this study, the XFN protocol was undergoing standardisation within the Audio Engineering Society. The AES-64 standard has now been approved. Most of the implementation mechanisms resulting from this study have been incorporated into this standard.

Acknowledgements

It is with immense gratitude that I acknowledge support and guidance from my supervisor, Professor Richard Foss, without whom this work would not have been possible.

I am grateful to Universal Media Access Networks (UMAN) GmbH for providing equipment and source code that was invaluable to this research. Robby Gurdan and Bradley Klinkradt deserve special mention for helping me generate ideas for solutions to various problems. In addition, I thank Melekam Tsegaye for the preliminary work done in Unos Creator. Thank you to Harold Okai-Tettey for encouraging me throughout my research.

Thank you to Philip Nye for sharing information regarding the Architecture for Control Networks (ACN) at the onset of my research.

Many thanks to my colleagues in the Audio Engineering Research Group at Rhodes University for sharing information and material that was relevant to this research.

This research was undertaken within the Telkom Centre of Excellence in Distributed Multimedia at Rhodes University.

I acknowledge financial support received from Rhodes University and The Beit Trust.

Lastly, I am grateful to my wife, parents, siblings, and friends for their continued support and encouragement throughout my research.

Contents

1 Introduction **1**

1.1 Problem Statement 3

1.2 Aims and Objectives 5

1.3 Document Structure 6

2 Parameter Relationships in Mixing Consoles **8**

2.1 Mixing Consoles 8

2.1.1 Control Grouping 9

2.1.1.1 VCA Grouping 10

2.1.1.1.1 VCA Faders 11

2.1.2 Audio Grouping 13

2.2 Manufacturer Specific Implementations of Mixing Consoles with Particular Reference to Control Grouping 15

2.2.1 Yamaha Corporation 16

2.2.1.1 01V96 Version 2 Digital Mixing Console 16

2.2.1.1.1 Input/Output Channel Pairing 16

2.2.1.1.2 Fader Groups 17
2.2.1.1.2.1 Standard Fader Groups 17
2.2.1.1.2.2 Master Fader Groups 19
2.2.1.1.3 Mute Groups 19
2.2.1.1.3.1 Standard Mute Groups 20
2.2.1.1.3.2 Master Mute Groups 20
2.2.1.1.4 Linking EQ and Compressor Parameters 21
2.2.2 Allen & Heath Limited 21
2.2.2.1 ML3000, ML4000, and ML5000 Live Sound Consoles 21
2.2.2.1.1 VCA Groups 22
2.2.2.1.1.1 VCA Group Examples 22
2.2.2.1.2 Mute Groups 26
2.2.3 D&R Electronica Weesp b.v. 27
2.2.3.1 PowerVCA 27
2.2.3.2 Sirius 27
2.3 Chapter Summary 28

3 Existing Multimedia Network Technologies 30

3.1 Open Sound Control (OSC) Protocol 31
3.1.1 Protocol Overview 31
3.1.2 Parameter Grouping Capabilities within the OSC Protocol 33
3.1.2.1 Alternative 1: Grouping Managed by a Controller 34

3.1.2.2 Alternative 2: Grouping Managed by a Device 36
3.1.3 Parameter Monitoring Capabilities within the OSC Protocol 38
3.2 IEC 62379 38
3.2.1 SNMP Overview 39
3.2.2 IEC 62379 Architecture 40
3.2.2.1 Block Framework 40
3.2.3 Parameter Grouping Capabilities within IEC 62379 42
3.2.3.1 Extensions for Parameter Grouping Relationships within IEC 62379 42
3.2.4 Event-Driven Parameter Monitoring Capabilities within IEC 62379 . . . 44
3.3 Audio Video Control (AV/C) Protocol 46
3.3.1 Function Control Protocol (FCP) 47
3.3.2 AV/C Protocol Overview 48
3.3.2.1 AV/C Model 48
3.3.2.2 AV/C Commands and Responses 50
3.3.2.3 Parameter Grouping Capabilities within the AV/C Protocol . . 51
3.3.2.4 Polling and Semi-Event-Driven Parameter Monitoring Capabilities within the AV/C Protocol 52
3.4 Architecture for Control Networks (ACN) 54
3.4.1 Parameter Grouping Capabilities within ACN 55
3.4.1.1 Master/Slave or Unidirectional Bindings 56
3.4.1.2 Bidirectional Bindings 57

3.4.1.3 Multiway Bindings 57
3.4.2 Event-Driven Parameter Monitoring Capabilities within ACN 58
3.5 Open Control Architecture (OCA) 59
3.5.1 OCA Device Model 61
3.5.2 Parameter Grouping Capabilities within OCA 62
3.5.3 Event-Driven Parameter Monitoring Capabilities within OCA 64
3.6 Chapter Summary 66

4 Requirements for Parameter Relationships in Multimedia Networks 68

4.1 Classification of Core Parameter Grouping Relationships 68
4.1.1 Relationship Types 69
4.1.1.1 Peer-to-Peer Groups 69
4.1.1.2 Master-Slave Groups 70
4.1.2 Relationship Sub-Types 70
4.1.2.1 Relative Relationships 71
4.1.2.1.1 Relative Peer-to-Peer Relationships 71
4.1.2.1.2 Relative Master-Slave Relationships 72
4.1.2.2 Absolute Relationships 73
4.1.2.2.1 Absolute Peer-to-Peer Relationships 73
4.1.2.2.2 Absolute Master-Slave Relationships 74
4.2 Relationships between Parameters on Different Devices 75
4.3 Relationships between Parameters with Unrelated Units 75
4.4 Requirements for Parameter Monitoring 76
4.5 Chapter Summary 77

5 Implementation Strategies to Fulfil Grouping Requirements 78

5.1 XFN Protocol Overview 79

5.2 Core Parameter Grouping Relationships 81

5.3 Managing XFN Parameter Group Lists 84

5.3.1 Creating Peer-to-Peer Parameter Relationships 84

5.3.2 Breaking Peer-to-Peer Parameter Relationships 89

5.3.3 Creating Master-Slave Parameter Relationships 90

5.3.4 Breaking Master-Slave Parameter Relationships 92

5.4 Handling XFN Parameter Value Updates 92

5.4.1 Handling Parameter Value Changes 92

5.4.2 Handling of Non-Additive Relationships 94

5.4.3 Common Value Unit 95

5.4.3.1 Global Unit Value Type Description 96

5.4.3.2 Global Unit Value Manipulation by Various Controllers 99

5.4.3.3 Global Unit Value to Application-Specific Unit Mapping . . . 100

5.4.3.3.1 Linear Global Unit Mappings 100

5.4.3.3.2 Non-Linear Global Unit Mappings 101

5.4.3.3.3 Considerations for Standardising the Mapping of Global Units to Universal Units 102

5.5 Parameter Grouping Rules 104

5.5.1 Hybridisation of Master-Slave and Peer-to-Peer Groups 105

5.5.1.1 Scenario 105

5.5.1.2 Rules Defined . . . 106
5.5.1.2.1 Rule 1: Master Becomes Master of all Peers . . . 107
5.5.1.2.2 Rule 2: Master Removed from all Peers . . . 108
5.5.1.2.3 Rule 3: Masters Merged for all Peers . . . 108
5.5.2 Inferring Relationships Types . . . 110
5.5.2.1 Scenario . . . 110
5.5.2.2 Rules Defined . . . 111
5.5.3 Retention of Permanent Relationships . . . 114
5.5.3.1 Scenario . . . 114
5.5.3.2 Rules Defined . . . 116
5.6 Future Work . . . 119
5.6.1 Cyclic Master-Slave Relationships . . . 119
5.6.1.1 Possible Solution using Graph Theory . . . 122
5.6.2 Usage of the Global Unit . . . 124
5.7 Chapter Summary . . . 124

6 Implementation Strategies for Parameter Monitoring 126

6.1 The USG Mechanism . . . 127
6.1.1 USG Mechanism Implementation Details . . . 127
6.1.1.1 Initialisation . . . 129
6.1.1.2 Parameter Bitmap Creation . . . 131
6.1.1.3 Subset Parameter Cache List Retrieval . . . 131

6.1.1.4 Global Unit Type Bitmap Creation 132

6.1.1.5 Retrieval of Global Unit Values and Indexes into Other-Unit Value Table 133

6.1.1.6 Retrieval of Other-Unit Value Table 134

6.1.1.7 USG Buffer Deallocation 135

6.2 Push Mechanism 135

6.2.1 Overview 137

6.2.2 Push Implementation Details 138

6.2.2.1 Device Initialisation 139

6.2.2.2 Controller Initialisation 139

6.2.2.2.1 Associating Controls with XFN Parameters 141

6.2.2.2.2 Learning the Monitored-Parameter List Layout . . . 142

6.2.2.2.3 Handler Configuration 143

6.2.2.3 Subscriber Registration 146

6.2.2.4 Value Change Events 148

6.3 Fulfilment of Requirements for Parameter Monitoring 150

6.4 Future Work 151

6.4.1 Dynamic Determination of Monitored-Parameter Lists 151

6.5 Chapter Summary 154

7 An XFN Control Application Example 156

7.1 Overview of Unos Creator 157

7.2 Deskitems 159

7.2.1 Deskitem Properties 161

7.2.1.1 Common Deskitem Properties 161

7.2.1.2 Background Deskitem Properties 163

7.2.1.3 Fader Deskitem Properties 164

7.2.1.4 Pot Deskitem Properties 167

7.2.1.5 Meter Deskitem Properties 168

7.2.1.6 Multi-I/O Deskitem Properties 171

7.2.1.7 Display Deskitem Properties 174

7.2.2 A GUI Editor for Deskitems 177

7.2.2.1 Procedure for Creating New Deskitems 178

7.3 Managing Grouping Relationships 185

7.3.1 Peer-to-Peer Relationships 187

7.3.1.1 Example of Creating Peer-to-Peer Relationships 188

7.3.1.2 Example of Breaking Peer-to-Peer Relationships 190

7.3.2 Master-Slave Relationships 193

7.3.2.1 Example of Creating Master-Slave Relationships 194

7.3.2.2 Example of Breaking Master-Slave Relationships 196

7.3.3 Creating Relationships Between Parameters on Different Devices 199

7.4 Parameter Monitoring 203

7.5 Test Environment 203

7.5.1 Deskitem Control Panel 206

7.5.2 Global Unit Value Mapping Tables 207

7.5.2.1 Gain Coefficients 207

7.5.2.2 Peak Values 208

7.5.3 Nature of Tests Carried Out 208

7.6 Comparison of Network Parameter Grouping Mechanisms with Mixing Console Capabilities 210

7.6.1 Linking of Controls 212

7.6.2 Fader Groups 212

7.6.3 Mute Groups 213

7.6.4 Grouping Restrictions 213

7.7 Future Work 213

7.8 Chapter Summary 214

8 Conclusion **215**

8.1 Core Requirements for Parameter Grouping and Monitoring in Multimedia Networks 216

8.1.1 Requirement 1: Utilisation of a Consistent Classification Scheme for Parameter Grouping Relationships 217

8.1.2 Requirement 2: A Standardised Mechanism for Relationships between Parameters on Physically Distant Devices 218

8.1.3 Requirement 3: A Capability for Relationships between Parameters with Different Underlying Units . . . 218

8.1.4 Requirement 4: Event-Driven Parameter Monitoring . . . 218

8.2 Implementation Strategies to Fulfil Parameter Grouping Requirements . . . 219

8.2.1 Management of Relationships between XFN Parameters . . . 219

8.2.2 Common Value Unit . . . 220

8.2.3 Rules Defined for Parameter Relationships . . . 220

8.2.3.1 Hybridisation of Master-Slave and Peer-to-Peer Relationships . 221

8.2.3.2 Inference of Implicit Relationships . . . 221

8.2.3.3 Persistence of Critical Relationships . . . 222

8.3 Implementation Strategies to Fulfil Parameter Monitoring Requirements . . . 222

8.4 An Example Control and Monitoring Application . . . 223

8.5 Future Work . . . 225

8.6 Significance of this Research . . . 226

References **228**

Glossary **232**

A Mixing Consoles **236**

A.1 Audio Grouping . . . 236

A.1.1 Master Groups . . . 236

A.1.2 In-Line Groups . . . 239

A.2 Procedures for Configuring Control Grouping Relationships in Manufacturer Specific Implementations of Mixing Consoles 241

A.2.1 Yamaha Corporation 241

A.2.1.1 01V96 Version 2 Digital Mixing Console 241

A.2.1.1.1 Procedures for Input/Output Channel Pairing 245

A.2.1.1.2 Procedures for Setting Up Fader Groups 249

A.2.1.1.2.1 Standard Fader Groups 249

A.2.1.1.2.2 Master Fader Groups 251

A.2.1.1.3 Procedures for Setting Up Mute Groups 253

A.2.1.1.3.1 Standard Mute Groups 253

A.2.1.1.3.2 Master Mute Groups 254

A.2.1.1.4 Procedure for Linking EQ and Compressor Parameters 255

A.2.2 Allen & Heath Limited 257

A.2.2.1 ML5000 Live Sound Console 257

A.2.2.1.1 VCA Groups 261

A.2.2.1.1.1 Procedures for Assigning and Clearing VCA Groups 262

A.2.2.1.2 Mute Groups 263

A.2.2.1.2.1 Procedures for Assigning and Clearing Mute Groups 264

A.2.3 D&R Electronica Weesp b.v. 266

A.2.3.1 PowerVCA 266

A.2.3.1.1 PowerVCA Grouping Procedures 268

A.2.3.2 Sirius 268

B Validation for Relationship Inference Rules **271**

B.1 Combination 1 Validation: Relative Peer-to-Peer and Absolute Peer-to-Peer . . . 271

B.2 Combination 2 Validation: Relative Peer-to-Peer and Relative Master-Slave . . . 273

B.3 Combination 3 Validation: Relative Peer-to-Peer and Absolute Master-Slave . . . 274

B.4 Combination 4 Validation: Relative Peer-to-Peer and Relative Peer-to-Peer 275

B.5 Combination 5 Validation: Absolute Peer-to-Peer and Relative Master-Slave . . . 276

B.6 Combination 6 Validation: Absolute Peer-to-Peer and Absolute Master-Slave . . 277

B.7 Combination 7 Validation: Absolute Peer-to-Peer and Absolute Peer-to-Peer . . . 278

List of Figures

2.1 A Basic Mixing Console with Two VCA Groups 10

2.2 Conventional Fader Circuitry 11

2.3 VCA Fader Circuitry 11

2.4 Single Fader Controlling Multiple Channel VCAs 12

2.5 VCA Fader Group Illustration 13

2.6 Compressor Ratios 14

2.7 Example of Compression Imbalance Requiring Stereo Linking 15

2.8 Standard Fader Group Example 18

2.9 VCA Group Example 1 23

2.10 VCA Group Example 2 23

2.11 VCA Group Example 3 26

3.1 Example of an OSC Address Space 32

3.2 Device Parameters Mapped to OSC Methods with a Single Controller 34

3.3 Device Parameters Mapped to OSC Methods with Two Controllers 36

3.4 OSC Address Space with Methods to Create and Delete Groups 37

3.5 Hierarchical Representation of IEC 62379 OIDs 39

3.6 Model of a Simple Audio Device 40

3.7 Hierarchical Layout of the Limiter Block Table 42

3.8 Limitations with Grouping at the Block Level 44

3.9 Hierarchical Layout of Audio Status Groups [International Electrotechnical Commission, 2008] 45

3.10 FCP Command and Response Registers as adapted from International Electrotechnical Commission [2003] 48

3.11 AV/C Unit Model 49

3.12 AV/C Subunit Model 49

3.13 AV/C Control Grouping 52

3.14 AV/C NOTIFY Command as adapted from 1394 Trade Association [2001] 53

3.15 Polling Alternative to the NOTIFY Command as Recommended by 1394 Trade Association [2001] 54

3.16 ACN Overview 55

3.17 ACN Subscriber-Event Mechanism 59

3.18 OCA Class Hierarchy Example 60

3.19 OCA Device Model: Supported Object Types 61

3.20 Grouper Example 63

3.21 Subscription Mechanism Example 65

4.1 Peer-to-Peer Group Example 69

4.2 Master-Slave Group Example 70

4.3 Relative Peer-to-Peer Relationships 71

4.4 Relative Master-Slave Relationships 72

4.5 Handling Slave Parameter Value Changes in Relative Master-Slave Relationships 73

4.6 Absolute Peer-to-Peer Relationships 74

4.7 Absolute Master-Slave Relationships 74

4.8 Example Monitoring Configuration 77

5.1 Typical XFN Device Layout 80

5.2 XFN Parameter Relationship Example 84

5.3 Creating a Peer-to-Peer Relationship: Initial State 85

5.4 Creating a Peer-to-Peer Relationship: After First List Merge 87

5.5 Creating a Peer-to-Peer Relationship: Final State 89

5.6 Creating a Peer-to-Peer Relationship: Summary of Steps 89

5.7 Breaking Peer-to-Peer Relationships: Final State 90

5.8 Creating a Master-Slave Relationship: Initial State 90

5.9 Creating a Master-Slave Relationship: Final State 91

5.10 XFN Parameter Value Updates 94

5.11 Value Modifier 95

5.12 Global Unit Value Range Overflow and Underflow 97

5.13 Global Unit Value Bits 98

5.14 Bitwise Global Unit Range Modification 99

5.15 Global Unit Value 8-bit Controller: Coarse Granularity 99

5.16 Global Unit Value 8-bit Controller: Finest Granularity 100

5.17 Cross-Vendor Parameter Relationships 103

5.18 Master to Slave in Peer Group 105

5.19 Master to Multiple Slaves in Peer Group 106

5.20 Creating a Master-Slave Relationship with a Peer Group 107

5.21 Breaking a Master-Slave Relationship with a Peer Group 108

5.22 Creating a Peer-to-Peer Relationship across Two Peer Groups 109

5.23 Addition of a New Parameter Relationship to Existing Peer Group 111

5.24 Master-Slave Relationship Chains without Inferred Relationships 113

5.25 Controller-Device Layout 115

5.26 Controller-Device Layout with Deskitem Joins 116

5.27 Controller-Device Layout with Permanent Absolute Peer-to-Peer Joins 117

5.28 Controller with Two Devices 118

5.29 Peer Group Lists: Controller with Two Devices 118

5.30 Peer Group Lists: Controller and Two Devices Including Extra Relationship . . . 119

5.31 XFN Parameter Value Updates 120

5.32 Cyclic Master-Slave Relationship Chains 121

5.33 Directed Acyclic Graph Illustration 122

6.2 USG Mechanism: Summary of Roles and Process 127

6.1 USG Mechanism Components 128

6.3 Parameter Bitmap 131

6.4 Global Unit Type Bitmap after Applying Parameter Bitmap 133

6.5 Device and Controller Roles 136

6.6 Overview of Push Mechanism 137

6.7 Example Meter Deskitem 140

6.8 Graphical Meters with XFN Parameter Associations 142

6.9 Device and Controller State after Handler Configuration 145

6.10 XFN Device Layout for Monitored Parameters 147

6.11 Periodic Push Processing 150

6.12 Overview of Push Mechanism 151

6.13 Multiple Controllers Monitoring Parameters on a Single Device 153

7.1 Unos Creator Application Overview 158

7.2 Media Player Control Surface 160

7.3 Procedure for Device-Specific Deskitem Extraction 161

7.4 Background Deskitem XML Element 164

7.6 Fader Deskitem XML Element 164

7.5 Annotated Fader Deskitem 166

7.7 Annotated Pot Deskitem 168

7.8 Pot Deskitem XML Element 168

7.9 Annotated Meter Deskitem 171

7.10 Meter Deskitem XML Element 171

7.11 Annotated Multi-I/O Deskitem 174

7.12 Multi-I/O Deskitem XML Element 174

7.13 Display Deskitem Overview 175

7.14 Display Deskitem XML Element 176

7.15 Unos Creator GUI Editor Tool 178

7.16 Adding a New Deskitem Page 179

7.17 Adding a Fader Deskitem 180

7.18 Uncustomised Fader Deskitem 180

7.19 Image Property Editor 181

7.20 Image File Browser 182

7.21 Fully Skinned Fader Deskitem 182

7.22 Deskitem Remote Parameter Information 183

7.23 Parameter Selection Window 184

7.24 Packaging Device Deskitems 185

7.25 Unos Creator - Fader Control Surface 187

7.26 Creating Relative Peer-to-Peer Relationships in Unos Creator 189

7.27 Peer-to-Peer Relationship State After Parameter Joining 190

7.28 Breaking Peer-to-Peer Relationships in Unos Creator 192

7.29 Peer-to-Peer Relationship State After Parameter Unjoining 193

7.30 Creating Relative Master-Slave Relationships in Unos Creator 195

7.31 Master-Slave Relationship State After Parameter Joining 196

7.32 Breaking Master-Slave Relationships in Unos Creator 198

7.33 Relationship State After Master-Slave Parameter Unjoining 199

7.34 Custom Control Surface in Unos Creator 201

7.35 Joining Parameters in Different Devices 202

7.36 Test Environment 204

7.37 DICE Mixer Matrix 205

7.38 DICE Mixer Matrix Deskitem 206

7.39 Integrated System - Mixing Console 211

7.40 Distributed System - Networked Media Devices and Control Application 211

A.1 A Basic Mixing Console with Audio Groups. 237

A.2 Signal Flow Diagram for Audio Groups. 238

A.3 Basic Schematic of Audio Groups 239

A.4 Signal Flow Diagram for In-line Groups. 240

A.5 01V96 Version 2 Digital Mixing Console 242

A.6 01V96 Mixing Console - Display Access Section 243

A.7 01V96 Mixing Console - Display Section 243

A.8 01V96 Mixing Console - Data Entry Section 244

A.9 01V96 Mixing Console - Partial Channel Strip Section 244

A.10 01V96 Mixing Console - Channel Pairing Window 246

A.11 01V96 Mixing Console - “Pair/Group | Input” Page 247

A.12 01V96 Mixing Console - “Pair/Group | Output” Page 249

A.13 01V96 Mixing Console - “Pair/Group | In Fader” Page 250

A.14 01V96 Mixing Console - “Pair/Group | In Fader” Page with Group Selected . . . 250

A.15 01V96 Mixing Console - “Pair/Group | In Fader” Page with Grouped Channels . 251

A.16 01V96 Mixing Console - “Pair/Group | In Fader” Page: Input Fader Master Selected 252

A.17 01V96 Mixing Console - “Pair/Group | In Master” Page 252

A.18 01V96 Mixing Console - “Pair/Group | In Mute” Page 254

A.19 01V96 Mixing Console - “Pair/Group | In Mute” Page: Input Mute Master Selected 255

A.20 01V96 Mixing Console - “Pair/Group | In EQ” Page 256

A.21 01V96 Mixing Console - “Pair/Group | In EQ” Page: Link Selection 256

A.22 01V96 Mixing Console - “Pair/Group | In EQ” Page: Link “b” Channels 257

A.23 ML5000 Mixing Console 258

A.24 ML5000 Mixing Console: Input Fader 259

A.25 ML5000 Mixing Console: Snapshot Memories 260

A.26 ML5000 Mixing Console: VCA Group Controls 261

A.27 ML5000 Mixing Console: Mute Group Controls 264

A.28 PowerVCA: Mix Screen [D&R Electronica Weesp b.v., 2010a] 266

A.29 PowerVCA: Channel Module [D&R Electronica Weesp b.v., 2010a] 267

A.30 Sirius Mixing Console Control Page [D&R Electronica Weesp b.v., 2010c] 269

A.31 Sirius Mixing Console Control Page: Group Buttons [D&R Electronica Weesp b.v., 2010b] 270

B.1 Combination 1: Relative Peer-to-Peer (P1-P2) and Absolute Peer-to-Peer (P1-P3) 272

B.2 Combination 2: Relative Peer-to-Peer (P1-P2) and Relative Master-Slave (P1-P3) 273

B.3 Combination 3: Relative Peer-to-Peer (P1-P2) and Absolute Master-Slave (P2-P3) 274

B.4 Combination 4: Relative Peer-to-Peer (P1-P2) and Relative Peer-to-Peer (P2-P3) 275

B.5 Combination 5: Absolute Peer-to-Peer (P1-P2) and Relative Master-Slave (P1-P3) 276

B.6 Combination 6: Absolute Peer-to-Peer (P1-P2) and Absolute Master-Slave (P1-P3) 277

B.7 Combination 7: Absolute Peer-to-Peer (P1-P3) and Absolute Peer-to-Peer (P2-P3) 278

List of Tables

3.1 Limiter Block Table Layout [International Electrotechnical Commission, 2008] . 41

3.2 Block Table with Group Identifier [Eales, 2012] 43

3.3 Relationship Table [Eales, 2012] 43

3.4 Relationship Type Table [Eales, 2012] 44

3.5 Status Entries for Audio Port Page [International Electrotechnical Commission, 2008] 45

3.6 AV/C Command Types as adapted from 1394 Trade Association [2001] 50

3.7 AV/C Response Types as adapted from 1394 Trade Association [2001] 51

3.8 Summary of Parameter Grouping and Monitoring Capabilities 66

5.1 XFN Parameter Group List Entry Fields 82

5.2 GET PTPGRP Command Response 86

5.3 SET MASTERS Command Layout 91

5.4 Global Unit Range Allocation 96

5.5 Example Linear dB-to-Global-Unit Mapping Table 101

5.6 Example Non-Linear dBu-to-Global-Unit Mapping Table 102

5.7 Vendor-Specific Global Unit to Decibel Mapping Tables 104
5.8 Permutations of Inference Rules 113
5.9 Combinations of Inference Rules Validated 114

6.1 USG Parameter Cache List 130
6.2 Subset USG Parameter Cache List 132
6.3 Subset USG Parameter Cache List Masked with Global Unit Type Bitmap 133
6.4 Subset USG Parameter Cache List with Values/Indexes 134
6.5 Other Unit Value Table 135
6.6 USG Buffer Learning Process Results 143
6.7 Controller-Parameter to Monitored-Parameter Mapping List 145
6.8 SET PUSH Command Structure 147
6.9 Monitored Parameter USG Buffer 153
6.10 Modified Monitored Parameter USG Buffer 154

7.1 Common Deskitem Properties 162
7.2 Background Deskitem Properties 164
7.3 Fader Deskitem Properties 165
7.4 Pot Deskitem Properties 167
7.5 Meter Deskitem Properties 168
7.6 Multi-I/O Deskitem Properties 172
7.7 Display Deskitem Algorithm Argument Properties 176
7.8 Gain Coefficient to dB Mapping 208
7.9 Peak Value to dBFS Mapping 208

B.1 Combinations of Inference Rules Validated 271

Chapter 1

Introduction

As network technologies mature, there is a growing trend towards networking of audio, video, and other data devices. This trend has created a need for distributed device discovery, connection management, control, and monitoring. Device discovery involves the scanning of a network by a device in order to identify other devices that exist on the same network. Connection management of audio or video is the process by which audio or video signals are routed from transmitting devices to audio or video receiving devices, where the devices can be in different physical locations. This routing is configurable via a device, typically referred to as a controller, that is part of the multimedia network. Control refers to the process by which properties of devices within a network are changed remotely, while monitoring refers to the manner in which the statuses of these properties are observed. This thesis focuses on control and monitoring.

A number of companies and standards bodies are continuously introducing technologies that enhance user experience within high-speed digital multimedia networks. These technologies focus on device discovery, connection management, control, monitoring, and transport of audio or video signals. A recent report produced by the Audio Engineering Society Technical Committee on Network Audio Systems on "Emerging Trends in Audio Engineering" [AES Technical Committee on Network Audio Systems, 2012] cited the following emerging trends, among others:

- EBU N/ACIP [European Broadcasting Union, 2008] - A framework defined by the Network Management Committee of the European Broadcasting Union for the transmission of audio over IP networks. The main goal of this framework is to achieve interoperability between audio over IP contribution devices mainly in the broadcasting industry. The

framework defines the transport protocols that are used on top of IP, media data formats, connection set up and termination procedures.

- Audio Video Bridging (AVB) [The Audio/Video Bridging Task Group, 2012] - A suite of standards that are aimed at providing high-speed, real-time performance to Ethernet networks. These standards make it possible for AVB compliant devices to coexist with standard (non-AVB compliant) Ethernet devices.
- RAVENNA [ALC NetworX GmbH, 2011] - A standards-based initiative that focuses on real-time delivery of audio and other media within IP networks. Standard protocols are mainly used for communication within local- and wide-area networks (LANs and WANs), media content data formats, media clock synchronisation, and quality of service (QoS).
- Dante [Audinate, 2009] - A proprietary network technology by Audinate for the transport of audio over IP. Dante provides simple plug and play operation, and PC sound card integration.
- Q-LAN [Gross, 2009] - A standards-based networked media distribution technology implemented by QSC Audio Products. Q-LAN uses gigabit Ethernet or higher rate IP networks.
- XFN [Audio Engineering Society, 2012] - An IP based peer to peer multimedia network control protocol. At the time of this research, the XFN protocol was undergoing standardisation within the AES; the AES-64 standard has now been approved. "XFN", short for "Cross-Fire Network" protocol, is the project code name that was used to refer to the AES-64 protocol prior to standardisation. This is the name used in this thesis.
- Open Control Architecture (OCA) [Berryman, 2013] - A control and monitoring architecture for media networks that was formed by a group of professional audio companies. The goal of OCA is to create an open media networking control system standard that will promote interoperability across vendors. This architecture is currently undergoing standardisation.

In addition to the emerging trends listed above, a number of frameworks and protocols for media control and monitoring have existed for some time. These include, but are not limited to:

- Open Sound Control (OSC) [Wright and Freed, 1997] - An open, transport-independent, message-based protocol for communication among computers, sound synthesizers, and

other multimedia devices. The protocol was developed by the Center for New Music and Audio Technologies (CNMAT), University of California at Berkeley.

- IEC 62379 [International Electrotechnical Commission, 2007] - A set of standards that defines a control framework for networked multimedia devices. The control framework was originally developed for radio broadcast and later extended to video and other time-critical media.
- Audio Video Control (AV/C) [1394 Trade Association, 2001] - A protocol that allows for control of audio or video devices on an IEEE 1394 bus. IEEE 1394 [1394 Trade Association, 2001] [IEEE Std. 1394a, 2000] [IEEE Std. 1394b, 2002], commonly known as *firewire*, is a high-performance serial bus standard.
- Architecture for Control Networks (ACN) [Entertainment Services and Technology Association, 2005] - A modular architecture for control protocols that was originally created for lighting control, although also applicable to other media networks. The architecture is composed of a number of protocols and languages that can be put together to build media network control systems.

1.1 Problem Statement

With the adoption of multimedia network technologies, the need for flexible control and monitoring mechanisms is evident. Parameters, such as volume, exist in devices that are spread across a network. It is often necessary for a change in one parameter to affect other parameters on the same or other devices on the network. Thus, relationships have to be set up between such parameters. For example:

- In a live concert venue, it might be desirable for the gains of all amplifiers in the venue to be controllable by a single gain control.
- In a hotel ballroom, there could be a need to raise or lower the volumes of all active speakers collectively.
- Show control systems require the linking of more than one production element together [Huntington, 2007]. For instance, a show control system might link the control of a fog machine with an audio playback system that generates maritime sound effects. This is

an example of a scenario where relationships are required between parameters that have different underlying value units.

- In other environments, it might be necessary for a synchronised change between volume and pan parameters to be made, where, for example, at higher volume levels, sound appears to be more concentrated on either the left or right stereo channel. Again, pan and volume parameters have different underlying value units.

The concept of parameter relationships is prevalent within mixing consoles, where it is possible to group controls (such as faders) on a single mixing console. This grouping then makes it possible for changes in value of one fader to cause changes in value of another fader, for example. Parameter relationships similar to those found in mixing consoles have not been comprehensively explored within the context of devices that are distributed across a network.

There have been attempts at grouping networked parameters at the application level, particularly in ACN [Entertainment Services and Technology Association, 2009a] and the OSC protocol [Wright, 2002]. However, implementing grouping at the application level implies that grouping relationships are reinvented by applications. None of these attempts comprehensively incorporates parameter relationships in a protocol. Thus, none have emerged as the *de facto* standard for governing the creation of relationships between parameters that reside in different devices on a network. If grouping of networked parameters is incorporated into a standard protocol, it is possible for all applications to depend on the grouping relationships, and not to reinvent grouping within applications. Consequently, the need for a standardised protocol that manages relationships between parameters residing in different devices on a multimedia network is evident.

The Open Control Architecture defines a mechanism for grouping of networked parameters with identical value types via a proxy node [Berryman, 2013]. This approach does not cater for scenarios where parameters with disparate value types are grouped, such as grouping of parameters that control a fog machine with audio properties, as described above. In addition, centralisation of control creates the potential for a single point of failure, where an entire group ceases to function when the proxy node is offline. Therefore, a mechanism that allows for more decentralisation of control and flexibility of grouping parameters with disparate value types is required.

In addition to relationships between parameters, it is important to monitor the statuses of various parameters of multimedia devices on a network. The monitored parameters can be of different types, such as voltage, audio level, and temperature parameters. Parameter monitoring is required in a number of scenarios, including, but not limited to:

- When the gain is set too high in an audio amplifier, such that the amplified audio signal exceeds the capabilities of the amplifier, the audio signal is said to be *clipped*. Audio clipping, also known as audio overload, is a type of distortion that limits an audio signal once it exceeds a threshold. In order to prevent clipping, it is often necessary to monitor the peak levels of the amplified audio signal and apply the necessary corrective action, such as reducing the gain.
- In some environments, it might be necessary to monitor the temperature of devices, such as amplifiers and video encoders/decoders, in order to prevent damage due to overheating.

Monitoring within any network protocol can be achieved by polling the statuses of parameters at regular intervals. However, polling increases network and device load, potentially causing a negative impact on network and device performance. Instead, event-driven notification mechanisms are preferred as a means for parameter monitoring. Multimedia networks can contain vast numbers of parameters that require monitoring. Consequently, a generic event-driven mechanism for bulk monitoring of parameters is required.

1.2 Aims and Objectives

The aims of this investigation were:

- To determine the feasibility of decentralised control over parameters that are spread across a high-speed multimedia network, by creating relationships between the parameters.
- To define a generic mechanism for bulk monitoring of parameters on a high-speed multimedia network.

In order to fulfil these aims, the following objectives were formulated:

- To identify the relationships that can be created between controls in mixing consoles and, subsequently, derive a classification scheme for creating relationships between parameters that are distributed across multimedia networks.
- To review the capabilities of existing control protocols and, subsequently, derive further requirements for parameter grouping and monitoring within multimedia networks.

- To propose implementation strategies to fulfil the requirements that were derived.
- To demonstrate the usability of the proposed implementation strategies in the context of a graphical workstation application.

This study has been published in the Journal of the Audio Engineering Society in "Parameter Relationships in High-Speed Audio Networks" [Chigwamba, Foss, Gurdan, and Klinkradt, 2012]. Some of the concepts derived from this study have been incorporated in the AES-64 standard. The study resulted in a fully functional software prototype that, among other features, enables the creation of relationships between parameters that are spread across a network in a manner that was previously not possible. The implementation uses XFN as the underlying control protocol. Motivations for the choice of XFN include:

- XFN is an IP-based transport independent control protocol. The protocol has been tested in Ethernet AVB and IEEE 1394 networks [Foulkes, 2011].
- XFN set out to become an open standard of the Audio Engineering Society (AES-64) [Audio Engineering Society, 2012].
- XFN models devices using XFN parameters that are hierarchically addressed according to the devices' functional layout. A callback mechanism associates XFN parameters with application-specific parameters, such as audio level parameters of a DSP. The existence of an XFN parameter per application-specific parameter makes it possible to define a parameter grouping and monitoring mechanisms at the XFN protocol level. Such parameter grouping and monitoring mechanisms cascade to corresponding application-specific parameters. Thus, any XFN-capable devices can reuse the standard parameter grouping and monitoring mechanisms that are defined within the XFN protocol.

1.3 Document Structure

Chapter 2 gives an overview of mixing consoles and the common grouping capabilities that exist. Grouping capabilities in mixing console implementations from various leading manufacturers are also reviewed. This chapter provides a basis for the requirements for parameter grouping relationships that are described in Chapter 4.

Chapter 3 describes the grouping and monitoring capabilities that exist in mainstream control protocols. The strengths and weaknesses of these existing capabilities are highlighted. This chapter also provides a basis for the requirements for parameter grouping and monitoring relationships that are described in Chapter 4.

Chapter 4 consolidates the results from the literature review of parameter grouping and monitoring capabilities within mixing consoles and mainstream control protocols given in Chapters 2 and 3. A generic set of requirements for parameter grouping and real-time monitoring within networks is laid out in this chapter.

Chapter 5 gives an overview of the XFN protocol. This is followed by a description of the implementation strategies that were proposed to fulfil the requirements for parameter grouping relationships in the context of the XFN protocol.

Chapter 6 describes an existing mechanism for bulk parameter value retrieval in the XFN protocol. This is followed by a description of the implementation strategies that were proposed to fulfil the requirements for parameter monitoring, in the context of the XFN protocol, by augmenting its bulk parameter value retrieval mechanism.

Chapter 7 gives a description of how the implementation strategies proposed for parameter grouping and monitoring were applied to an XFN-protocol-compliant graphical control application.

Chapter 8 concludes the thesis by highlighting the important aspects of the work carried out in this investigation.

Chapter 2

Parameter Relationships in Mixing Consoles

Chapter 1 has indicated that parameters exist in devices that are spread over a network, where it is often necessary for a change in one parameter to affect other parameters. Thus, relationships have to be set up between such parameters. Similar groupings of parameters exist in some mixing consoles. A mixing console can be viewed as a mini audio network, and many of the grouping issues in a mixing console will parallel those on a network. This chapter begins by giving a broad overview of what mixing consoles are and how they work, followed by a review of parameter grouping capabilities within mixing consoles. Descriptions of examples of mixing consoles that have been created by some of the leading manufacturers will be provided with particular emphasis on an evaluation of their grouping capabilities. Based on the review of parameter grouping relationships that are possible in mixing consoles, a classification scheme for parameter relationships within high-speed multimedia environments was derived. A detailed description of this classification scheme will be given in Chapter 4.

2.1 Mixing Consoles

Izhaki [2013] and Rumsey and McCormick [2009] contextualise a mixing console as a device that is used alongside a multitrack recorder during a recording or mixing session. Mixing consoles are also used in a number of other contexts, such as public address systems and live sound

environments. However, for the purposes of this section the multitrack recording context will be used.

Functions implemented by mixing consoles include, but are not limited to:

- Summing - the process by which a number of input audio signals are combined (also referred to as mixing) to produce a combined output signal, typically a pair of audio signals (stereo). Depending on the type of mixer, both analogue and digital audio signals can be mixed to produce the combined output signal.
- Processing - manipulating various components of an audio signal, such as frequency, in order to produce a resulting audio signal that is different from the original signal.
- Routing - this functionality allows for the use of external devices to further process or add effects to audio signals via insert points, auxiliary sends, and routing matrices.

Mixing consoles have two main sections, namely a channel section and a master section [Izhaki, 2013]. The channel section contains a number of physical strips, where each strip is a channel that corresponds to a track on the multitrack recorder. The master section is a centralised control section of a mixing console that is responsible for the mixing console's global functions.

Logical groups, for example drum groups and string orchestras, comprising a number of individual tracks, are common within recorded productions. When mixing audio, engineers often need to change the level of, mute, solo, or process a group of channels simultaneously. Groups in mixing consoles are broadly classified under two categories, namely control groups and audio groups. These two group types are described below.

2.1.1 Control Grouping

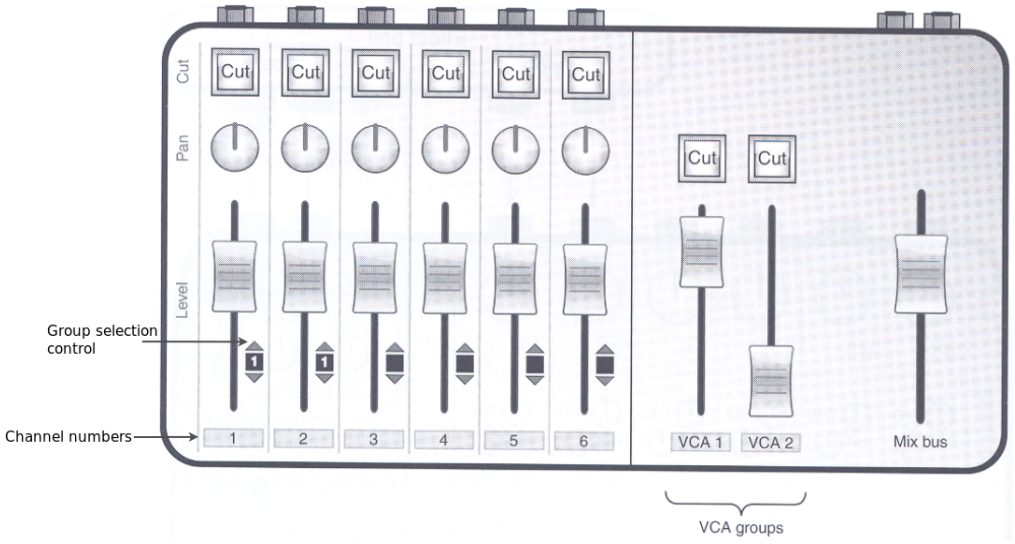
With control groups, a number of channels are allocated to a group such that when one fader moves, the other faders in the group also move simultaneously [Izhaki, 2013; Rumsey and McCormick, 2009]. Control groups behave in two main ways, namely:

1. Linking, where a change that is applied to one channel is also applied to other channels. For example, soloing or cutting one channel also solos or cuts other group channels. The solo function enables an audio engineer to listen to specific tracks in isolation, while a cut function switches the signal of specific tracks on or off.

2. Master-slave relationships, where only changes in the master channel result in changes in the slave channel, while changes in the slave channels do not influence other channels. These relationships are typically implemented by VCA grouping functionality, although it is possible to implement similar group behaviour by other means.

2.1.1.1 VCA Grouping

A number of master VCA group controls such as faders, cut, and solo buttons typically exist in the master section of a mixing console. Figure 2.1 shows an illustration of a basic mixing console with two VCA groups, namely VCA 1 and VCA 2. Each individual channel can be assigned to one of the two VCA groups via a dedicated group selection control per channel strip. In Figure 2.1, channels 1 and 2 are assigned to VCA 1. A change in level of a master VCA group fader results in a change in level of all the channels that belong to the VCA group. Similarly, cutting or soloing a VCA group also cuts or solos all the assigned channels.



Republished with permission of Taylor and Francis Group LLC Books, from Mixing Audio, Roey Izhaki, 2nd Edition, 2013; permission conveyed through Copyright Clearance Center, Inc.

Figure 2.1: A Basic Mixing Console with Two VCA Groups

A description of how VCA faders differ from conventional faders is given below.

2.1.1.1.1 VCA Faders

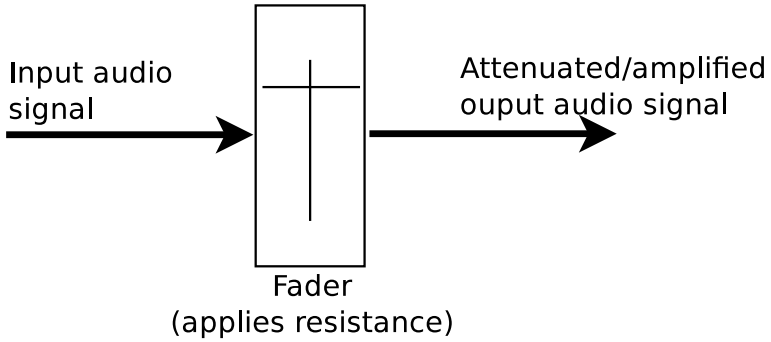


Figure 2.2: Conventional Fader Circuitry

In conventional fader circuitry, as shown in Figure 2.2, audio channel signals pass through the fader for signal level control [Rumsey and McCormick, 2009]. Different fader positions correspond to different resistance values being applied to the fader circuitry, and hence affect the magnitude of signal level attenuation. In contrast, with a VCA fader, audio channel signals pass through a voltage controlled amplifier (VCA) chip as shown in Figure 2.3. The signal level of audio channels is controlled by a variable DC voltage that is fed to the control port of the VCA.

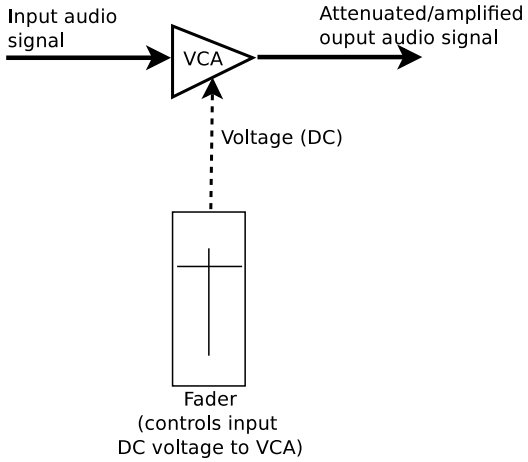


Figure 2.3: VCA Fader Circuitry

Two alternatives for VCA groups are indicated in “Mixing Live with VCAs” [Allen & Heath Limited], namely:

1. Using a single fader to control more than one channel VCA as shown in Figure 2.4.

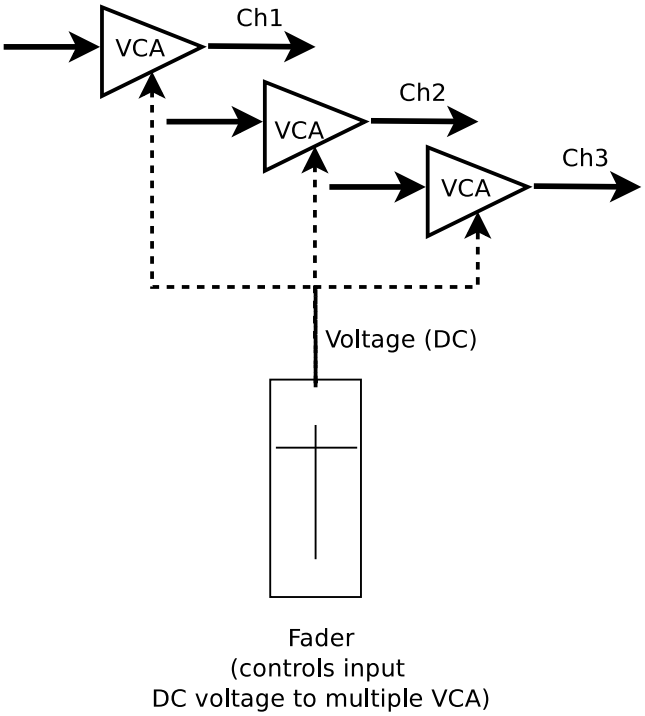


Figure 2.4: Single Fader Controlling Multiple Channel VCAs

2. Using several faders to control the same channel VCA. In this case, the voltages from each of the faders are combined and the resulting voltage is fed to the VCA. Such flexibility in VCA groups allows for complex relationships to be created. Figure 2.5 illustrates this scenario, where there are three channels, labelled Ch1, Ch2, and Ch3. Each channel is controlled by its own VCA fader. Ch1 and Ch2 are also under the control of a master VCA fader. The output audio signal for Ch1 and Ch2 is determined by the voltages of the dedicated channel fader and the master fader. On the other hand, the output audio signal level for Ch3 is determined by the voltage of its channel fader only, since it is not under the control of the master fader.

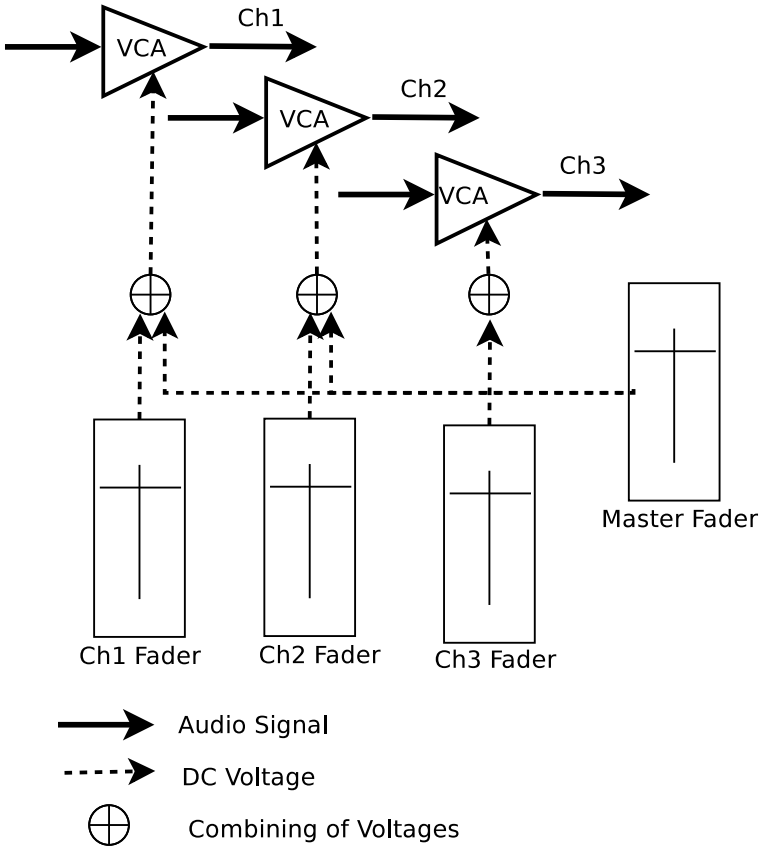


Figure 2.5: VCA Fader Group Illustration

2.1.2 Audio Grouping

As described above, control grouping allows for simultaneous control of multiple channels, where the original signal is not altered. As mentioned at the beginning of this section, one of the functions of mixing consoles is processing, where the original signal is altered to produce another signal. It is often necessary for audio engineers to collectively process multiple signals, such as when compressing the overall drum-mix. In such situations, audio grouping is used.

With audio grouping, a group of channels is summed to a group bus (also referred to as subgrouping) [Izhaki, 2013]. The group signal is then processed, for example by applying compression, and then routed to the mix bus where the output signal is derived. Individual channels are assigned to different groups via routing matrices. There are two main classes of audio groups, namely master groups and in-line groups. These two forms of audio groups are described in Section A.1. We note that audio grouping in this manner is outside the scope of this investigation.

An alternative to subgrouping involves linking the processing of individual channels. This form of audio grouping is described below.

Linking of Processing

Linking of processing enables identical processing of audio channels. In dynamic range compression, for example, audio signals beyond a specified threshold value are either amplified or attenuated by shrinking the audio signal's dynamic range[1] [Izhaki, 2013]. Table 2.6 shows the graph of a compressor with a threshold of -40 dB. Here, input audio signals between -80 dB and -40 dB (below the threshold) are not affected, while input signals above the threshold are halved. Thus, gain depends on the level of the incoming signal.

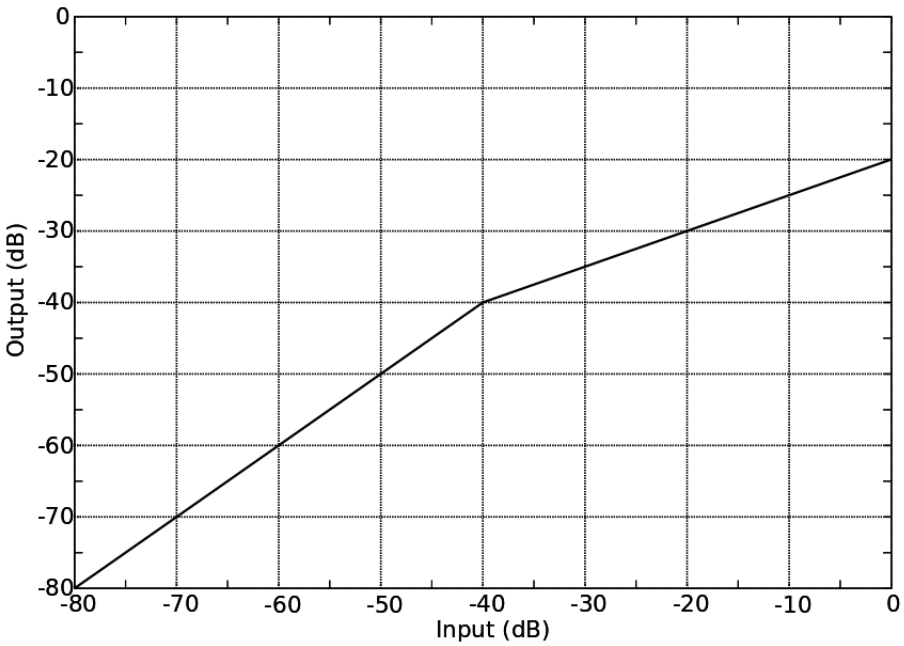


Figure 2.6: Compressor Ratios

Figure 2.7 show an example of a scenario where stereo linking is required. This scenario builds

[1]Dynamic range is the difference between softest and loudest possible sounds of a system. For example, the dynamic range of a system is 80 dB if the softest sound is -80 dB, while the loudest sound is 0 dB.

on the dynamic range description above. Here, there is a pair of stereo audio signals, namely left and right channels. Each audio signal is processed by a compressor. The left audio channel input level is -50 dB, resulting in an uncompressed audio output level of -50 dB. On the other hand, the right audio channel is louder with an input level of -20 dB, resulting in a compressed audio output of -30 dB. The stereo image is affected when compression is applied to only one channel of the stereo pair. Linking the processing of the two compressors ensures that the gain reduction is identical in both stereo channels. This investigation focused on grouping of controls. Thus, linking of processing is outside the scope of this investigation.

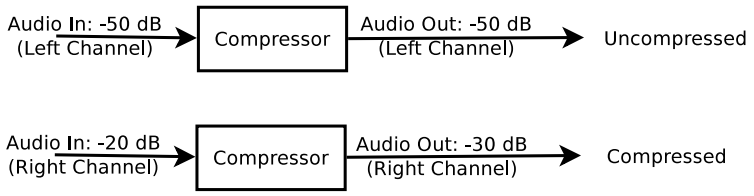


Figure 2.7: Example of Compression Imbalance Requiring Stereo Linking

2.2 Manufacturer Specific Implementations of Mixing Consoles with Particular Reference to Control Grouping

Section 2.1 has given a general overview of mixing consoles and their grouping capabilities. A number of manufacturers have created mixing consoles that implement these grouping capabilities. We now provide descriptions of the control grouping capabilities provided in some of the mixing consoles that have been created by the following manufacturers:

1. Yamaha Corporation
2. Allen & Heath Limited
3. D&R Electronica Weesp b.v.

These choices were made for the reason that the consoles are dissimilar with respect to their grouping capabilities. The human user interfaces and step-by-step procedures for setting up and clearing groups are described in detail in Section A.2. These interfaces provided a reference point for network-based grouping. A description of the graphical user interfaces that were proposed for network-based grouping in this study is given in Chapter 7.

2.2.1 Yamaha Corporation

An overview of the grouping capabilities of the 01V96 mixing console will be given in this section. A detailed description of the human user interface for the 01V96 is given in Section A.2.1.1 on page 241.

2.2.1.1 01V96 Version 2 Digital Mixing Console

The 01V96 features a number of grouping relationships between various controls and parameters. These relationships are laid out in the "01V96 (Version 2) Digital Mixing Console Owner's Manual" [Yamaha Corporation, 2004]. Four main grouping relationships have been identified from the manual, namely:

- Pairing of input/output channels.
- Fader groups.
- Mute groups.
- Linking of equaliser (EQ) and compressor parameters.

A brief description of the capabilities of each of these grouping relationships follows.

2.2.1.1.1 Input/Output Channel Pairing

Adjacent odd-even input/output channels of the 01V96 can be paired together for stereo operation. Examples of input channel pairs are channels 1 and 2, channels 3 and 4, channels 5 and 6, and so on. When two channels are paired, the value of one of the channels is immediately updated to match the other. Thereafter, any changes applied to one channel of the pair are also copied to the other channel in the pair. This type of relationship will be classified as an *absolute peer-to-peer relationship* in Chapter 4.

Parameters that are linked together as a result of pairing include, but are not limited to, channel on/off, solo on/off, equaliser (EQ) settings, compressor settings, mute groups, and fader groups. Fader groups, mute groups, and further linking capabilities of EQ and compressor parameters

will described in Sections 2.2.1.1.2, 2.2.1.1.3, and 2.2.1.1.4 below, respectively. The sections will highlight how these parameters are affected by pair relationships.

Step-by-step descriptions of the procedures followed when setting up and breaking channel pairs are given in Section A.2.1.1.1 on page 245. An analysis of these procedures reveals two limitations with reference to channel pairing, namely:

1. Only channels of similar types can be paired. For example, it is not possible to pair an input channel with an output channel.
2. Pairing is restricted to selected adjacent channel combinations. For example, given a channel strip section with eight faders corresponding to channels 1 to 8, pairing is only possible between channels 1 and 2, channels 3 and 4, channels 5 and 6, and channels 7 and 8. It is not possible to pair other channel combinations, such as channels 1 and 8, channels 3 and 5, *et cetera*.

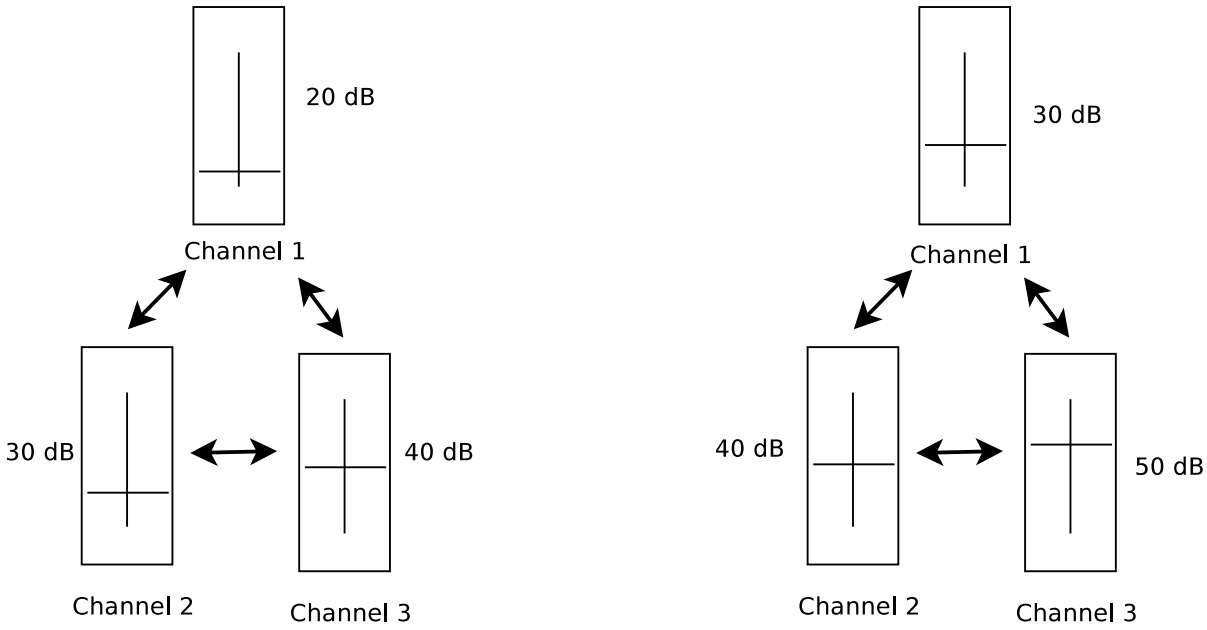
The limitations above are specific to the 01V96 mixing console and should not be imposed in a generic parameter grouping mechanism for high-speed multimedia network environments.

2.2.1.1.2 Fader Groups

The 01V96 has a number of fader controls that can be used to control the levels of both input and output audio signals. The faders can be grouped such that changes in value of one fader result changes in values of one or more additional faders. There are two types of fader groups, described below, namely standard fader groups and master fader groups.

2.2.1.1.2.1 Standard Fader Groups

When a standard fader group is formed, the relative level difference of each fader to all other faders in the same group is stored. Whenever the value of a fader in a group is changed, the other faders within the group also have their values changed automatically in order to maintain the relative level difference that would have been stored at the time of creation of the group. Such grouping relationships between faders will be classified as *relative peer-to-peer relationships* in Chapter 4.



(a) At group creation time

(b) Channel 1 becomes 10 time louder

Figure 2.8: Standard Fader Group Example

We illustrate the behaviour of standard fader groups with an example. Figure 2.8a shows a group that is created with three faders, namely channel 1, channel 2, and channel 3. It should be noted that unlike channel pairing, where only two channels can be paired, standard fader groups allow more than two channels to be added to a group. At the time of group creation the level of channel 1 is 20 dB, channel 2 is 30 dB, and channel 3 is 40 dB. Thus, channel 2 is 10 dBs louder than channel 1, while channel 3 is 10 dBs louder than channel 2. When channel 2 is made 10 dBs louder by increasing its value to 40 dB, the values of channel 1 and channel 3 also become 10 dBs louder and are automatically increased to 30 dB and 50 dB, respectively. This maintains the relative level difference at the time that group was created, where channel 2 is 10 dBs louder than channel 1, while channel 3 is 10 dBs louder than channel 2.

A detailed description of the steps followed when setting up and breaking fader groups via the 01V96 human user interface is given in Section A.2.1.1.2.1 on page 249. Similar to channel pairing, fader groups may consist of either input channels only or output channels only, but not both. However, there is an additional requirement, where the addition of a paired channel to a group also results in the addition of the paired partner channel to the group. For example, if channels 5 and 6 are paired, adding channel 5 to a group with channels 1, 2, and 3 results in the addition of channel 6 to the group; this results in a group comprising channels 1, 2, 3, 5, and 6, thus

retaining the integrity of the relationship between channels 5 and 6. In the context of parameter relationships in a multimedia network, it is important to ensure that the integrity of relationships is maintained when new relationships are established between parameters that potentially spread across a network. Therefore, rules must be defined in order to ensure determinism in the manner in which previous relationships are maintained.

2.2.1.1.2.2 Master Fader Groups

In addition to standard fader grouping as indicated above, the 01V96 has a Fader Group Master function that allows users to control the levels of all grouped channels via a single Group Master level while maintaining the relative level differences between the channels. In this mode, a change in value of any of the grouped faders, other than the Group Master level, results in an adjustment of the relative level difference between the grouped faders as opposed to a relative change in levels. The actual level of each of the channels is determined by the sum of the Group Master level and the individual level of the grouped channel. The Fader Group Master function operates much like a VCA fader as described in Section 2.1.1.1.1, where several faders can be used to control the same channel VCA, and the voltages from each of the faders are summed to give a resulting voltage that is fed to the VCA. The Fader Group Master function will be classified as a *relative master-slave relationship* in Chapter 4.

Detailed step-by-step descriptions of the procedures followed when enabling or disabling the Fader Group Master function are given in Section A.2.1.1.2.2 on page 251. The Fader Group Master function can only be applied to an existing standard fader group, thus it bears the same limitations as standard fader groups regarding its applicability to either input channels only or output channels only, but not both.

2.2.1.1.3 Mute Groups

Each input or output channel can be muted or unmuted by an “on/off” button that resides in the channel strip section. Similar to channel levels, these “on/off” states can be grouped. Similar to fader groups, there are two types of mutes groups, described below, namely standard mute groups and master mute groups.

2.2.1.1.3.1 Standard Mute Groups

Mute buttons have two states, either "on" or "off", therefore standard mute groups have the effect of toggling the state of the other grouped buttons. These relationships are determined at the time of group creation when each button is added to a group. For example:

- If a button that is in the "on" state is grouped with another that is in the "off" state, switching one to the "on" state results in the other switching to the "off" state. This is an example of an inverse relationship.
- If two buttons are grouped while both are in the same state, the state of each of the buttons will always be the same, either both "on" or both "off".

The manner in which standard mute groups operate will be classified as an *absolute peer-to-peer relationship* in Chapter 4. However, this classification does not hold for inverse relationships, such as when buttons are grouped while in different states, where when one is in the "on" state, the other is in the "off" state.

Detailed step-by-step descriptions of the procedure followed when creating and breaking standard mute groups are given in Section A.2.1.1.3.1 on page 253. An analysis of this procedure reveals that mute groups, similar to fader groups, apply to either the input channels only or output channels only, but not both. In addition, when the "on/off" button of a channel that is paired to another is added to a group, the paired partner is also added to the group in order to retain the integrity of the relationship between the paired channels.

2.2.1.1.3.2 Master Mute Groups

Similar to fader groups, the 01V96 has a Mute Group Master function that allows users to mute grouped channels via a single Master Mute button. However, when the Mute Group Master function is enabled, changing the "muted/unmuted" state of any of the grouped channels does not affect the other group members. Instead, when the Master Mute button is in the "muted" state, all grouped channels will be in the "muted" state. In contrast, when the Master Mute button is in the "unmuted" state, all grouped channels will be in the "unmuted" state. This implies that the Mute Group Master function takes precedence over the standard mute groups, while only one

of the two types of groups can be used at any point. The Mute Group Master function will be classified as an *absolute master-slave relationship* in Chapter 4.

Detailed step-by-step descriptions of the procedures followed when enabling or disabling the Mute Group Master function are given in Section A.2.1.1.3.2 on page 254. This function can only be applied to existing standard mute groups, therefore it bears the same limitations as standard mute groups regarding its applicability to either input channels only or output channels only, but not both.

2.2.1.1.4 Linking EQ and Compressor Parameters

Compressor and EQ parameters on the 01V96 can be grouped, although their grouping is somewhat different from that of fader and mute groups. The main restriction with EQ and compressor parameter groups is that all parameters share the same value, where the settings for the first channel added to the group are applied to all subsequently added group members. This type of relationship will be classified as an *absolute peer-to-peer relationship* in Chapter 4.

Detailed step-by-step descriptions of the procedure followed when linking EQ and compressor parameters are given in Section A.2.1.1.4 on page 255. Again, it should be noted that when a paired channel is added to a link, the pair partner is also added to the link in order to retain the integrity of the relationship between the paired channels.

2.2.2 Allen & Heath Limited

This section gives an overview of the grouping capabilities of the ML3000, ML4000, and ML5000 mixing consoles that have been created by Allen & Heath Limited.

2.2.2.1 ML3000, ML4000, and ML5000 Live Sound Consoles

The ML3000, ML4000, and ML5000 are live sound consoles that can be configured for front-of-house (FOH) or stage monitor mixing [Allen & Heath Limited, 2003a,b,c]. Although these sound consoles have a number of features and capabilities, two forms of control grouping relationships are of relevance to this investigation, namely VCA groups and mute groups. For the

remainder of this section, we focus on the ML5000 mixing console, since it is at the top of the ML mixing console series and it implements grouping capabilities that are similar to the ML3000 and ML4000 mixing consoles.

2.2.2.1.1 VCA Groups

The ML5000 mixing console has eight VCA groups to which mono and stereo input channels can be assigned. Each VCA group has a fader associated with it. It is possible to assign a channel to more than one of the eight VCA groups. The "0" position of the VCA group fader is the nominal operating setting, where the levels of the channels assigned to the VCA group are unaffected by the VCA group fader. However, any adjustment made to the group fader offsets the levels of the channels assigned to the VCA group by an equivalent amount. This relationship will be classified as a *relative master-slave relationship* in Chapter 4.

Allen & Heath Limited [2003c] give an example of a theatre musical production to illustrate the assignment of a single channel to multiple VCA groups, where:

- Stage microphones are assigned to group 1.
- Radio microphones are assigned to group 2.
- All microphones are assigned to group 3.
- All channels are assigned to group 8 as "grand master" that controls the overall volume.

In this example, a radio microphone would be assigned to groups 2, 3, and 8. A detail description of the step-by-step procedures for assigning and clearing VCA groups is given in Section A.2.2.1.1. A description of the functionality provided by VCA groups has been given in section 2.1.1.1 on page 10. Examples of VCA grouping scenarios are analysed below.

2.2.2.1.1.1 VCA Group Examples

Three VCA group examples are described in "ML5000 User Guide AP3736 Issue 5" [Allen & Heath Limited, 2003c], where a single channel is assigned to three VCA groups. An analysis of these examples, particularly the second example, highlights the behaviour of VCA groups, where it may be necessary to combine gains by multiplication as opposed to addition.

Example 1

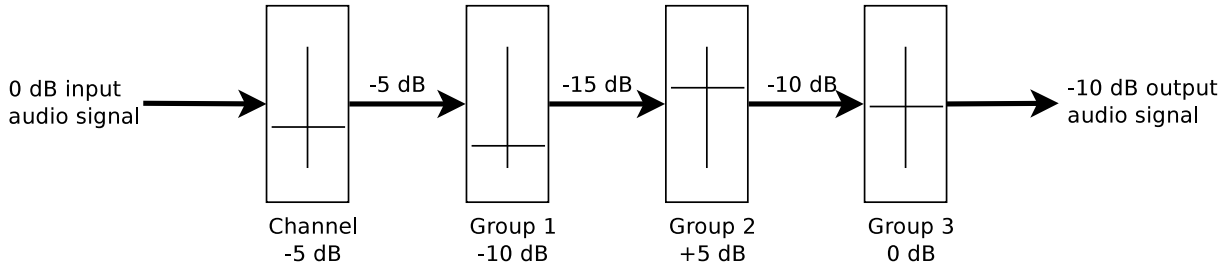


Figure 2.9: VCA Group Example 1

Figure 2.9 illustrates the following:

1. A 0 dB signal passes through the channel.
2. The channel fader attenuates the 0 dB signal by 5 dB, resulting in a -5 dB signal.
3. Group 1 further attenuates the -5 dB signal by 10 dB, resulting in a -15 dB signal.
4. Group 2 boosts the -15 dB signal by 5 dB, resulting in a -10 dB signal.
5. Group 3 is set to 0 dB and hence has no effect on the signal level, resulting in a final signal level of -10 dB.

In this example, it is possible to compute the output audio signal by addition of the decibel values of corresponding fader gains. However, the next example illustrates a scenario where computation by multiplication is preferred to addition.

Example 2

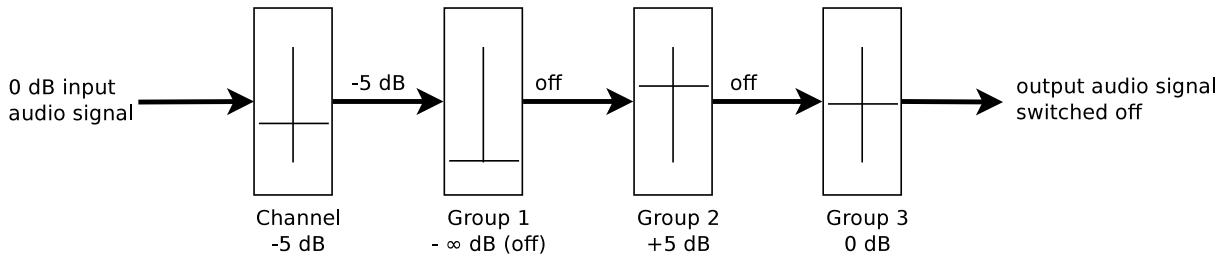


Figure 2.10: VCA Group Example 2

Figure 2.10 illustrates the following:

1. A 0 dB signal passes through the channel.
2. The channel fader attenuates the 0 dB signal by 5 dB, resulting in a -5 dB signal.
3. Group 1 is set to minimum (-∞), resulting in the signal being turned off.
4. There is no output regardless of the settings of groups 2 and 3.

This example highlights an important consideration for VCA groups, where the output audio signal is always switched off (-∞) whenever one of the faders is switched off. Here, the values of the channel, group 1, group 2, and group 3 faders are -5 dB, -∞ dB, +5 dB, and 0 dB, respectively. Using addition to combine the gains is not possible, unless the value of -∞ is defined. Instead, a multiplication based calculation yields the expected results. Audio gain is typically the ratio of output voltage to input voltage, calculated as follows:

$$G = 20\,log_{10}\frac{V_{out}}{V_{in}}$$

where:

- G is gain in decibels
- V_{in} is the input signal voltage
- V_{out} is the output signal voltage

When an audio signal is switched off (-∞ dB), the output voltage is 0, resulting in a gain calculation of:

$$G = 20log_{10}\frac{0}{V_{in}}$$

$$\implies G = 20log_{10}0$$

$$\implies G = 20(-\infty)$$

$$\implies G = -\infty$$

In the context of this example, the sum of the gains (in decibels) is:

$$G_{sum} = G_{channel} + G_{group1} + G_{group2} + G_{group3}$$

$$\implies G_{sum} = 20log_{10}\frac{V_{out_{channel}}}{V_{in_{channel}}} + 20log_{10}\frac{V_{out_{group1}}}{V_{in_{group1}}} + 20log_{10}\frac{V_{out_{group2}}}{V_{in_{group2}}} + 20log_{10}\frac{V_{out_{group3}}}{V_{in_{group3}}}$$

$$\implies G_{sum} = 20\left(log_{10}\frac{V_{out_{channel}}}{V_{in_{channel}}} + log_{10}\frac{V_{out_{group1}}}{V_{in_{group1}}} + log_{10}\frac{V_{out_{group2}}}{V_{in_{group2}}} + log_{10}\frac{V_{out_{group3}}}{V_{in_{group3}}}\right)$$

$$\implies G_{sum} = 20log_{10}\left(\frac{V_{out_{channel}}}{V_{in_{channel}}} * \frac{V_{out_{group1}}}{V_{in_{group1}}} * \frac{V_{out_{group2}}}{V_{in_{group2}}} * \frac{V_{out_{group3}}}{V_{in_{group3}}}\right)$$

Given that the ratio between the output of group 1 and its input is zero $\left(\frac{V_{out_{group1}}}{V_{in_{group1}}} = 0\right)$, the resulting sum of the gain (in decibels) becomes:

$$G_{sum} = 20log_{10}\left(\frac{V_{out_{channel}}}{V_{in_{channel}}} * 0 * \frac{V_{out_{group2}}}{V_{in_{group2}}} * \frac{V_{out_{group3}}}{V_{in_{group3}}}\right)$$

$$\implies G_{sum} = 20log_{10}\left(0\right)$$

$$\implies G_{sum} = 20\left(-\infty\right)$$

$$\Longrightarrow G_{sum} = -\infty$$

The example above illustrates a case where the combination of gains by multiplication is preferred to addition.

Example 3

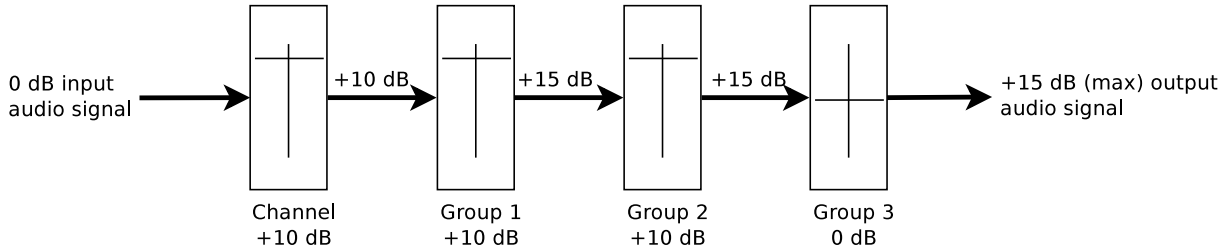


Figure 2.11: VCA Group Example 3

Figure 2.11 illustrates the following:

1. A 0dB signal passes through the channel.
2. The channel fader, group 1, and group 2 faders are set to +10dB boost.
3. The group 3 fader is set to 0 dB boost, thus has no effect on the resulting signal level.
4. The channel VCA reaches its maximum of +15dB regardless of the combined 30dB boost.

2.2.2.1.2 Mute Groups

The ML5000 mixing console implements eight mute groups [Allen & Heath Limited, 2003c]. These mute groups allow users to turn a selected combination of channels on or off with a single button press. When the mute group button is turned on, the mute group buttons of the assigned channels are also turned on. Similarly, when the mute group button is turned off, the mute group buttons of the assigned channels are also turned off. This type of relationship will be classified as an *absolute master-slave relationship* in Chapter 4. Detailed step-by-step descriptions of the procedure for assigning and removing channels to and from mute groups are given in Section A.2.2.1.2 on page 263.

2.2.3 D&R Electronica Weesp b.v.

In this section, we look at a software automation application, PowerVCA, and a standalone mixing console, Sirius, that have been created by D&R Electronica Weesp b.v.

2.2.3.1 PowerVCA

D&R has developed proprietary PowerVCA software that allows for automated fader-mix movements on D&R consoles [D&R Electronica Weesp b.v., 2010a]. PowerVCA records and plays mix movements to and from PC storage. SMPTE timecode is used to provide a reference for editing recorded mixes as well as for synchronisation of audio and/or video. PowerVCA provides a number of graphical controls, such as channel faders that control the signal level of an audio channel, and buttons that can be used to mute audio channels of the attached D&R mixing console.

Of relevance to this investigation, PowerVCA implements a grouping capability. There are eight group master faders available. Each audio channel fader (slave) can be associated with only one master fader, while a master can have more than one slave. When a master fader's level changes, a similar relative change in the level of the associated slave fader levels occurs. However, when a slave fader's level changes, the associated master fader levels are not altered. This type of relationship will be classified as a *relative master-slave relationship* in Chapter 4.

The procedures for assigning and removing channels to and from masters using PowerVCA are given in Section A.2.3.1.1. Unlike VCA grouping in the descriptions above, where multiple master faders can become master to a single slave fader, PowerVCA allows slave faders to have only one master. In distributed multimedia environments, it is important to enable the flexibility of multiple masters per slave. Thus, the parameter grouping scheme proposed by this study supports multiple masters per slave.

2.2.3.2 Sirius

The Sirius console is a software based digital mixing console that implements four VCA fader groups [D&R Electronica Weesp b.v., 2010b]. Only one group master can be assigned per group, although it is possible to have more than one slave per master. This relationship will be classified

as a *relative master-slave relationship* in Chapter 4. The procedures followed when creating and breaking these master-slave relationships are described in Section A.2.3.2 on page 268.

Unlike the other mixing console implementations that were reviewed, the grouping capabilities of the Sirius mixing console are limited. Similar to PowerVCA, only master-slave relationships between channels are supported. No further information regarding other grouping capabilities was available.

2.3 Chapter Summary

A description of mixing consoles and their functions has been given in this chapter. Control grouping capabilities within mixing console implementations from three vendors have been reviewed in detail. Five types of parameter grouping relationships have been identified across the entire set of mixing console implementations under review. These relationship types have been further classified based on a scheme that will be described in Chapter 4. The relationship types identified are:

1. Linking
   Changes applied to one channel in a group are copied to all other channels in the group, resulting in all channels taking on the same value. This relationship will be classified as *absolute peer-to-peer* in Chapter 4.
2. Standard fader groups
   The relative level difference of each fader to all other faders in the same group is stored when a group is created. Whenever the level of a fader in a group is changed, this relative level difference is kept constant by appropriately adjusting the levels of the other faders in the group. This relationship will be classified as *relative peer-to-peer* in Chapter 4.
3. Master fader groups
   A single group master fader is used to control the levels of all grouped channels, while the relative level differences between the channels are maintained. Changes in level of any of the grouped channels other than the group master fader result in an adjustment of the relative level difference between the grouped channels. This relationship will be classified as *relative master-slave* in Chapter 4.

4. Standard mute groups
   Mute/unmute buttons of input or output channels are grouped. This grouping has the effect of toggling the states of other grouped buttons upon a change in state of any of the grouped buttons. This relationship will be classified as *absolute peer-to-peer* in Chapter 4.
5. Master mute groups
   Grouped channels are muted via a single master mute button. When the master mute button is in the "muted" state, the mute buttons of the grouped channels will also be in the "muted" state. In contrast, when the master mute button is in the "unmuted" state, all mute buttons for the grouped channels will also be in the "unmuted" state. This relationship will be classified as *absolute master-slave* in Chapter 4.

Some points of consideration for network-based parameter grouping include:

- Retention of the integrity of pre-existing parameter relationships upon creation of additional parameter relationships.
- The manner in which values are combined, either by addition or multiplication.
- The co-existence of multiple relationships, for example, the ability to control a slave parameter with multiple master parameters.
- The elimination of restrictive parameter relationships, where possible. For example, allowing relationships to be created between input and output parameters.

The human user interfaces for creating and clearing groups on the various mixing console implementations have also been described in the appendix associated with this chapter - Appendix A. These interfaces provide a reference point for network-based grouping. In the context of this investigation, Chapter 7 will describe the human user interfaces that were developed for setting up and clearing relationships between parameters in multimedia network environments. The next chapter reviews the parameter grouping and monitoring capabilities within some mainstream multimedia network technologies.

Chapter 3

Existing Multimedia Network Technologies

A number of multimedia network technologies have been created to help users achieve connection management, control, and monitoring of signals. This chapter reviews the control grouping and parameter monitoring capabilities of five mainstream multimedia network technologies, namely:

1. Open Sound Control (OSC) Protocol
2. IEC 62379
3. Audio Video Control (AV/C)
4. Architecture for Control Networks (ACN)
5. Open Control Architecture (OCA)

A brief overview of the underlying mechanism for each technology will be given, followed by an examination of the grouping and monitoring capabilities, if any, provided by the network technology. This review highlights the need for a standard parameter grouping mechanism that is defined at the protocol level. In addition, the review identifies the preferred approach to parameter monitoring in multimedia networks.

3.1 Open Sound Control (OSC) Protocol

The OSC protocol was created by the Center for New Music and Audio Technologies (CNMAT), at the University of California, Berkeley. Wright and Freed [1997] describe the OSC protocol as "an open, efficient, transport-independent, message-based protocol developed for communication among computers, sound synthesizers, and other multimedia devices". In particular, the protocol describes the formatting of data that is transmitted via a number of network technologies, such as IEEE 1394, Ethernet, and Fast Ethernet. In this section, an overview of the fundamentals of the OSC protocol is presented. Grouping capabilities that are enabled by the OSC protocol will also be reviewed.

3.1.1 Protocol Overview

All OSC data is transmitted in the form of *OSC packets* [Wright, 2002]. OSC defines two types of applications that can either send or receive OSC packets, namely an *OSC client* and an *OSC server*, respectively. It is possible for a single application to send and receive OSC packets simultaneously, hence acting as both an OSC server and OSC client.

An OSC server typically implements a number of *OSC methods* that are arranged in a tree structure called an *OSC address space*. The leaf nodes of the tree represent the OSC methods that are implemented by the OSC server, while the branch nodes comprise *OSC containers*. OSC methods are the points of control for various parameters in a device. Each OSC container other than the root container has a symbolic name that consists of an ASCII string of printable characters that are chosen from a defined set of possible characters. An OSC client typically invokes an OSC method within an OSC server by sending an OSC packet which contains a symbolic name of the full path to the OSC method within the OSC address space. The symbolic name that gives the full path to the OSC method is known as the *OSC address*.

An OSC address begins with a forward slash character ('/'), followed by the names of all the containers starting from the root of the tree to the OSC method. The names of containers are also separated by forward slash characters. Figure 3.1 shows an example of an OSC address space that contains two OSC methods, where the two OSC methods are addressed as follows:

- /a/b/c/d/x

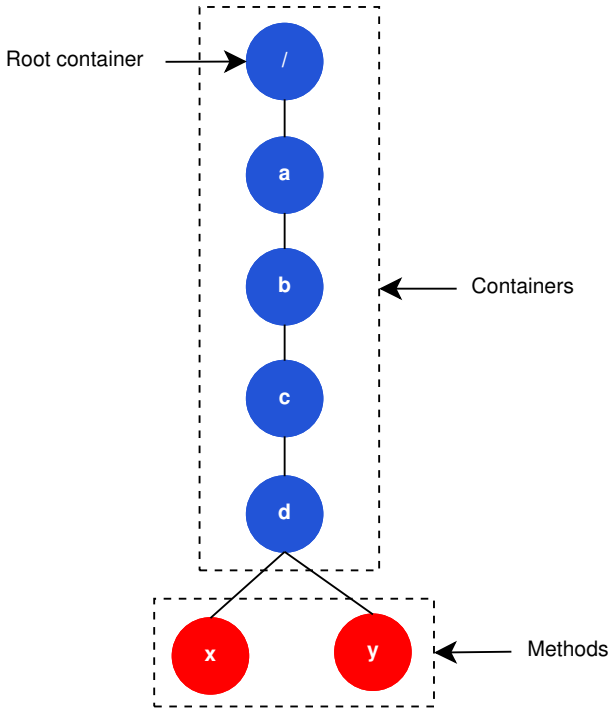


Figure 3.1: Example of an OSC Address Space

- /a/b/c/d/y

OSC method invocation is similar to a procedure call, hence OSC methods are invoked with a variable number of arguments that are provided by an OSC client within an OSC packet. An OSC packet can either be an *OSC message* or an *OSC bundle*. An OSC message comprises an OSC address pattern and one or more arguments, while an OSC bundle wraps one or more OSC messages or bundles for execution at a specified time. OSC address "patterns" are used within OSC messages. This makes it possible for one OSC message to target multiple methods, since all methods that match a given pattern are invoked with all the arguments that follow the OSC address pattern of the OSC message. A number of characters are reserved for pattern matching purposes and are not allowed to be used in the symbolic names for OSC containers and methods. Pattern matching rules that are similar to regular expressions have been defined for OSC address patterns, where:

1. A "?" matches any single character.
2. A "*" matches any sequence of zero or more characters.

3. A string of characters inside square brackets, such as "[abc]", matches any character in the string.
4. A comma-separated list of strings in curly braces, such as "{abc,xyz}", matches any of the strings in the list.

3.1.2 Parameter Grouping Capabilities within the OSC Protocol

Section 3.1.1 has described how multiple OSC messages can be sent in a single packet (OSC bundle). In addition, the pattern matching rules described in the same section make it possible for a single OSC message to invoke multiple OSC methods. From these two types of behaviours, it is clear that the OSC protocol provides a mechanism for an OSC client to invoke multiple OSC methods simultaneously. Simultaneous method invocation can be viewed as a way of enabling grouped control over multiple parameters, where the OSC client manages the group behaviour and uses OSC protocol's data representations to atomically alter the parameter values of grouped parameters.

No explicit parameter grouping mechanisms have been defined for OSC. However, ongoing research within the Audio Engineering Research Group at Rhodes University, South Africa, has shown that OSC is flexible enough to allow for application-specific implementations of parameter grouping [Eales, 2012]. Eales [2012] focuses on controlling mixing consoles using a number of control protocols. For the remainder of this section, we discuss the recommendations that have resulted from his research with regards to OSC grouping. For clarity and simplicity, Eales [2012] defines two types of applications, namely a controller (OSC client) and a device (OSC server), where:

- A controller has a number of controls (typically graphical widgets).
    - Each control has one or more control values.
- A device has a number of parameters that represent, for example, the digital signal processor (DSP) of a mixing console.
    - Each parameter is addressable in order to allow controllers to modify the parameter's value.

Eales [2012] uses the classification of parameter grouping relationships that resulted from this study as a basis for relationships, where master-slave and peer-to-peer relationships are further classified as either relative or absolute. This classification scheme for relationships will be discussed in Chapter 4. We now describe two main alternatives for OSC parameter grouping given by Eales [2012], namely one where grouping is managed by the controller and another where grouping is managed by the device.

3.1.2.1 Alternative 1: Grouping Managed by a Controller

In this alternative, device parameters are associated with OSC methods. A controller discovers all the OSC methods of a device and hence, indirectly, discovers the device's parameters. Each of the controls on the controller is then associated with the device's parameters via the corresponding OSC methods. The controller groups the controls. Grouping of controls implies that the associated device parameters are grouped.

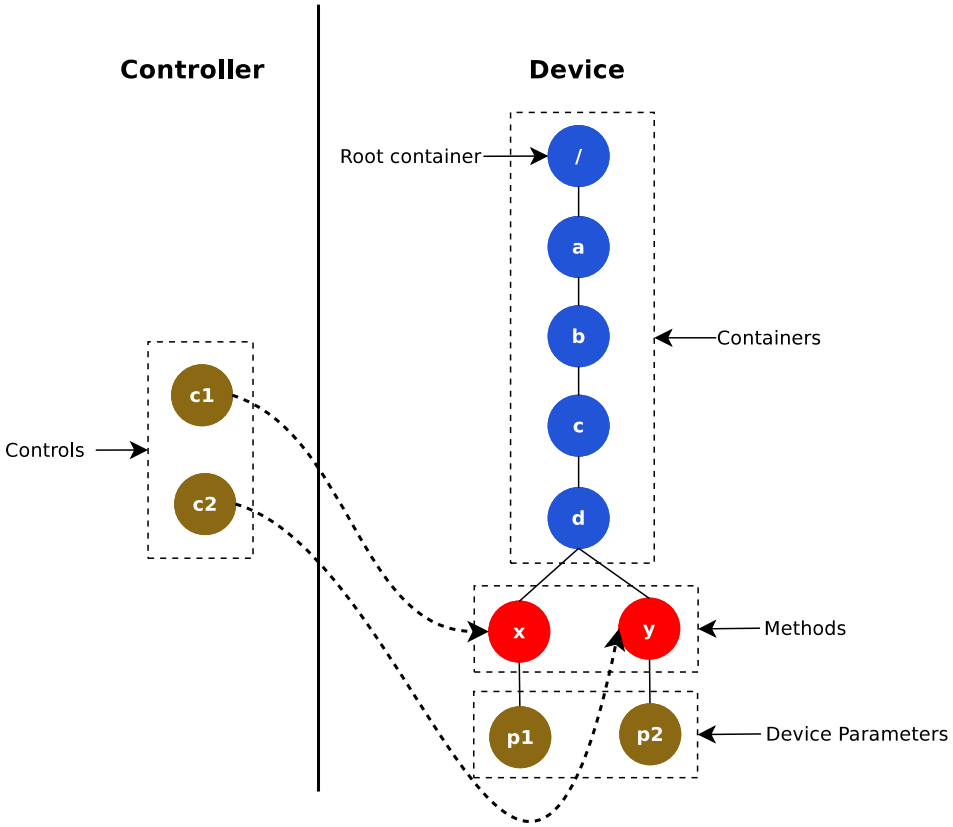


Figure 3.2: Device Parameters Mapped to OSC Methods with a Single Controller

Figure 3.2 shows an illustration of this grouping mechanism that builds up from our earlier example in Section 3.1.1, showing:

- A controller with two controls, named “c1” and “c2”.
- A device with two OSC methods at the addresses “/a/b/c/d/x” and “/a/b/c/d/y”.
- Control “c1” is associated with method “x”.
- Control “c2” is associated with method “y”.
- Method “x” is associated with the device parameter named “p1”.
- Method “y” is associated with the device parameter named “p2”.

Assume that the controller in Figure 3.2 groups controls “c1” and “c2” such that there is a relationship between them. When the value of “c1” changes, the following sequence of events would typically take place:

1. The controller identifies controls that are grouped with “c1” and finds “c2”.
2. The controller computes the new value of “c2” based on the relationship between “c1” and “c2”.
3. The controller creates an OSC bundle with two OSC messages, one for OSC method “x” and another for OSC method “y”. The OSC message for method “x” contains an argument that sets “p1” to the value of “c1”, while the OSC message for method “y” contains an argument that sets “p2” to the value of “c2”.

Points of Consideration

Parameter grouping in the manner described above is tightly coupled to a single controller. Therefore, this method of managing parameter grouping is not ideal for environments with multiple controllers, where all controllers must maintain consistent parameter relationships. Figure 3.3 illustrates a scenario where an additional controller (controller 2) is added to the configuration shown previously in Figure 3.2, where the additional controller has controls “d1” and “d2” associated with methods “x” and “y” of the device, respectively. In this configuration, controller 1 maintains relationships between its controls, “c1” and “c2”, while controller 2 maintains relationships between its controls, “d1” and “d2”. The management of relationships by the

controllers makes it possible for the two controllers to create conflicting relationships between controls that are associated with the same underlying device parameters. In order to prevent such conflicts, additional synchronisation is required between all controllers that are associated with the same underlying device parameters. In addition, further synchronisation is required in order to ensure that changes in values of device parameters are mirrored in all associated controls.

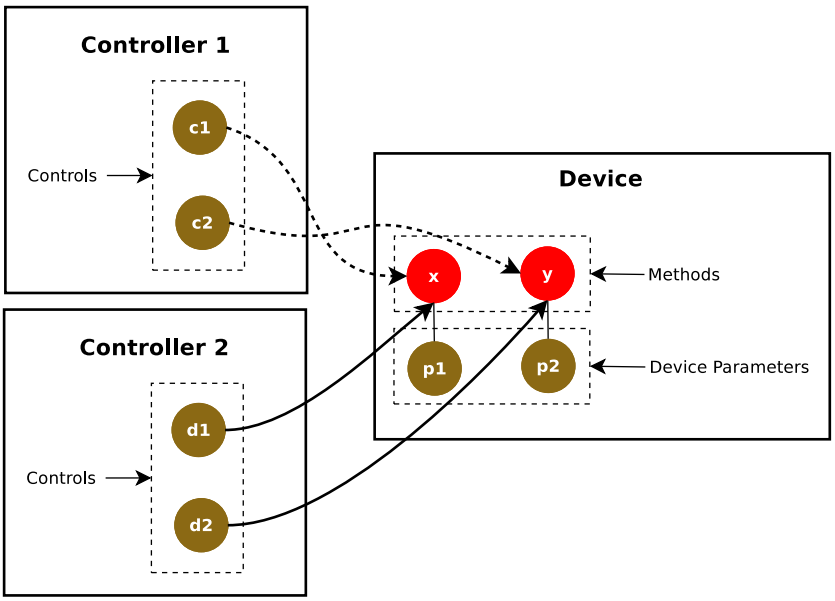


Figure 3.3: Device Parameters Mapped to OSC Methods with Two Controllers

3.1.2.2 Alternative 2: Grouping Managed by a Device

In this alternative, groups and relationships are stored in the devices. Similar to alternative 1 described above, each parameter is mapped to an OSC method and each OSC method is associated with a control on the controller. However, an additional configuration section exists within the OSC address space as shown in Figure 3.4. This configuration section has OSC methods that are used to create relationships using the classification scheme that was derived from this study, namely:

- Create master-slave groups ("/config/createMS" method).
- Create peer-to-peer groups ("/config/createPTP" method).
- Delete groups ("/config/deleteGroup" method).

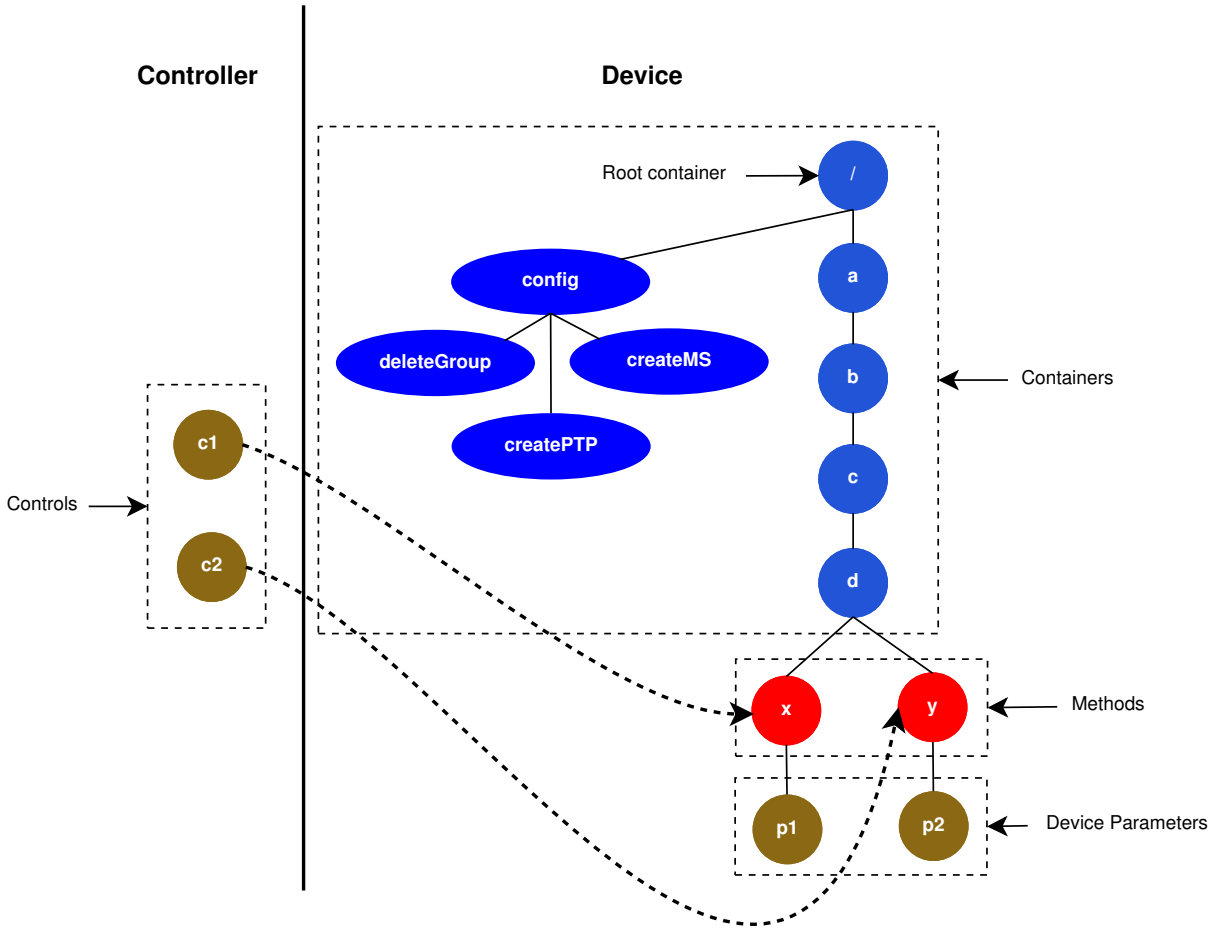


Figure 3.4: OSC Address Space with Methods to Create and Delete Groups

The controller is responsible for creating and deleting the groups. When a parameter value is updated, the associated OSC method first checks if the parameter is part of a group. If the parameter is not part of a group, the parameter value is simply updated. However, if the parameter is part of a group, further group processing is done. Group processing involves:

1. Retrieval of the network address and OSC address for each parameter in the group. The network address can be, for example, the IP address of a device that hosts the given parameter. Keeping track of the network address makes it possible for parameters on different devices to be grouped.
2. Calculation of the new value for each of the parameters in the group based on its relationship with the one that is being updated.
3. Setting of the new value of each of the parameters in the group using a combination of its

network address and OSC address.

Points of Consideration

It is possible for multiple vendors to implement their application-specific parameter grouping mechanisms within devices. However, the use of such application-specific implementations potentially result in interoperability challenges, particularly when multiple implementations exist. In order to prevent such interoperability challenges, parameter grouping mechanisms must be defined at the protocol level.

3.1.3 Parameter Monitoring Capabilities within the OSC Protocol

No explicit parameter monitoring capabilities are defined for the OSC protocol. However, it is possible for controllers to monitor the statuses of parameters by polling the parameter statuses at regular intervals using standard OSC messages.

3.2 IEC 62379

The IEC 62379 set of standards defines a control framework for networked multimedia devices [International Electrotechnical Commission, 2007]. The control framework was originally developed for radio broadcast and later extended to video and other time-critical media. Part 1 of the IEC 62379 set of standards describes multimedia devices as being composed of multiple interconnected functional blocks. A number of standard functional blocks are defined in the various parts of the set of standards, although it is also possible to define new vendor-specific functional blocks. The simple network management control protocol (SNMP) is used for the configuration and monitoring of multimedia devices. This section begins by giving a brief overview of SNMP. An overview of the architecture of IEC 62379 and its use of SNMP will be given. Possible extensions, proposed by Eales [2012], that allow for some level of parameter control grouping relationships will be described. We also look at the parameter monitoring capabilities that are inherent within the control framework.

3.2.1 SNMP Overview

SNMP is a network protocol that allows for control and monitoring of networked devices [Case, Fedor, Schoffstall, and Davin, 1990]. There are three main components within SNMP managed networks, namely:

- Managed device (also known as network element)
- Management agent
- Network management station (also referred to as manager)

Managed devices are devices that are controlled and/or monitored on a network. Each managed device runs software, referred to as the management agent, that handles requests from network management stations. Network management stations execute applications that control and monitor managed devices. SNMP is used for communication between network management stations and management agents. SNMP defines management operations in terms of reads and writes on a hierarchically organised collection of managed objects known as a *management information base* (MIB) [International Electrotechnical Commission, 2007]. Each managed object is uniquely identified by an *object identifier* (OID) that is represented by a sequence of numbers separated by dots. For example, OIDs of objects defined by the IEC 62379 standard begin with "1.0.62379.p", where "p" is the part number of the standard as shown in Figure 3.5. For completeness, suffice to mention that product-specific OIDs can be defined at the following hierarchy level "1.3.6.1.4.1.n", where "n" is the enterprise number of the manufacturer of the product.

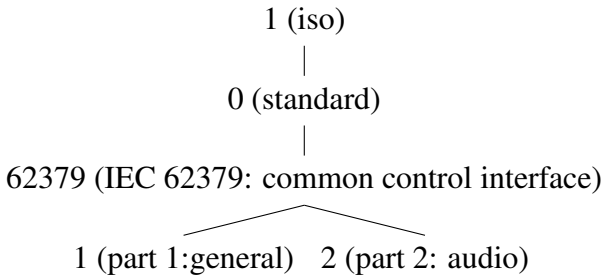


Figure 3.5: Hierarchical Representation of IEC 62379 OIDs

3.2.2 IEC 62379 Architecture

3.2.2.1 Block Framework

The IEC 62379 control framework models multimedia devices as *units* [International Electrotechnical Commission, 2007]. Each unit contains a number of functional elements known as *blocks*. Blocks may be composed of inputs, outputs, and some internal functionality. An output of one block is typically connected to an input of the next block in a processing chain. Each block can be associated with some control parameters and status monitoring capabilities that allow for connection management, control, and monitoring. The control framework defines two main tables for each unit, namely a table of blocks contained in the unit (block table) and a table of connections that exist between the various blocks (connectivity table).

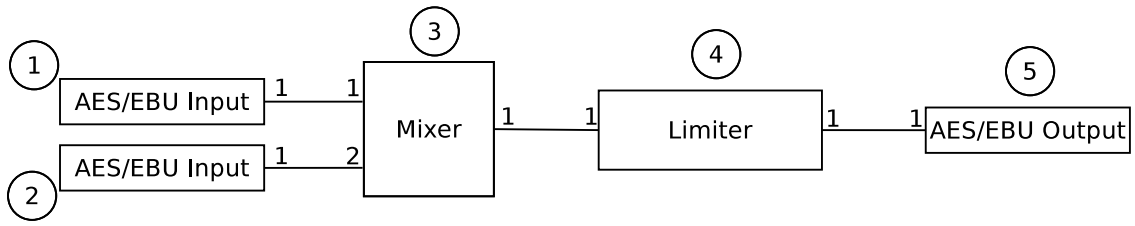


Figure 3.6: Model of a Simple Audio Device

Figure 3.6 shows the model of an audio device as adapted from the International Electrotechnical Commission [2008]. The figure shows five blocks with block ids 1 to 5 (the circled numbers), where:

- Blocks 1 and 2 are AES/EBU audio input ports that receive audio from an external AES/EBU audio source.
- Block 3 is a mixer block that receives the output from both AES/EBU audio input ports (blocks 1 and 2).
- Block 4 is a limiter block that receives the output from the mixer block (block 3).
- Block 5 is an AES/EBU audio output port that receives the output from the limiter block (block 4) and sends it to an external AES/EBU audio receiver.
- The connections between the blocks are as follows:
    - Output 1 of block 1 is connected to input 1 of block 3.

- Output 1 of block 2 is connected to input 2 of block 3.
- Output 1 of block 3 is connected to input 1 of block 4.
- Output 1 of block 4 is connected to input 1 of block 5.

Each block has a set of objects that model its control and status parameters. These objects are represented by one or more optional tables that are associated with each of the blocks. As an example, we look closer at the limiter block shown in Figure 3.6. The limiter block is described by a limiter block table (*aLimiterBlockTable*) with the OID of 1.0.62379.2.1.5.1. Since there is only one limiter block in the example, the limiter block table has only one entry (*aLimiterBlock-Entry*). Table 3.1 shows the layout of the limiter block table that belongs to the root OID of 1.0.62379.2.1.5.

.aLimiterTable.aLimiterBlockEntry (.1.1)					
aLimiter BlockId (1)	aLimiter Threshold (2)	aLimiter AttackTime (3)	aLimiter GainMakeup (4)	aLimiter RecoveryTime (5)	aLimiter RecoveryMode (6)
4					

Table 3.1: Limiter Block Table Layout [International Electrotechnical Commission, 2008]

The equivalent hierarchical representation of the limiter block table is shown in Figure 3.7. In order to control a parameter of a block, management applications typically set the value of the parameter using the parameter's OID. For example, setting the threshold parameter in the simple example described above is achieved by setting the value of the object with the OID of 1.0.62379.2.1.5.1.1.2.4

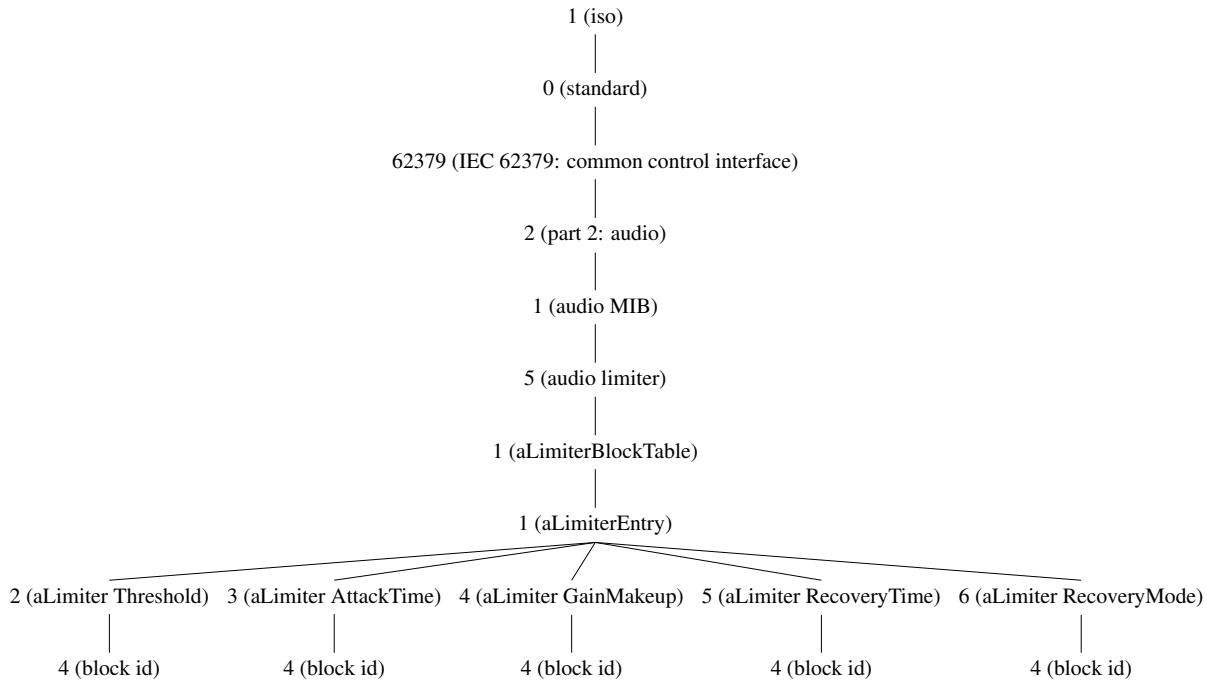


Figure 3.7: Hierarchical Layout of the Limiter Block Table

3.2.3 Parameter Grouping Capabilities within IEC 62379

The IEC 62379 set of standards does not explicitly define any grouping capabilities between the control parameters within the various blocks. However, Eales [2012] proposes some extensions that allow for grouping relationships between blocks. We now describe the nature of these extensions.

3.2.3.1 Extensions for Parameter Grouping Relationships within IEC 62379

Eales [2012] proposes the creation of grouping relationships between blocks as opposed to individual control parameters. In order to make this possible, an additional group identifier entry is required within the block table. The group identifier entry indicates the group membership of each block. Table 3.2 shows an example with four volume controls, where:

- Each of the volume controls belongs to one of four blocks with block ids ranging from 1 to 4.
- Volume controls on blocks 1, 2, and 3 are assigned to group 1.

- The volume control on block 4 is not assigned to any group, hence a group id of 0.

It should be noted that the OID of 1.0.62379.2.1.10 referred to in Table 3.2 is a recommendation from Eales [2012] that is currently not officially defined by the IEC 62379 set of standards.

Block Id	**Block Type OID**	***Group Id***
1	1.0.62379.2.1.10 (Volume)	*1*
2	1.0.62379.2.1.10 (Volume)	*1*
3	1.0.62379.2.1.10 (Volume)	*1*
4	1.0.62379.2.1.10 (Volume)	*0*

Table 3.2: Block Table with Group Identifier [Eales, 2012]

In order to describe the nature of relationships within each of the groups, Eales [2012] defines a relationship table. Eales [2012] classifies relationships based on the scheme that was derived from this study. This scheme will be described in Chapter 4, where master-slave and peer-to-peer relationships are further classified as either relative or absolute. The relationship table defines two types of relationships, namely master-slave and peer-to-peer relationships. The relationship table associates each group id with a master block id, where:

- A master block id that is greater than zero indicates that the group is a master-slave group, where the block with a block id corresponding to the master block id is the master of the group.
- A master block id of zero indicates that the group is a peer-to-peer group.

Table 3.3 is an example of a relationship table, where group 1 is a master-slave group with block 3 as master, while group 2 is a peer-to-peer group.

Group Id	**Master Block Id**
1	3
2	0 (peer)

Table 3.3: Relationship Table [Eales, 2012]

A relationship type table is used to specify more complex relationships, such as whether or not the relationship between two blocks is absolute or relative. Table 3.4 is an example of a relationship type table that describes an absolute relationship between blocks 3 and 4, and a relative relationship between blocks 2 and 3.

Group Id	Block Id	Block Id	Relationship Type
1	3	4	1 (absolute)
1	2	3	2 (relative)

Table 3.4: Relationship Type Table [Eales, 2012]

Points of Consideration

The level of granularity achieved by grouping at the block level is limited, particularly when considering blocks containing multiple parameters, such as the limiter block illustrated in Figure 3.7 on page 42. Figure 3.8 illustrates an example of parameter relationships that are not possible with the grouping mechanism proposed above. The figure shows two blocks, block 1 and block 2. Block 1 contains two parameters, A and B, while block 2 contains two parameters, X and Y. Using the proposed grouping mechanism, only one relationship is possible between block 1 and block 2. It is not possible to create relationships between individual parameters of the block, for example, between parameter A and parameter X, and between parameter B and parameter X. The need for a grouping mechanism at the parameter level is evident.

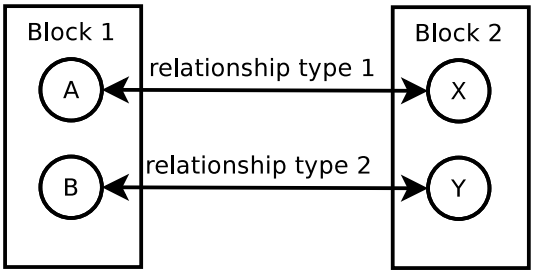


Figure 3.8: Limitations with Grouping at the Block Level

3.2.4 Event-Driven Parameter Monitoring Capabilities within IEC 62379

The IEC 62379 set of standards defines a mechanism for monitoring multimedia devices and their blocks through the transmission of *status pages* [International Electrotechnical Commission, 2007]. Status pages are messages composed of structured values that represent the internal state of a multimedia device. Multimedia devices may support multiple types of status pages, where each status page has a predefined format. Status pages contain information that either relates to a multimedia device as a whole or to particular blocks within the unit. Related status

pages are clustered together into status broadcast groups, where each status broadcast group is uniquely identified by an OID.

Part 2 of the IEC 62379 set of standards defines three status page groups that are supported by audio devices, namely audio ports (1.0.62379.2.3.1), standard audio blocks (1.0.62379.2.3.2), and audio alarms (1.0.62379.2.3.3). Figure 3.9 shows a hierarchical layout of these three page groups. Each of the page groups contains one or more status pages. For example, the audio ports page group comprises two pages, namely an audio port page (page 1) and an AES3 ancillary data page (page 2). Each status page has a predefined format. For completeness, Table 3.5 shows the format of the audio ports page that is part of the audio ports page group. The audio ports page monitors the various audio channel peak levels in a given block.

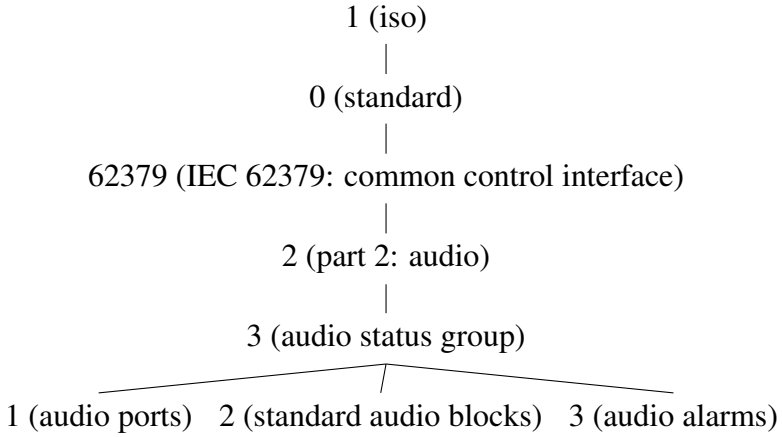


Figure 3.9: Hierarchical Layout of Audio Status Groups [International Electrotechnical Commission, 2008]

Octets	**Description**
1..2	Page number
3..4	Block identifier
5..8	Audio format
9..10	Channel 1 peak level
...	repeat previous entry for each remaining channel

Table 3.5: Status Entries for Audio Port Page [International Electrotechnical Commission, 2008]

Control applications on a network typically configure the transmission of status broadcasts by sending appropriate management commands to multimedia devices. These management commands contain the following information:

- The block id of the block for which status broadcasts are required (this value is null if the status broadcasts relate to the multimedia device as a whole).
- The page group OID of the status broadcast group that the control application is interested in.
- The rate at which the status pages are to be transmitted by the multimedia device.

Multiple control applications can receive monitoring information contained in status pages of the same page group on a given multimedia device, since the status pages are broadcast on the network. The clustering of related notifications into status pages reduces the network load in the event of changes in multiple related parameters, since one status page containing multiple notifications is sent, as opposed to a network message per parameter notification.

Points of Consideration

The layout of status pages as shown in Table 3.5 contains a series of position-specific octets that represent various attributes of parameters being monitored. This implies that client applications receiving the status pages must have prior knowledge of the status page layout, since status pages do not contain meta data that describes their layout. When monitoring devices from multiple vendors, where each vendor defines their own set of monitored parameters, a generic mechanism for "learning" the layout of the status pages is required in order for client applications monitor remote devices without requiring prior knowledge of remote devices.

3.3 Audio Video Control (AV/C) Protocol

The AV/C protocol allows for control over audio/video devices on an IEEE 1394 bus [1394 Trade Association, 2001]. IEEE 1394, commonly known as *firewire*, is a high-performance serial bus standard that supports two forms of data transfer, namely isochronous and asynchronous data transfers [Anderson, 1999]. Isochronous transfers ensure that data is transmitted at a constant rate, at 125μs intervals, without guaranteed delivery. In contrast, asynchronous transfers provide a means of data transmission with guaranteed packet delivery. Isochronous transfers are used for the transmission of real-time audio and video data, while asynchronous transfers are used for the transmission of control data. This section begins with a description of the *function control*

protocol (FCP), followed by an overview of the AV/C protocol. FCP uses the asynchronous data transfer capabilities of IEEE 1394. The AV/C protocol uses FCP as its underlying data transfer mechanism. The standard model of an AV/C device will be described, where an AV/C device is modelled as an AV/C unit that contains a number of AV/C subunits, input plugs, and output plugs. With knowledge of the model of an AV/C device, the nature of commands defined for the AV/C protocol will be discussed. The section ends with a discussion of the parameter control grouping and parameter monitoring capabilities that are possible within the AV/C protocol.

3.3.1 Function Control Protocol (FCP)

The IEC 61883-1 specification defines a *function control protocol* (FCP) that is used to control devices on an IEEE 1394 bus [International Electrotechnical Commission, 2003]. The AV/C protocol uses FCP as its underlying data transmission protocol as described in Section 3.3.2 to follow. FCP classifies IEEE 1394 nodes that control other nodes as *controllers*, while nodes that are controlled by others are classified as *targets*. Communication between controllers and targets makes use of asynchronous write transactions that are defined for IEEE 1394. Each asynchronous packet encapsulates an FCP *frame*. Frames that are sent from a controller to a target are known as *command frames* and are addressed to a *command register* that is implemented by the target. Frames that are sent from a target to a controller are known as *response frames* and are addressed to a response register that is implemented by the controller. Figure 3.10 shows an example of a controller node (node A) and target node (node B) on an IEEE 1394 bus, where node A sends a command frame to the command register of node B, while node B responds by sending a response frame to the response register of node A.

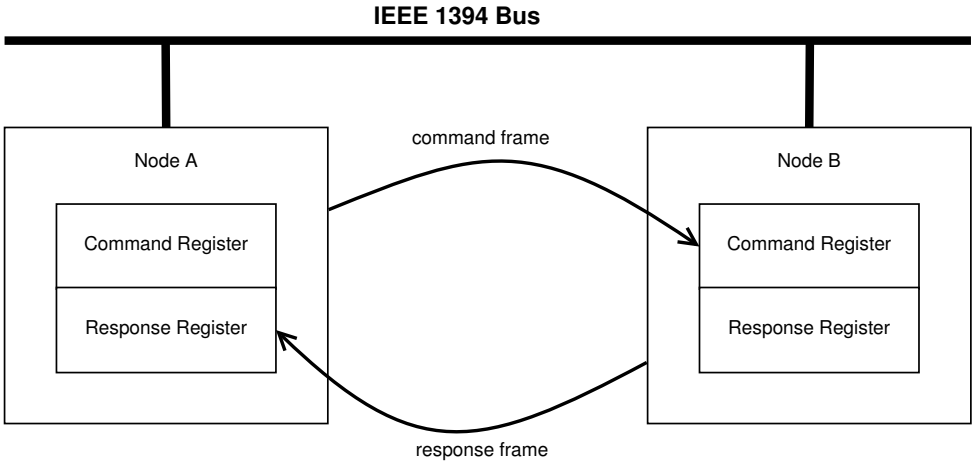


Figure 3.10: FCP Command and Response Registers as adapted from International Electrotechnical Commission [2003]

3.3.2 AV/C Protocol Overview

As mentioned earlier, the AV/C protocol is a control protocol for IEEE 1394 devices. We now provide an overview of the model of an AV/C device and some of the commands that can be used to read and/or modify parameters on a device. Control grouping and parameter monitoring capabilities that are supported by the AV/C protocol will also be described.

3.3.2.1 AV/C Model

An AV/C device is modelled as an *AV/C unit* that is composed of one or more *AV/C subunits*, *input plugs* and *output plugs* [1394 Trade Association, 2001]. Figure 3.11 shows an illustration of this model, where there is an AV/C unit containing two AV/C subunits and a number of AV/C unit plugs. The AV/C unit plugs are either input or output plugs, where input plugs receive data into the AV/C unit, while output plugs send data out of the AV/C unit. There are three main types of AV/C unit plugs, namely *serial bus isochronous plugs*, *serial bus asynchronous plugs*, and *external plugs*. Serial bus isochronous plugs are virtual connection end points on the AV/C unit that are used to transmit isochronous data, such as audio and video, to and from the AV/C unit. Serial bus asynchronous plugs are virtual connection end points on the AV/C unit that are used to transmit asynchronous data, such as control data, to and from the unit. External plugs are used to transfer data between the AV/C unit and other media apart from IEEE 1394.

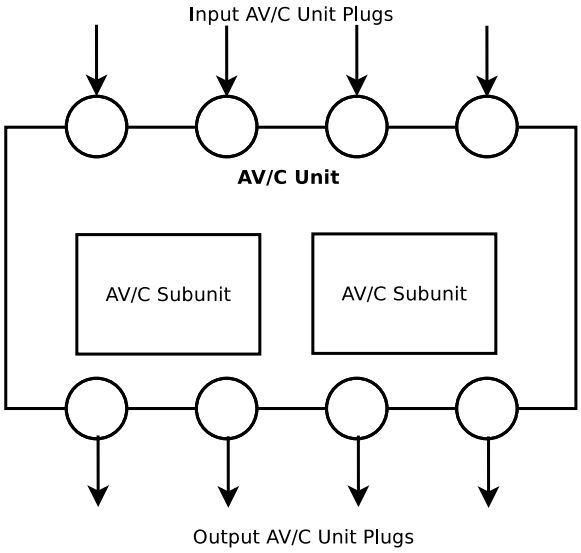


Figure 3.11: AV/C Unit Model

The AV/C subunit is composed of *source plugs*, *destination plugs*, and optionally one or more *function blocks* as shown in Figure 3.12. Function blocks of the AV/C subunit process input signals received by the subunit destination plugs. The processed signals are sent out of the AV/C subunit via the subunit source plugs.

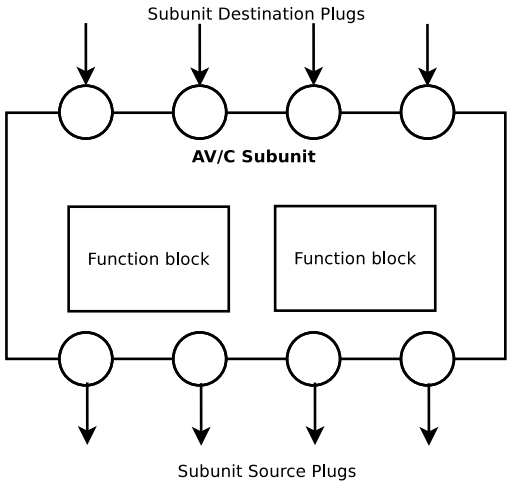


Figure 3.12: AV/C Subunit Model

3.3.2.2 AV/C Commands and Responses

AV/C makes use of FCP as the underlying data transmission mechanism. AV/C command and response frames are encapsulated within FCP frames. All AV/C frames, command frames or response frames, contain, among other fields, a *subunit type*, a *subunit id*, and an optional number of *operands*. The combination of subunit type and subunit id identifies an AV/C unit or a particular subunit within an AV/C unit, while the operands provide command- or response-specific meta data. AV/C command frames are typically sent by a controller and contain an additional *command type code* that indicates the command being invoked. There are five command types defined for AV/C as shown in Table 3.6.

Command Type	**Description**
CONTROL	Used to request an operation to be performed on the target.
STATUS	Used to retrieve the current status of a device.
SPECIFIC INQUIRY	Used to query if a target supports a given CONTROL command, with a specified number of operands.
NOTIFY	Used to receive notifications when the state of a device changes.
GENERAL INQUIRY	Used to query if a target supports a given CONTROL command, without specifying the operands.

Table 3.6: AV/C Command Types as adapted from 1394 Trade Association [2001]

Targets send AV/C response frames back to controllers upon receipt of AV/C commands. AV/C response frames contain an additional *response code* that is specific to the command that was invoked. There are seven response types defined for AV/C as shown in Table 3.7.

Response Type	Description
NOT IMPLEMENTED	Indicates that the target does not implement the specified command or subunit.
ACCEPTED	Indicates that the target is executing, or has executed, the command.
REJECTED	Indicates that the target implements the command specified but cannot execute the command due to, for example, the current state of the device.
IN TRANSITION	Indicates that the target implements the STATUS command, although it is in a state of transition.
IMPLEMENTED STABLE	For SPECIFIC INQUIRY and GENERAL INQUIRY commands, this indicates that the target implements the command. For STATUS commands, this indicates that the command has been executed and the response contains the current status of the device.
CHANGED	Indicates that the response frame contains a notification of a change in state of a device.
INTERIM	For CONTROL commands, this indicates that target cannot return the requested information within 100 milliseconds. For NOTIFY commands, this indicates that the target will notify the controller when the state of the target changes at a later time.

Table 3.7: AV/C Response Types as adapted from 1394 Trade Association [2001]

3.3.2.3 Parameter Grouping Capabilities within the AV/C Protocol

There are no explicit parameter grouping capabilities defined in the AV/C protocol. However, a controller is capable of managing groups of controls and making use of fundamental AV/C commands to update the target parameters. Figure 3.13 shows an example of a sequence of events that take place when a controller is managing groups. While the figure shows one target, it should be borne in mind that a controller typically interacts with more than one target. Here are the AV/C commands that are used for interaction between a controller and the associated targets:

1. The controller sends STATUS commands to retrieve the initial values of parameters within target devices on the IEEE 1394 bus.
2. Target devices respond with STABLE responses that contain the values of the parameters requested. A controller would typically associate each parameter on the target devices with

a control that resides in the controller. In addition to this association, a controller can also define relationships between its controls.

3. The value of a control changes, hence the controller calculates the new values for each of the controls that are grouped with the one that has changed value.
4. For each of the grouped controls that changes value, the controller sends CONTROL commands to modify the associated parameter value on the corresponding target device.
5. Each of the target devices receiving a CONTROL command modifies the parameter's value and responds with an ACCEPTED response indicating that the parameter value on the target device has been modified.

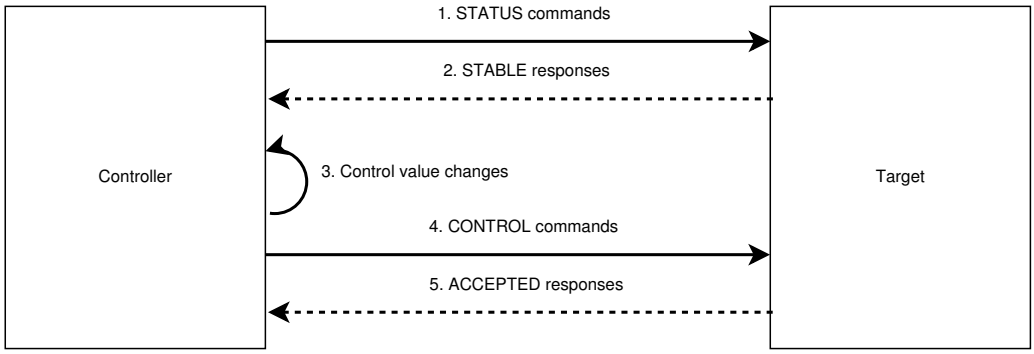


Figure 3.13: AV/C Control Grouping

Points of Consideration

Similar to the first alternative for parameter grouping within the OSC protocol, described in Section 3.1.2.1, this approach to parameter grouping is tightly coupled with a single controller. Thus, it does not guarantee consistency within networks that have multiple controllers, where controllers must maintain identical relationships for associated device parameters.

3.3.2.4 Polling and Semi-Event-Driven Parameter Monitoring Capabilities within the AV/C Protocol

Parameter monitoring is possible using the NOTIFY command [1394 Trade Association, 2001]. Figure 3.14 shows the main sequence of events that take place when a controller is interested in receiving a notification upon a target device's change in state, where:

1. The controller sends a NOTIFY command to request a notification when the state of a target device changes.
2. The target immediately responds with a INTERIM response that contains the current status of the target device.
3. The target device goes into an idle state or continues with other processing until the first change in state on the target device is registered.
4. The target sends a CHANGED response that contains the target's new status to the controller.

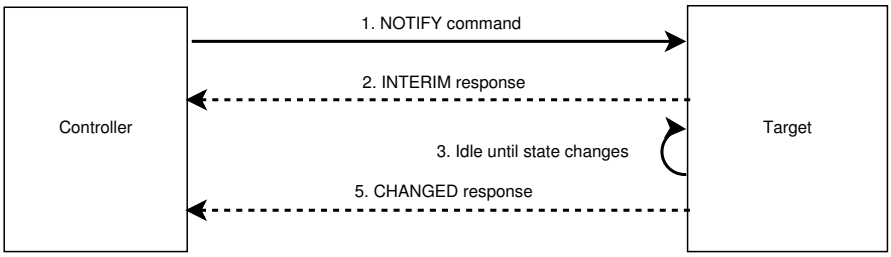


Figure 3.14: AV/C NOTIFY Command as adapted from 1394 Trade Association [2001]

As indicated above, after a NOTIFY command has been received successfully by a target device, the target notifies the controller of the first change in state. This mechanism does not allow for continuous updates from the device without controller intervention. Further monitoring of state can be done by repeating the cycle, where the controller sends another NOTIFY command. Therefore, this method of parameter monitoring is semi-event-driven.

The description above assumes successful execution of the NOTIFY command. However, execution of the NOTIFY command could fail, for example, when a target device does not implement the NOTIFY command or when a target device cannot handle NOTIFY commands from more than one controller simultaneously. In the case of the former, the target responds with a NOT IMPLEMENTED response, while in the latter case, the target responds with a REJECTED response. When a NOTIFY command fails, a controller can optionally start polling the target device status at regular intervals as shown in Figure 3.15, where:

1. The controller sends a NOTIFY command to request a notification when the state of a target device changes.
2. The target cannot execute the NOTIFY command successfully, hence responds with either a NOT IMPLEMENTED or REJECTED response.

3. Upon receiving a failure response from the target device, the controller starts a process that periodically:

   (a) sends a STATUS command to the target device, and

   (b) receives the target device's current status via a STABLE response to the STATUS command.

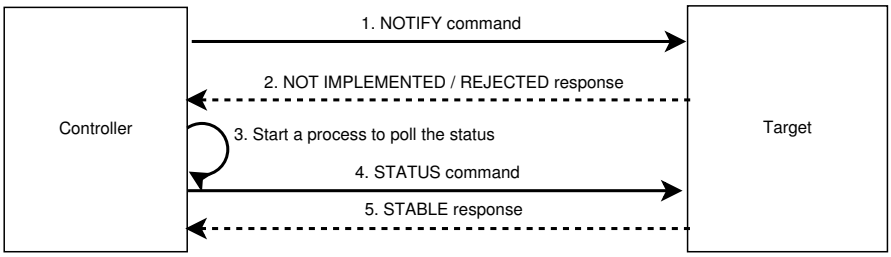


Figure 3.15: Polling Alternative to the NOTIFY Command as Recommended by 1394 Trade Association [2001]

Points of Consideration

The semi-event-driven parameter monitoring mechanism above is not ideal when monitoring constantly changing parameters, such as audio level parameters, where changes might be required at short intervals, for example, every 25 ms. There is additional network overhead due to the retransmission of the NOTIFY command upon receiving each change notification. However, the overhead resulting from the semi-event-driven parameter monitoring approach is less than that of the polling alternative to parameter monitoring, where there is constant network load regardless of the presence or absence of value changes.

3.4 Architecture for Control Networks (ACN)

ACN is a modular architecture for control protocols that was originally created for lighting control, although applicable to other areas such as multimedia networks [Entertainment Services and Technology Association, 2005]. ACN comprises a number of protocols and languages that can be put together to build network control systems. The protocols and languages include *root layer protocol*, *session data transport (SDT) protocol*, *data management protocol (DMP)*, *device*

description language (DDL), and *service location protocol (SLP)*. Figure 3.16 shows how these modules are combined together. A number of interoperability profiles have been defined to allow for communication between different modules. The SDT and root layer protocols are responsible for the packaging and transmission of ACN messages across various transport mediums, while SLP is used for device discovery. Parameter grouping and monitoring capabilities within ACN are enabled by the DMP and DDL.

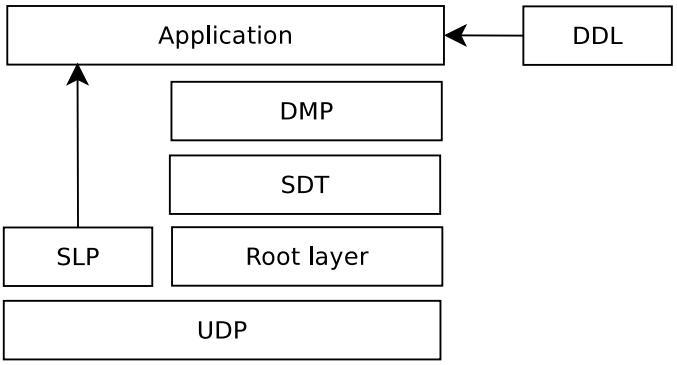


Figure 3.16: ACN Overview

Any distinct endpoint that transmits or receives ACN data is known as a *component*. The functions of any component are represented by a set of *properties*. The DMP defines messages to access and modify the values of properties, namely *Get_property* and *Set_property* messages, respectively [Entertainment Services and Technology Association, 2009c]. In addition, the DMP also defines an addressing scheme for the properties within components. The DMP is only responsible for accessing and modifying properties, without adding any semantics to the properties. Instead, semantics are added by DDL, which provides a mapping between properties and specific functions of components. DDL is an XML-based language that describes the properties within components using a combination of hierarchical tree structures and behaviours [Entertainment Services and Technology Association, 2009b]. Behaviours define the meaning of associated properties, and the actions and constraints that apply to the properties upon value changes. We now focus on the parameter grouping and monitoring capabilities that are possible within ACN.

3.4.1 Parameter Grouping Capabilities within ACN

Recall that behaviours within ACN specify how properties should respond to value changes. A core set of behaviours, known as the ACN base behaviour set, is defined within ACN [Entertain-

ment Services and Technology Association, 2009a]. The base behaviour set defines a *binding* behaviour that enables the creation of relationships between properties.

The binding behaviour connects two or more properties in such a way that changes in value of one property result in the same value being applied to all other properties. Bindings can be either network bindings or internal bindings. Network bindings commonly exist between properties that reside in separate devices, and thus require some network operations in order to synchronise property values. However, it should be borne in mind that it is possible to have network bindings between properties within the same device, where synchronisation of property values does not necessarily require a network operation. On the other hand, internal bindings are between properties that reside within the same device for all possible configurations of the binding, and thus do not require network operation to synchronise the bound properties.

For any two properties that are bound, one is referred to as the *anchor property*, while the other is referred to as the *remote property*. The anchor property is the permanent member of the binding relationship that keeps a reference to the remote property. Depending on the nature of the binding, both anchor and remote properties could reside in the same or different devices. Two mechanisms are defined for the synchronisation of values of bound properties, namely *push* and *pull* bindings. Synchronisation of bound properties is described relative to the anchor property. Within push bindings, whenever the value of the anchor property changes, the new value is "pushed" (sent) to associated remote properties. If the remote property resides on a different device from the one hosting the anchor property, the value of the anchor property is typically sent via a network write operation. Within pull bindings, when the value of the remote property changes, the same value is applied to the anchor property. If the remote property resides on a different device, the anchor property typically performs a network read operation of the remote property value, or alternatively, the remote property publishes its value upon every change.

Three main relationship types exist between bound properties, namely master/slave (or unidirectional), bidirectional, or multiway bindings. A description of each of these follows.

3.4.1.1 Master/Slave or Unidirectional Bindings

Push and pull binding mechanisms described above are both variants of unidirectional bindings. Here, changes in value of one property are applied to the other property. Therefore, this type of relationship is a master/slave relationship, since the value of the slave property always changes to

match the value of master property, while the value of the master property is independent. In the context of push bindings, the anchor property is the master property, while the remote property is the slave property. In contrast, with pull bindings, the anchor property is the slave property, while the remote property is the master property. Master/slave relationships described by push and pull bindings will be classified as *absolute master-slave relationships* in Chapter 4.

3.4.1.2 Bidirectional Bindings

Bidirectional bindings can include multiple properties. When the value of a bound property is changed by some external means, such as user action, the other bound properties are updated. Thus, values are both pushed and pulled between two bound properties. For example, when the value on anchor property changes, the same value is sent (pushed) to the remote property, while when the value of the remote property changes, the same value is fetched (pulled) by the anchor property. It is possible to have a group of properties that are all bound to each other in a bidirectional manner such that they all keep the same value. The relationship described by bidirectional bindings between properties will be classified as an *absolute peer-to-peer relationship* in Chapter 4.

3.4.1.3 Multiway Bindings

Multiway bindings comprise a group of unidirectional bindings, where one or more properties are involved in multiple bindings. A typical scenario would be a single master property that is bound to multiple slave properties, where the master property is the anchor property, while the bound slave properties are remote properties. Similar to unidirectional bindings, Chapter 4 will classify the relationships described by multiway bindings as *absolute master-slave relationships* between one master and a number of slaves.

Points of Consideration

Although ACN supports some parameter grouping relationships via the ACN base behaviour set, the extent of these relationships is limited. In particular, the only relationships that are supported by default are ones where one parameter's value is copied to another parameter (referred to as absolute relationships in Chapter 4). It is not possible to create relationships, where for example:

- Relative value differences between grouped parameters are maintained, such as when a gain parameter, *X*, is always 10 dB greater than another gain parameter *Y* (similar to some mixing console parameter grouping relationships).
- Hybrid relationships where master-slave and peer-to-peer relationships are used concurrently.
- There are relationships between parameters that have different underlying value units. Such relationships are applicable in, for example, show control systems, where the linking of control between parameters having different value units is required. Huntington [2007] contextualises show control systems as systems that connect two or more entertainment control systems, resulting in a global system that is composed of two or more sub-systems. Examples of entertainment control systems include fog machines, audio/video playback systems, and dimmer control consoles for lighting. Each of these entertainment control systems does not individually comprise a show control system. However, a show control system would, for example, link the control of an audio playback system with that of dimmer switches, effectively creating a relationship between light intensity and some audio property.

In order to support more complex relationships in ACN, additional application-specific binding behaviours are required. However, the lack of standardisation in application-specific parameter grouping mechanism potentially results in interoperability challenges, even between ACN devices from different vendors. A standard, comprehensive, parameter grouping mechanism is required at the protocol level in order to ensure that all devices implementing a given control protocol can rely on the same core set of parameter grouping capabilities.

3.4.2 Event-Driven Parameter Monitoring Capabilities within ACN

The DMP defines an event-driven mechanism for ACN controllers to monitor the state of properties within components as shown by the sequence of events in Figure 3.17 [Entertainment Services and Technology Association, 2009c]. An ACN controller typically sends a subscription request to a component. The subscription request contains a list of properties for which the controller should receive notifications upon value changes. The component sends a subscribe accept response to the controller, indicating a successful subscription. After the subscribe accept message has been sent, the component sends a sync event message to the controller. The sync

event message contains the current values of the properties that the controller has subscribed to. Sending of the current values ensures that the controller has the same initial value as the component. The component continues with other processing until any of the subscribed properties changes or upon events being triggered by the component. Consequently, the component sends event messages that contain the new property values to all subscribers.

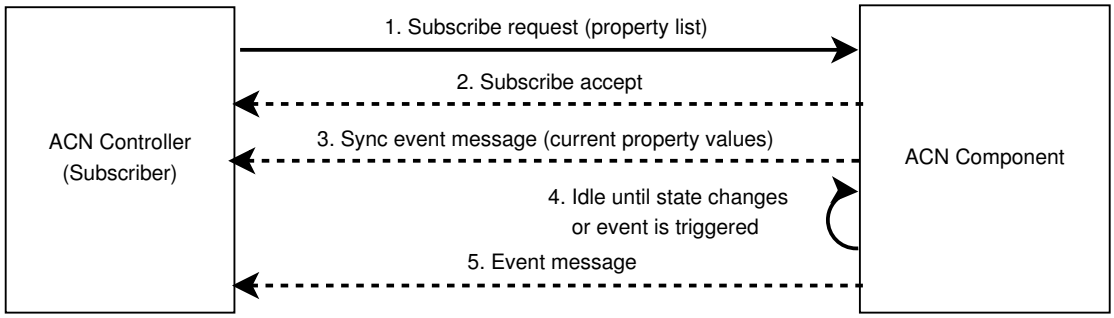


Figure 3.17: ACN Subscriber-Event Mechanism

In order to preserve bandwidth, events are not sent for properties without subscribers. However, there are some instances, where due to performance benefits, events can contain a range of properties that include properties that are not subscribed to. Additional properties may also be set in order to control a device's event publishing mechanism. For example, components can expose a property to control the maximum event publishing frequency, that is, the maximum number of events per second. Therefore, the subscriber-event mechanism enables remote metering with adjustable frequency. In addition, components could also expose properties that indicate the minimum change in a property's value that would cause an event to be published.

3.5 Open Control Architecture (OCA)

OCA is a transport-protocol-independent control and monitoring architecture for media networks that was formed by a group of professional audio companies [Berryman, 2013]. The features of OCA include device discovery, media stream connection management, control, and monitoring. OCA is currently undergoing standardisation.

OCA defines a number of protocols for different types of networks. OCA protocols are object-oriented, where communicating devices are described by objects that are addressable via unique

32-bit object numbers. These objects are instances of particular *OCA classes*. OCA classes are defined as nodes that are arranged in a tree structure by order of inheritance. The concept of inheritance is similar to that in programming languages, where specialised OCA classes are derived from generic parent classes. The *OcaRoot* class is the base class of all OCA classes. OCA classes comprise:

- A set of elements that define protocol exchanges between devices and controllers, namely properties, methods and events.
- A unique class identifier.
- Exactly one parent class, except for the OcaRoot class that does not have a parent.

Figure 3.18 shows the UML notation of an inheritance hierarchy for the *OcaGain* class that implements audio gain functionality. As shown in the figure, the OcaGain class inherits from the *OcaActuator* class, where the OcaActuator class inherits from the *OcaWorker* class, while the OcaWorker class inherits from the OcaRoot class. Descriptions of the purposes of worker and actuator classes are given in Section 3.5.1 on the following page.

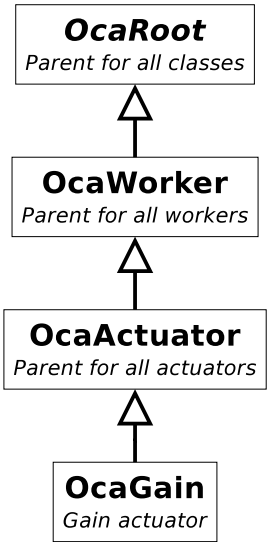


Figure 3.18: OCA Class Hierarchy Example

The remainder of this section gives an overview of the OCA device model, followed by descriptions of parameter grouping and monitoring capabilities within OCA. These descriptions are based on “The Open Control Architecture” [Berryman, 2013].

3.5.1 OCA Device Model

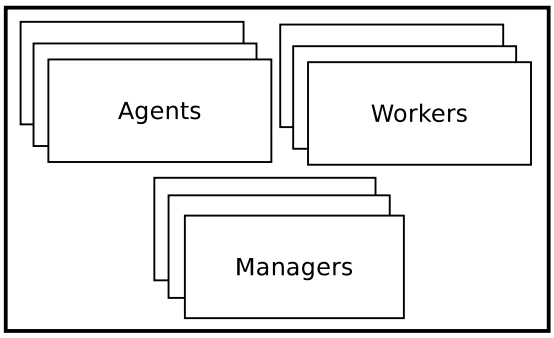


Figure 3.19: OCA Device Model: Supported Object Types

Figure 3.19 shows an overview of the OCA device model. As shown in the figure, an OCA-compliant device contains three main types of objects, namely *managers*, *workers*, and *agents*. The functions of these objects are:

- Managers
  Objects pertaining to the overall state of a device, such as firmware manager and subscription manager.
- Workers
  Objects pertaining to a device's application-specific functions, such as mute switches, gain controls, level sensors, image processing parameters, *et cetera*. Workers are further classified as *actuators*, *sensors*, *blocks*, *matrices*, and *networks*, where:
  - Actuators control application parameters, such as a switch.
  - Sensors detect and report the status of signal parameters to controllers.
  - Blocks enable the grouping of related objects into collections.
  - Matrices assemble objects into two-dimensional arrays.
  - Networks describe the digital networks to which a device is connected.
- Agents
  Intermediary objects that affect control actions within a device. For example, the *grouper* described in Section 3.5.2 on the following page implements parameter grouping relationships in a manner that is similar to VCA grouping in analogue mixing consoles. VCA grouping is described in Section 2.1.1.1 on page 10.

3.5.2 Parameter Grouping Capabilities within OCA

Parameter grouping capabilities within OCA are implemented by the *OcaGrouper* class. The OcaGrouper class is a specialised form of the agent class. An instance of the OcaGrouper class is referred to as a grouper. Groupers associate OCA objects in a manner that makes their properties controllable by a single value, where the group's value for a particular property is known as the group *setpoint*. It is possible for an object to belong to more than one group. For example, in a sound system, the left channel woofer amplifier gain might be controllable by a master gain group, a left-side gain group, and a woofer gain group. The grouper has knowledge of all groups that each object belongs to, as well as the *aggregation rules* that govern the manner in which value changes are applied to each group member. Thus, the grouper is a central relationship manager.

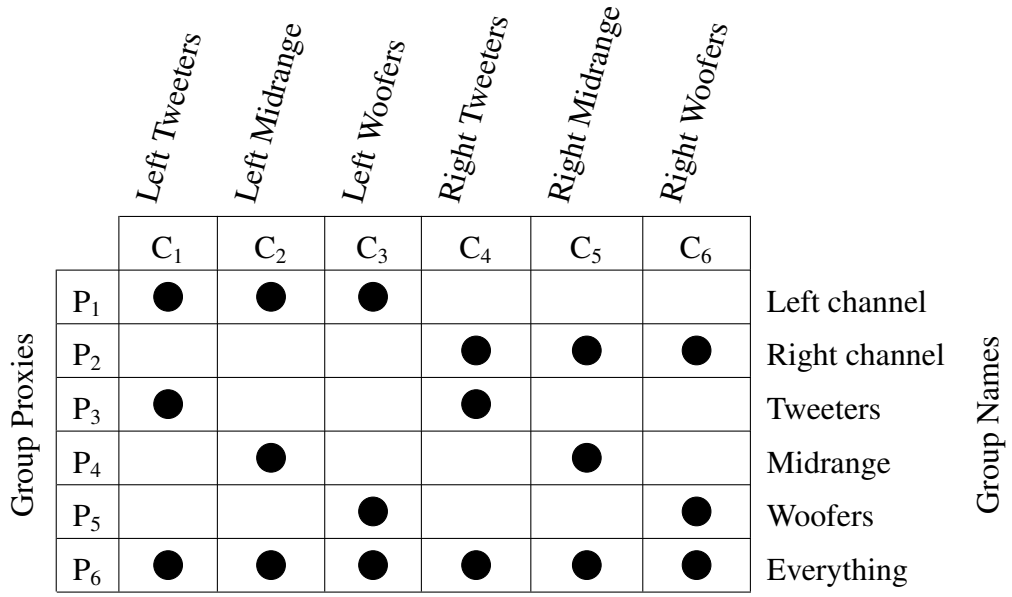
The following terms are used within the OCA grouping mechanism:

Citizen	An object that a grouper is aware of.
Group	A group that a grouper is aware of.
Enrolment	Associating a citizen with a group.
Member	A citizen that is enrolled in a group.

Figure 3.20 on the following page shows a matrix representation of an example grouper, where rows are groups, columns are citizens, and each crosspoint (cell) depicts the membership of the citizen to the corresponding group. Citizens that belong to a group can reside anywhere on the network. However, the citizens for a grouper should be instances of the same class. The setpoint of a group is only modifiable or accessible via a *group proxy* object. Thus, a grouper that manages *n* groups contains *n* group proxies. Group proxies are instances of the same class as the group's citizens.

The functions of a grouper include:

- Creating and deleting groups and their associated group proxies.
- Enrolment and de-enrolment of citizens to/from groups.
- Updating values of affected citizens upon changes in value of a group setpoint.
- Handling error conditions due to connection failures.



P_x = Proxy object *x*; C_y = Citizen object *y*; ● Enrolment

Figure 3.20: Grouper Example

Points of Consideration

Three shortcomings were identified in this approach to grouping, namely:

1. The restriction whereby all citizens must be instances of the same class implies that it is not possible to create relationships between different object types. For example, it is not possible to create relationships between audio parameters and light intensity parameters.
2. The grouper relies on the group proxy to update all citizens, potentially resulting in a single point of failure. Thus, if the group proxy is not reachable on the network, all grouping capabilities cease to function.
3. The nature of relationships that are possible between parameters is defined for each grouper in the form of aggregation rules. Consequently, there is no standard set of parameter grouping relationships that all parameters within a network can rely on, since it is possible for aggregation rules to vary from one grouper to another.

3.5.3 Event-Driven Parameter Monitoring Capabilities within OCA

OCA defines a subscription mechanism for parameter monitoring, where objects transmit property value update messages to other objects automatically and repetitively. Objects that transmit value update messages are known as *emitters*, and the messages that they transmit are referred to as *notifications*. Objects that receive notifications are referred to as *subscribers*. The subscription mechanism addresses three issues, namely:

1. Efficiency
   Subscriptions are classified as either reliable or fast, where reliable subscriptions transmit notifications via standard OCA protocol commands, while fast subscriptions transmit notifications using less reliable, but more efficient, means. In IP-based networks, connection-oriented protocols, such as the Transmission Control Protocol (TCP), are used for reliable subscriptions, while connectionless protocols, such as the User Datagram Protocol (UDP), are used for fast subscriptions.
2. Multiple subscriptions per emitter
   Emitters can be associated with more than one subscriber, making it possible to send notifications to more than one destination.
3. Abandoned subscriptions
   When a subscriber goes offline unexpectedly, emitters cancel all subscriptions for the corresponding subscriber.

Subscriptions result in the binding of an *event* to an *event handler*. An event is an object-specific method, while an event handler is an *OnEvent* method of an instance of the *OcaEventHandler* class. An event handler is owned by a subscriber. Subscriptions are created by a device's subscription manager object. The subscription mechanism sends notifications to event handlers upon occurrences of subscribed events. The transmission of notifications is dependent on one of three conditions, namely:

1. When a property value meets a specified criterion, such as exceeding a given threshold value.
2. At regular intervals, regardless of the property value.
3. A combination of the two above, at regular intervals but only when the property value meets a specified criterion.

Instances of the *OcaNumericObserver* class implement the three notification mechanisms listed above. Figure 3.21 shows an example of how a controller monitors the presence of a signal within a device. The property that stores the value of the signal level is implemented by an *OcaAudioLevelSensor* object. The following sequence is executed:

1. The OcaNumericObserver object reads the value of the property from the OcaAudioLevelSensor object.
2. The OcaNumericObserver object triggers the *Observation* event.
3. The *OcaSubscriptionManager* object sends a notification, across the network, to the subscriber (controller) by calling the OnEvent method of the subscriber's OcaEventHandler object.
4. The OnEvent method of the controller's OcaEventHandler handles the notification by, for example, illuminating an LED.

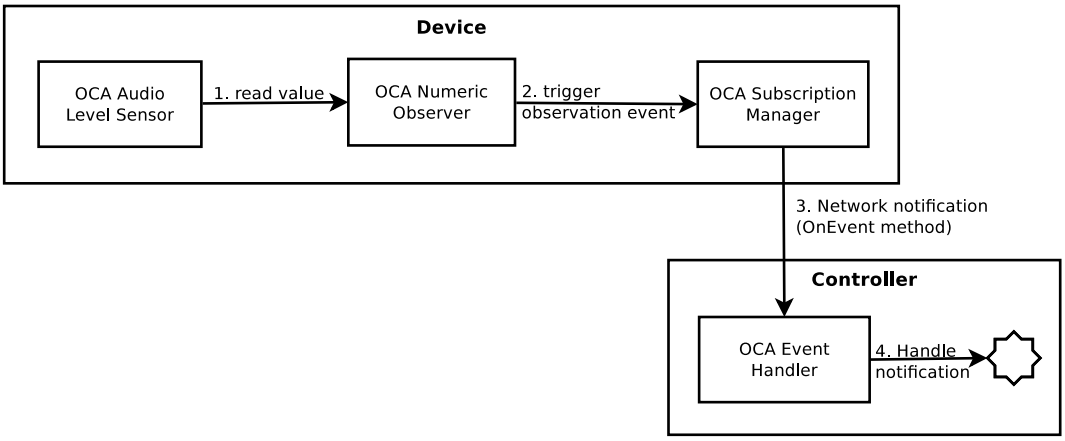


Figure 3.21: Subscription Mechanism Example

Points of Consideration

The subscription mechanism defined by OCA supports the transmission of notifications from individual objects upon value changes. There are scenarios when parameters, such as audio level parameters, change values at short intervals, for example, every 25 ms. When monitoring a large number of parameters that are constantly changing, it is more efficient to bundle value changes of multiple objects into a single batch notification message. Thus, an event-driven, bulk parameter monitoring mechanism is required for high-speed multimedia networks.

3.6 Chapter Summary

Five mainstream multimedia network protocols have been reviewed in this chapter, namely Open Sound Control (OSC), IEC 62379, Audio Video Control (AV/C), Architecture for Control Networks (ACN), and Open Control Architecture (OCA). The review focused on the parameter grouping and monitoring capabilities that are provided by each of the protocols. Table 3.8 summarises the capabilities that were identified in each of the protocols that were under review.

Protocol	**Parameter Relationship Capabilities**	
	Grouping	**Monitoring**
OSC	Application-specific	Polling
IEC 62379	Application-specific	Event-driven (status pages) Polling
AV/C	Application-specific	Semi-event-driven (notify command) Polling
ACN	Binding behaviours Application-specific behaviours	Event-driven (subscribe-event) Polling
OCA	OcaGrouper class	Event-driven (subscription mechanism)

Table 3.8: Summary of Parameter Grouping and Monitoring Capabilities

As shown in Table 3.8, ACN and OCA are the only protocols that define standard parameter grouping mechanisms. ACN implements parameter grouping using binding behaviours that are part of the ACN base behaviour set. However, the binding behaviours only allow for absolute relationships to be created between ACN properties. In order to allow for more flexible types of relationships within ACN, it is possible for application-specific behaviours to be defined. OCA implements parameter grouping using the OcaGrouper class. However, the capabilities of OcaGrouper class were found to be limited, since grouping is managed in a centralised manner and it is not possible to group parameters of different types. In contrast to ACN and OCA, OSC, AV/C, and IEC 62379 all do not define standard parameter grouping mechanisms within the protocols. Thus, application-specific grouping mechanisms are required to provide parameter grouping capabilities within these protocols. The use of application-specific parameter grouping mechanisms, as well as application-specific behaviours in the context of ACN, potentially results in the existence of multiple incompatible parameter grouping mechanisms within devices that implement the same protocol. Without defining a standard parameter grouping mechanism within a protocol, interoperability between devices from various manufacturers could become problematic, although possible. It is apparent that there is no comprehensive standards-based

mechanism for parameter grouping. Therefore, a need for a standard parameter grouping mechanism that encapsulates all common relationship types between parameters is evident.

In any of the protocols, parameters can be monitored by polling their statuses. However, the main disadvantage with polling is that it increases the load on the network and the amount of processing that is required by each of the devices being monitored. A controller typically requests the status of parameters at regular intervals, and responses to these polling requests could flood the network even when none of the parameter values have changed. AV/C defines a NOTIFY command that is used for asynchronous one-shot parameter value change notifications. In order to constantly monitor an AV/C device, the controller is required to send further NOTIFY commands to the device upon receiving each change notification. Thus, the NOTIFY command behaves in a similar way to polling although messages from the controller are triggered by the value changes on the target device. Given the one-shot nature of the AV/C NOTIFY command, such monitoring is not suited for parameters that change constantly, such as audio level parameters. IEC 62379, ACN, and OCA all define event-driven monitoring capabilities, where controllers typically send monitoring requests once, and the devices being monitored send all value change notifications thereafter. Unlike polling, event-driven monitoring bears the least network and device processing overhead, since change notifications are sent on demand. It is evident that event-driven monitoring creates a basis for monitoring within multimedia network protocols at large.

This study resulted in the implementation of parameter grouping and monitoring mechanisms. The mechanisms were implemented for the XFN protocol. Prior to this study the XFN protocol did not have any mechanisms for parameter grouping and monitoring. An overview of the XFN protocol will be given in Section 5.1 on page 79. The next chapter, Chapter 4, lays out general requirements for grouping and monitoring that are essential for the proper functioning of multimedia networks.

Chapter 4

Requirements for Parameter Relationships in Multimedia Networks

An analysis of parameter relationships that exist within audio mixing console implementations has been presented in Chapter 2. From this analysis, a number of core grouping relationship types that exist between parameters in mixing consoles have been introduced. Similarly, Chapter 3 presented an analysis of parameter grouping relationships that exist within some of the mainstream multimedia network technologies. In addition, parameter monitoring capabilities within the mainstream multimedia network technologies were reviewed. Based on the analyses of parameter relationships presented in Chapters 2 and 3, this chapter lays out a set of generic requirements for parameter grouping as well as a mechanism for real-time parameter monitoring within multimedia networks. The chapter concludes by listing the criteria for parameter relationships within networked multimedia environments.

4.1 Classification of Core Parameter Grouping Relationships

In various sections of Chapters 2 and 3, four phrases pertaining to relationship classification were referred to, namely *absolute peer-to-peer*, *relative peer-to-peer*, *absolute master-slave*, and *relative master-slave*. These phrases resulted from a classification scheme that was defined in the course of implementation to model the core relationships that exist between parameters within audio mixing console implementations and some existing multimedia network technologies. The

classification scheme identifies two relationship types that are further classified into two relationship sub-types as described below. The strategies for the implementation of the relationship classification scheme within the XFN protocol will be described in Chapter 5.

4.1.1 Relationship Types

Two main types of relationships were identified between multimedia parameters, namely peer-to-peer and master-slave relationships. Descriptions of each of these relationship types are given below. When referring to parameter relationships, a collection of parameters sharing a relationship is known as a *group*.

4.1.1.1 Peer-to-Peer Groups

Peer-to-peer groups comprise two or more peer parameters, where any parameter is capable of controlling all other parameters in the group. Hence, a change in value of one parameter causes a change in value of all other parameters. Figure 4.1 illustrates the nature of peer-to-peer relationships among three peer parameters, namely “a”, “b”, and “c”. In the context of a mixing console, for example, these parameters can correspond to the audio level parameters of three input channels. A change in value of each of these parameters causes a change in value of the other two, creating three alternatives, where:

- Any change in value of parameter “a” causes a change in value of parameters “b” and “c”.
- Any change in value of parameter “b” causes a change in value of parameters “a” and “c”.
- Any change in value of parameter “c” causes a change in value of parameters “a” and “b”.

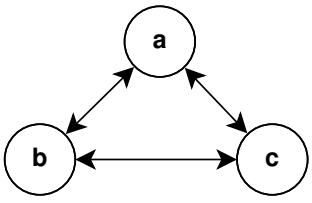


Figure 4.1: Peer-to-Peer Group Example

4.1.1.2 Master-Slave Groups

Master-slave groups comprise two or more parameters, where one parameter is a designated master parameter, while the remaining parameters are designated slave parameters. Within a master-slave group, the master parameter can control the slave parameters, while the slave parameters cannot control the master parameter. Figure 4.2 shows an example of a master-slave group containing three parameters, namely "a", "x", and "y". Here, parameter "a" is the designated master parameter, while parameters "x" and "y" are designated slave parameters. A change in value of the master parameter, "a", causes a change in value of both slave parameters, "x" and "y". In contrast, a change in value of either of the slave parameters, "x" or "y", does not result in a change in value of any other parameter. In the context of a mixing console, for example, parameter "a" can correspond to a master fader, while parameters "x" and "y" can be input/output channel faders being controlled by the master fader. Chapter 5 will describe the implementation of more complex variations of master-slave relationships, where it is possible for changes in one slave parameter value to influence value changes in other slave parameters.

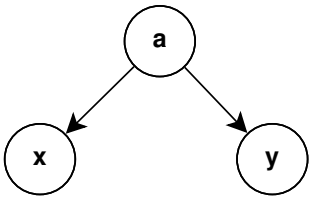


Figure 4.2: Master-Slave Group Example

4.1.2 Relationship Sub-Types

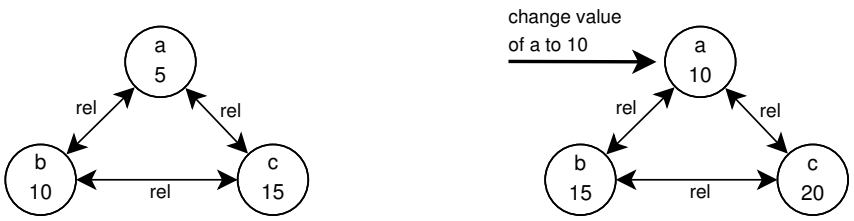
The relationship types described above both operate in such a way that a change in value is initiated in one parameter thereby causing a change in value of one or more other parameters by virtue of a relationship between the two parameters. For illustration purposes, we refer to the parameter where the value change is initiated as the *initiator parameter*, while the parameters that subsequently have their values changed will be referred to as *affected parameters*. Thus, in a peer-to-peer group, when one peer parameter (initiator) initiates a change in value, the rest of the peer parameters in the group become affected parameters. For master-slave groups, the master parameter is always the initiator, while the slave parameters are always affected parameters. Master-slave and peer-to-peer relationships described in Section 4.1.1 were further classified as either *relative* or *absolute*. A description of each of these two sub-types follows.

4.1.2.1 Relative Relationships

Relative relationships are ones where a change in value of the initiator parameter causes a change in value of the affected parameters such that a fixed offset is maintained between the value of the initiator parameter and each of the affected parameters. We now look at relative peer-to-peer and relative master-slave relationships with the aid of some illustrations.

4.1.2.1.1 Relative Peer-to-Peer Relationships

Continuing from the generic illustration of peer-to-peer relationships shown in Figure 4.1, Figure 4.3a shows parameters “a”, “b”, and “c” in a relative peer-to-peer relationship with initial values of, for example, 5, 10, and 15 units, respectively. Notice that the value of “b” is 5 units greater than that of “a” (also 5 units less than the value of “c”), and the value of “c” is 5 units greater than that of “b” (also 10 units greater than the value of “a”). Figure 4.3b shows how the values of parameters “b” and “c” are affected when the value of parameter “a” changes to, for example, 10 units. By virtue of a relative relationship the offsets between the values ought to be maintained. Hence, the new values of parameters “b” and “c” become 15 and 20 units, respectively. Again, notice that the original offsets between values are maintained, where the new value of “b” is 5 units greater than that of “a” (also 5 units less than the new value of “c”), and the new value of “c” is 5 units greater than that of “b” (also 10 units greater than the new value of “a”). It is important to note that whenever any of the peer parameters changes value, all the other peers are adjusted accordingly such that the offsets between values are maintained.

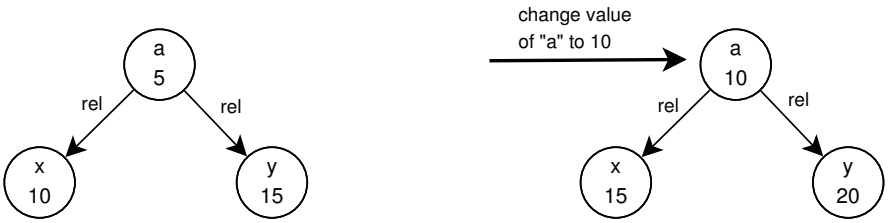


(a) Initial State (b) After Value Change of “a”

Figure 4.3: Relative Peer-to-Peer Relationships

4.1.2.1.2 Relative Master-Slave Relationships

Continuing from the generic illustration of master-slave relationships shown in Figure 4.2, Figure 4.4a shows parameters “a”, “x”, and “y”, where “a” is the master parameter, while “x” and “y” are slave parameters. The initial value of “a” is 5 units, while the initial values of the slave parameters “x” and “y” are 10 and 15 units, respectively. Here, we note that the master-slave relationships only exists between “a” and “x”, and between “a” and “y”, where the value of “x” is 5 units greater than that of “a”, while the value of “y” is 10 units greater than that of “a”. Figure 4.4b shows how a change in value of the master parameter (“a”) to, for example, 10 units, causes the slave parameter values to be adjusted such that the new values maintain the original offsets relative to the master parameter. In particular, the new value of “x” becomes 15 units, hence remaining 5 units greater than the new value of parameter “a”. The new value of “y” becomes 20 units in order to remain 10 units greater than the new value of “a”.



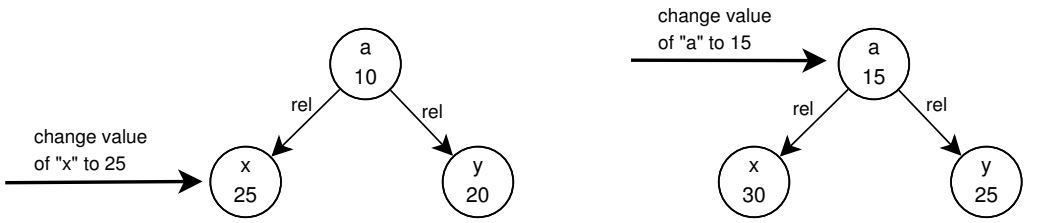
(a) Initial State (b) After Value Change of “a”

Figure 4.4: Relative Master-Slave Relationships

By definition of a master-slave relationship, changes in value of any of the slave parameters have no influence on the master parameter. However, in practice, it was found to be desirable for the offset between a master parameter value and a slave parameter value to be adjusted in the event that the slave parameter value is changed independently of its master parameter value.

Figure 4.5a continues from the illustration given in Figure 4.4b. Here, the slave parameter value of “x” is changed to, for example, 25 units. Since “x” is a slave parameter of “a”, this change neither influences the value of the master parameter (“a”) nor the values of any other slaves of “a” (“y” in this case). Now, if the value of “a” changes to, for example, 15 units, as shown in Figure 4.5b, the new value of “x” becomes 30 units, while the new value of “y” becomes 25 units. In practice, this situation typically exists when, for example, a master volume fader on an audio mixing console is required to control volumes of both the left and the right channels of a stereo pair of output audio channels. Here, it may be necessary for a sound engineer to

adjust the balance between the left and right channels individually, whilst retaining the relative master-slave relationships.



(a) Change in Slave Parameter Value

(b) Change in Master Parameter Value after Slave Changed

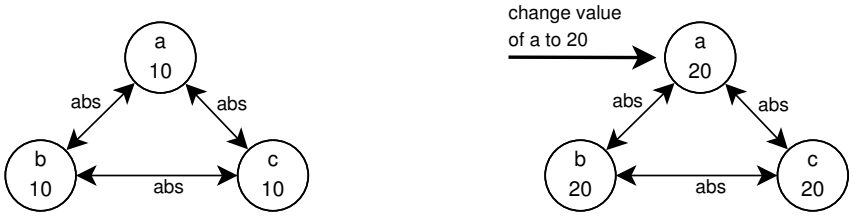
Figure 4.5: Handling Slave Parameter Value Changes in Relative Master-Slave Relationships

4.1.2.2 Absolute Relationships

Absolute relationships are ones where a change in value of the initiator parameter causes value changes of the affected parameters such that the affected parameters take on the new value of the initiator parameter. We now look at absolute peer-to-peer and absolute master-slave relationships with the aid of some illustrations.

4.1.2.2.1 Absolute Peer-to-Peer Relationships

With reference to the generic example shown in Figure 4.1, Figure 4.6a shows a similar arrangement of three parameters, namely “a”, “b”, and “c”, in an absolute peer-to-peer relationship with each other. It is important to note that when there is an absolute relationship between parameters, the associated parameters all share the same value. For example, Figure 4.6a shows parameters “a”, “b”, and “c” all with a value of 10 units. Whenever the value of any of the peer parameters changes, the values of the affected peer parameters are also changed to the new value of the initiator parameter. This is illustrated in Figure 4.6b, where a change in value of parameter “a” to, for example, 20 units, causes values of parameters “b” and “c” to be updated to 20 units as well.

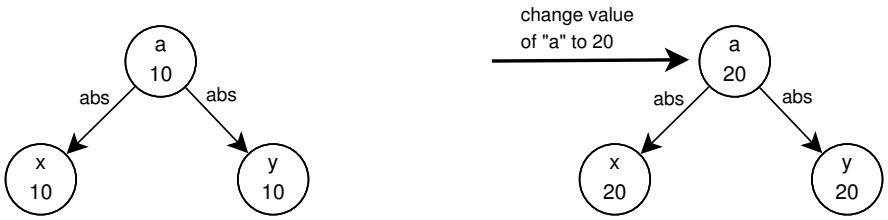


(a) Initial State (b) After Value Change of “a”

Figure 4.6: Absolute Peer-to-Peer Relationships

4.1.2.2.2 Absolute Master-Slave Relationships

Continuing from the illustration of master-slave relationships shown in Figure 4.2, Figure 4.7a shows an illustration of parameters “a”, “x”, and “y”, where parameter “a” is in an absolute master-slave relationship with both parameters “x” and “y”. Here, it is important to note that whenever the master parameter changes, both slave parameter values change to the same value as the master parameter. Figure 4.7a shows an initial state where all parameters have a value of 10 units. In the event that the value of the master parameter (“a”) is changed to, for example, 20 units, the values of both slave parameters “x” and “y” are also changed to 20 units as shown in Figure 4.7b. Independent changes in slave parameter values can occur. However, such slave parameter value changes only persist until the next change in value of the master parameter. Once again, it should be noted that independent changes in any slave parameter values do not cause changes in the value of the master parameter. Unlike in the case of the relative master-slave relationship described in Section 4.1.2.1.2, there is no need to keep track of the offset between the value of the master parameter and a slave parameter when the relationship is absolute, since slave parameters are always forced to take on the value of the master parameter.



(a) Initial State (b) After Value Change of “a”

Figure 4.7: Absolute Master-Slave Relationships

4.2 Relationships between Parameters on Different Devices

With reference to audio mixing consoles, Chapter 2 described how relationships can be created between parameters that reside in the same device. Chapter 3 then described how relationships can be created between parameters that reside either on the same device or on two or more separate devices within a network. Of the media network technologies reviewed, ACN and OCA are the only technologies that define mechanisms for relationships between parameters on different devices. ACN implements these using network binding behaviours, while the OCA implementation is based on the OcaGrouper class. However, the default binding behaviours defined in ACN are not comprehensive enough to cater for all possible types of relationships, thus supplementary application-specific behaviours are required. In addition, the relationships implemented by the OcaGrouper class are limited to parameters of the same type and are managed by a centralised node on the network.

A common feature of all the multimedia network technologies reviewed is the ability to provide remote control over parameters via reads and writes of their values. Using standard reads and writes to parameters, it is possible to create application-specific mechanisms to enable relationships between parameters on different devices. However, the lack of standardisation in the application-specific mechanisms required to manage the relationships was identified as a potential source of incompatibilities between separate devices, particularly devices from different vendors. It is evident that a standard mechanism for creating relationships between parameters on separate devices in a network is desirable. Such a mechanism must be incorporated in standard form into a protocol, such that all devices can depend on it.

4.3 Relationships between Parameters with Unrelated Units

A need for relationships between parameters that use unrelated value units was identified. Descriptions of the grouping capabilities given in the previous chapter involve grouping parameters with the same underlying value units, for example, relationships between volume parameters only. It is not possible to create relationships between parameters with different value units, such as relationships between light intensity and volume parameters, where luminous intensity is measured by the candela (cd) SI unit, while volume is measured using the decibel (dB) unit.

As a guiding principle, the ability of any parameter on the network to influence another parameter without necessarily knowing the underlying value units was found to be desirable. Such capability allows for the creation of customisable control applications that can interpret values sent by parameters without prior knowledge of the associated parameters. In the context of the XFN protocol, Chapter 5 will describe how this requirement was fulfilled through the definition of a common value unit.

4.4 Requirements for Parameter Monitoring

Two approaches to parameter monitoring have been described in Chapter 3, namely a polling approach and an event-driven approach. The polling approach bears more processing and bandwidth overhead. Thus, an event-driven approach was found to be more desirable. We now describe the requirements for an event-driven approach to monitoring with the aid of an example. Figure 4.8 shows a configuration with two networked control applications, where both are monitoring a single networked multimedia device's parameters. The typical sequence of events for monitoring would be as follows:

1. One or more control applications register to monitor a device's parameters.
2. The device keeps track of all registered control applications.
3. One or more monitored parameters in the device change in value, resulting in a message containing the new parameter values being sent to each of the registered control applications.
4. Upon receiving the latest parameter values, each control application console updates its cache and/or graphical representation of the values corresponding to the monitored parameters.
5. All registered control applications continue to receive value change notifications until they are deregistered from the monitored device.

In the context of the XFN protocol, Chapter 6 will describe an implementation of a *push* mechanism that provides parameter-oriented, event-driven, single-target or multi-target messaging. This push mechanism fulfils the requirements for parameter monitoring within multimedia networks as described in this section.

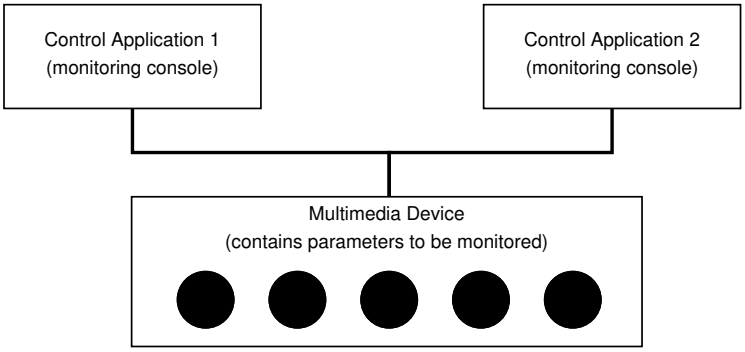


Figure 4.8: Example Monitoring Configuration

4.5 Chapter Summary

This chapter has described various requirements for parameter relationships within multimedia networks. These requirements are the basis for the remaining chapters, thus creating criteria against which implementations of parameter relationships in multimedia networks can be measured. The requirements have revealed the following key criteria:

1. The need for a standard network-wide grouping mechanism that allows for consistent parameter relationships across multimedia networks. This has been illustrated by the classification of core relationships as either master-slave or peer-to-peer. These relationships are further qualified as either absolute or relative.
2. The need for a standard mechanism to create relationships between parameters that reside in different devices.
3. The need for a mechanism to create relationships between parameters with unrelated value units. For example, a synchronised change of an audio level parameter and a light intensity parameter.
4. The need for a mechanism to allow for event-driven parameter monitoring.

The next chapter (Chapter 5) gives an overview of the XFN protocol. In the context of the XFN protocol, the chapter will describe the implementation strategies that meet criteria 1, 2, and 3 above. Chapter 6 will then describe the implementation strategies, for the XFN protocol, that meet criterion 4 above.

Chapter 5

Implementation Strategies to Fulfil Grouping Requirements

The previous chapter has laid out some key requirements for parameter grouping relationships across networks. The review of existing multimedia network technologies revealed that there is no existing multimedia network technology that fulfils all of these requirements. In the context of the XFN protocol, this chapter will describe a number of implementation strategies that were proposed in order to fulfil these requirements. The XFN protocol was chosen for this implementation, since it was undergoing standardisation within the Audio Engineering Society at time of this research; the AES-64 standard has now been approved. The chapter will begin with an introduction to the XFN protocol, followed by a detailed description of the proposed implementation strategies for parameter grouping relationships. These implementation strategies have been incorporated into the AES-64 standard. The proposed strategies are based on an implementation that was done in the course of this investigation; contributions from this study include:

- The implementation of a mechanism to manage relationships between XFN parameters, where each XFN parameter keeps lists of parameters that it has relationships with. A number of new XFN commands were added to facilitate management of these lists.
- The implementation of a mechanism to enable the relationships between parameters that have disparate value types using a common value unit known as the global unit. The global unit was also defined such that it is controllable by variable-sized controllers.

- The implementation of rules that enable determinism in the behaviour of grouped parameters.

The conceptualisation of these implementation strategies was done in consultation with Universal Media Access Networks GmbH and implemented in this study. The global unit was referred to as the "XFN unit" at the time of implementation. The name was changed to "global unit" upon recommendation by members of the AES standards working group SC-02-12 under project AES-X170.

The parameter grouping mechanisms implemented were applied to a graphical control application known as Unos Creator. These mechanisms were tested and proven in a laboratory environment. A description of the tests that were carried out will be given in Section 7.5.3 on page 208.

5.1 XFN Protocol Overview

XFN is a generalised IP-based protocol that integrates control, monitoring, and connection management for multimedia networks [Audio Engineering Society, 2012]. XFN is a peer to peer protocol, where each device can independently send or receive messages to or from other devices on the network. XFN is intended to provide an open-standards-based cross-vendor approach to connection management, control, and monitoring within multimedia networks.

The most basic building blocks of XFN are *XFN parameters*. XFN parameters are addressable via either fixed 7-level hierarchical addresses that are based on the functional layout of a device or via unique IDs. XFN parameters are associated with application-specific callback functions that allow for access and modification of application-specific parameters, such as DSP parameters within an amplifier. A number of XFN messages that contain various commands directed at XFN parameters are defined. These commands include, but are not limited to, commands to get and set parameter values.

Figure 5.1 shows a typical layout of an XFN device. All XFN devices implement an XFN protocol stack that is responsible for storing XFN parameters and processing messages directed at XFN parameters that reside in the local device or other devices on the network. In IP-based networks a device is identified by an IP address. It is possible for an XFN device to act as a proxy for other non-XFN devices, such as AV/C devices. In order to address multiple devices

that are under the control of a single XFN device, an XFN stack implements a number of XFN nodes as shown in Figure 5.1. Each XFN node contains 7-level hierarchies of XFN parameters. The 7-level hierarchies are terminated by XFN parameters that can each be addressable by a unique ID. Furthermore, XFN parameters contain application-specific callback functions that are responsible for accessing or modifying application-specific parameters that are modelled by the XFN parameters.

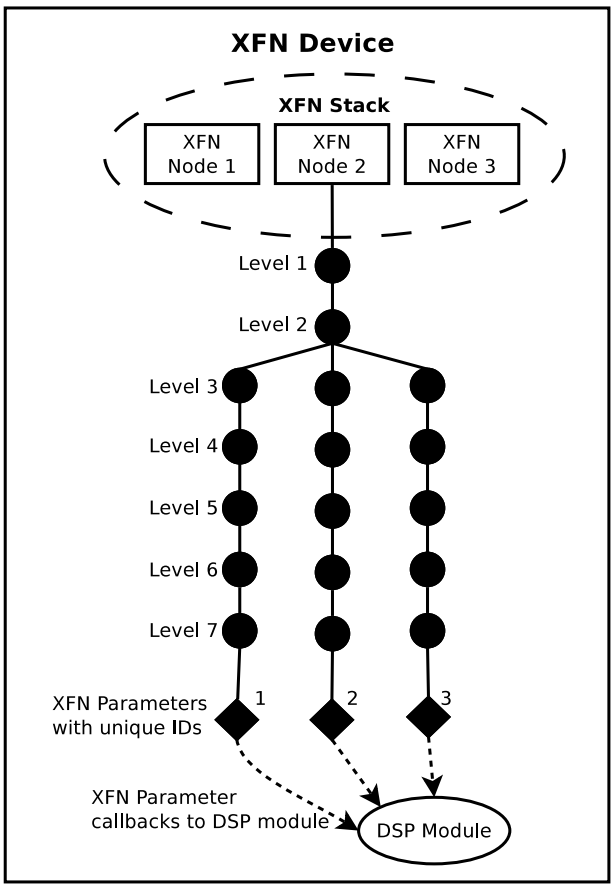


Figure 5.1: Typical XFN Device Layout

A detailed description of the XFN protocol is available in "AES64-2012: AES Standard for Audio Applications of Networks - Command, Control, and Connection Management for Integrated Media" [Audio Engineering Society, 2012]. Recall from Chapter 4 that four main requirements were identified, namely:

1. The need for a consistent parameter grouping scheme that resulted in the classification of core relationships, where relationships are either master-slave or peer-to-peer. These

relationships are further qualified as either absolute or relative.

2. The need for a standard mechanism to create relationships between parameters that reside in different devices.
3. The need for a mechanism to create relationships between parameters with unrelated value units.
4. The need for a mechanism to allow for event-driven parameter monitoring.

In the context of the XFN protocol, we now look at the implementation strategies that were developed to fulfil the first three requirements listed above.

5.2 Core Parameter Grouping Relationships

XFN parameters are the fundamental building blocks of the XFN protocol. Therefore, it was appropriate to implement relationships between XFN parameters. Thus, all the necessary parameter grouping information is kept within XFN parameters. Relationships between XFN parameters are referred to as *joins*. In order to implement the parameter grouping (join) scheme indicated in requirement 1 in Section 5.1, each XFN parameter keeps three parameter group lists, namely:

1. A peer group list that contains one or more entries that describe XFN parameters that are involved in peer-to-peer relationships with the parameter hosting the list.
2. A master group list that contains one or more entries that describe XFN parameters that are designated masters in master-slave relationships, where the parameter hosting the list is a designated slave.
3. A slave group list that contains one or more entries that describe XFN parameters that are designated slaves in master-slave relationships, where the parameter hosting the list is a designated master.

Each of the three lists indicated above contains entries that describe other XFN parameters that are involved in some relationship with the parameter that hosts the list. The information contained in each list entry may include the fields shown in Table 5.1, namely *device identifier*, *XFN node ID*, *XFN parameter ID*, *status*, *join type*, *offset*, and *flags*. The device identifier is a 32-bit

integer that uniquely identifies an XFN device on the network. In practice, the device identifier is typically mapped to either an IPv4 or IPv6 address. Recall from Section 5.1 that an XFN device implements one or more XFN nodes. The XFN node ID is a 32-bit integer that uniquely identifies each of these nodes. The XFN parameter ID is a 32-bit integer that uniquely identifies an XFN parameter, as an alternative to the 7-level hierarchical address. The status field keeps track of the online/offline state of the parameter referred to by the given entry. The join type field keeps track of whether a relationship is either absolute or relative. For relative relationships, the offset field keeps track of the relative differences in values between parameters. The offset (V_{offset}) between a remote parameter's value (V_{remote}) and the value of the parameter that hosts the group list (V_{host}) is calculated using the following formula:

$$V_{offset} = V_{remote} - V_{host} \tag{5.1}$$

Section 5.3 will describe how this formula was applied in the implementation. A number of join flags were defined during the course of implementation in order to cater for some of the special cases that were identified. The join flags field of the group list entry indicates the state of these flags. Section 5.5 will describe some of the special cases that were identified and the rules that were defined in order to cater for the special cases.

Field	**Size**	**Field contained in**		
		peer group list entry	**master group list entry**	**slave group list entry**
Device identifier	32-bit	Yes	Yes	Yes
XFN node ID	32-bit	Yes	Yes	Yes
XFN parameter ID	32-bit	Yes	Yes	Yes
Status	8-bit	Yes	Yes	Yes
Join type	8-bit	Yes	No	Yes
Offset	32-bit	Yes	No	Yes
Join flags	8-bit	Yes	No	Yes

Table 5.1: XFN Parameter Group List Entry Fields

Table 5.1 shows that peer and slave parameter group lists contain all the fields described above. However, master parameter group list entries do not contain the join type, offset, and join flags fields. This is due to the unidirectional nature of master-slave relationships, where value changes in the master parameter affect associated slave parameters, while value changes in the slave parameter values do not cause value changes in the associated master parameter values. This

will become clearer in Section 5.4, where a detailed description of the handling of parameter value updates is given.

The three XFN parameter group lists and the fields that each entry contains, as described above, provide a model for absolute or relative master-slave and peer-to-peer relationships. This modelling is sufficient to fulfil requirement 1 as mentioned in Section 5.1. The second requirement for parameter grouping relationships described in Section 4.2 is the need for relationships that span across different devices on a network. A parameter-centric approach to modelling parameter groups was found desirable, since it fulfils the second requirement for parameter grouping. By keeping information that fully identifies grouped XFN parameters at the parameter level, it is possible to manage XFN parameter groups in a generic decentralised manner. Such decentralisation, coupled with the ability of XFN devices to communicate between each other independently, is necessary in order to fulfil the second requirement as described in Section 4.2.

We now illustrate the concept of group lists for modelling of parameter grouping relationships with the aid of an example. Figure 5.2 shows an example layout of five XFN parameters, namely P1, P2, P3, P4, and P5. Here, P1, P2, and P3 are involved in peer-to-peer relationships, while P2 is also master of slave parameters P4 and P5. For simplicity, we will not consider the type of relationship (absolute or relative). The diagram shows the nature of each XFN parameter's group lists. The next section will describe how these group lists are managed and how value updates are handled.

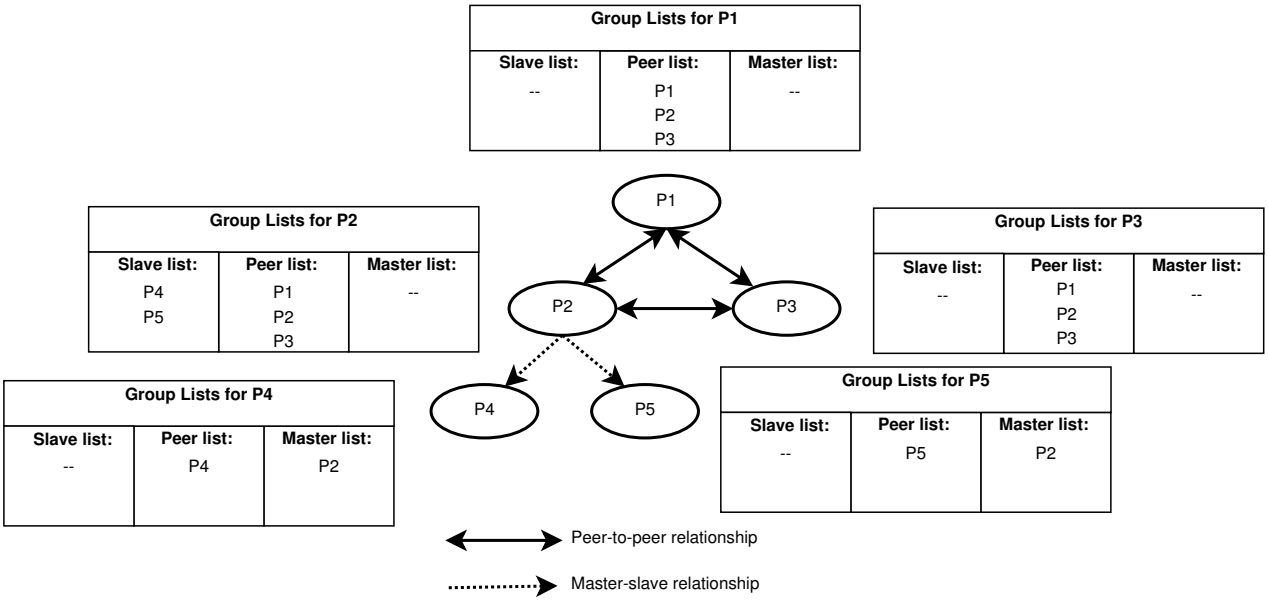


Figure 5.2: XFN Parameter Relationship Example

5.3 Managing XFN Parameter Group Lists

The previous section has described how the parameter grouping relationships are modelled by lists that reside in each XFN parameter. This section describes how the parameter group lists are managed. XFN parameter group lists change as relationships between parameters are created and broken. In particular, peer group lists are modified by creating or breaking peer-to-peer relationships, while slave group lists and master group lists are modified by creating or breaking master-slave relationships. The procedures for these actions are described below. It should be noted that the procedures below are simplified. Section 5.5 will describe how these procedures were modified to take into account a number of special cases that were identified throughout the course of implementation.

5.3.1 Creating Peer-to-Peer Parameter Relationships

We now describe the steps followed when creating a peer-to-peer parameter relationship between any two XFN parameters, P1 and P2. Here, one of the two parameters, P1 for example, initiates the process by sending commands that establish the relationship with the other parameter, P2 in this case. It should be noted that P1, P2, or both might already have peer-to-peer relationships with other parameters. For illustration purposes, we shall assume that:

- P1 is an XFN parameter that has no relationships with other XFN parameters.
- P2, P3, and P4 are XFN parameters that have peer-to-peer relationships.

Figure 5.3 shows the initial state of parameter group lists for parameters P1, P2, P3, and P4.

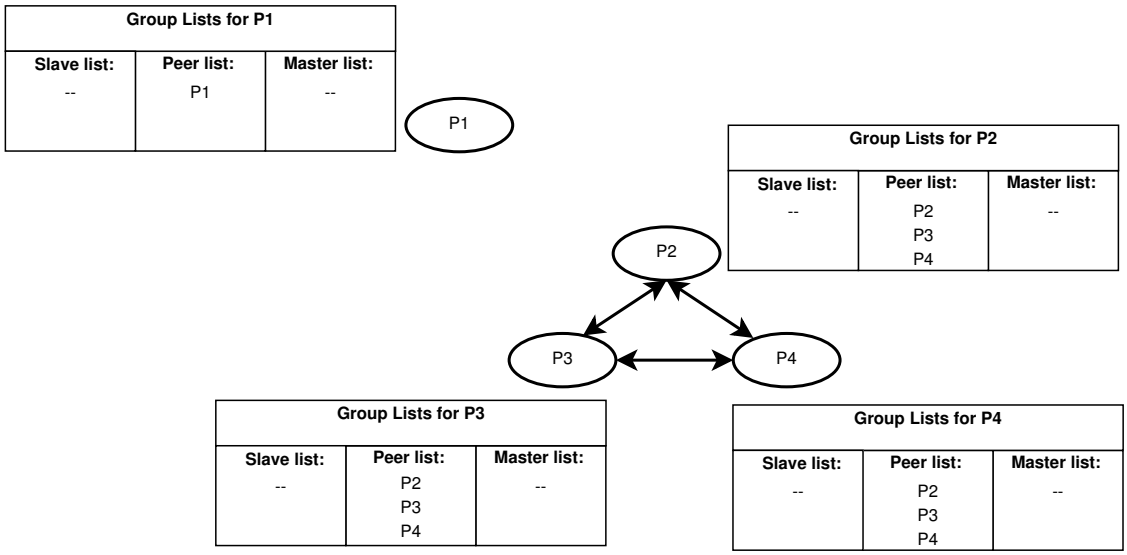


Figure 5.3: Creating a Peer-to-Peer Relationship: Initial State

The steps followed when creating a new relative or absolute peer-to-peer relationship between P1 and P2 are:

1. P1 requests the peer group list of P2 by sending a GET PTPGRP command to P2.
2. In response to the GET PTPGRP command from P1, P2 responds with a list containing its current parameter value, an indication of the number of entries in the list, and fields containing information relating to one or more peer group entries in P2. The fields relevant to peer group entries are shown in Table 5.1, namely device identifier, XFN node ID, XFN parameter ID, status, join type, offset, and join flags. Table 5.2 shows the structure of the response.
3. Recall from Section 4.1.2.2.1 that parameters in an absolute peer-to-peer relationship always share the same value. Therefore, upon receiving the response to the GET PTPGRP command, if the relationship to be created is absolute, P1 updates its value such that the value snaps to the current value of the parameter that sent the peer group list, namely P2.

Field	Size	Description
Current value	32-bits	The value of the parameter that is sending this response
Number of entries (n)	32-bits	The number of peer group entries in the peer group list of the parameter that is sending the response.
Peer group entry 1	152-bits	The peer group entry fields for the entry in position 1 of the list. Device identifier, XFN node ID, XFN parameter ID, status, join type, offset, and join flags.
:	:	:
:	:	:
Peer group entry n	152-bit	The peer group entry fields for the entry in position n of the list.

Table 5.2: GET PTPGRP Command Response

4. P1 adds any peer group list entries of P2 that are not already contained in the peer group list of P1. Figure 5.4 shows the state of group lists after this addition. Notice that at this point, in addition to its own entries, P1 contains all peer-to-peer group entries from P2. The addition of any peer group entry, PX, from P2's list to P1's list results in the formation of a new peer-to-peer relationship between P1 and PX. Thus, the fields of the group list entries need to be determined in terms of the group list hosted by P1. In particular, the offset and the join type fields need adjustment.

   (a) **Offset determination:** Equation 5.1 on page 82 shows that the value of the offset of a remote parameter that is stored in the group list of another parameter corresponds to the value of parameter that hosts the list subtracted from the value of the remote parameter. When a peer group entry for a parameter, PX, of P2's peer group list is added to the peer group list of P1, the offset of PX relative to P1 is recalculated using the formula

   $$V_{PX_{offset_{P1\,list}}} = V_{P2} + V_{PX_{offset_{P2\,list}}} - V_{P1} \tag{5.2}$$

   where:

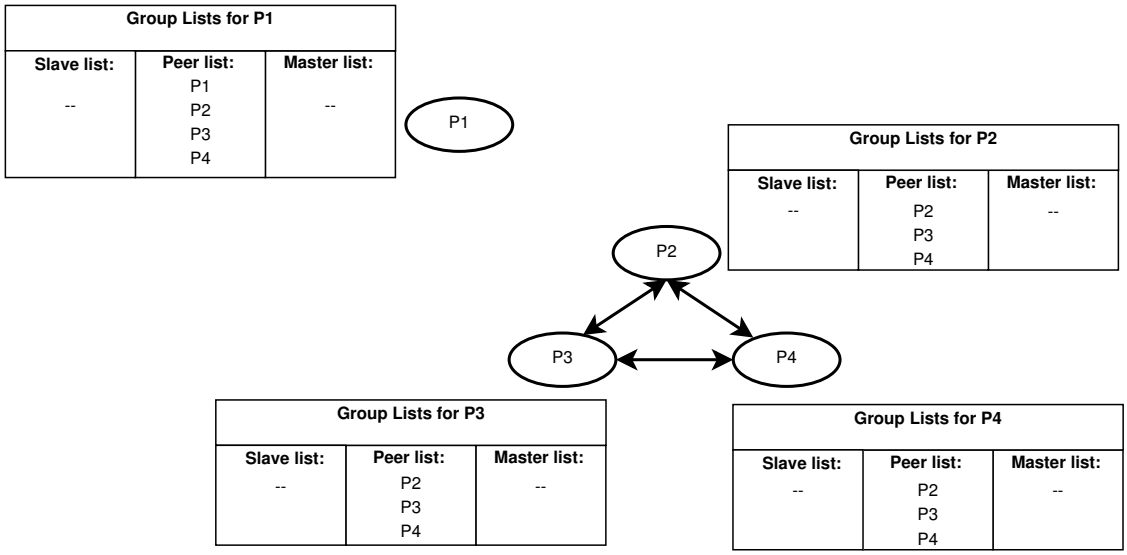


Figure 5.4: Creating a Peer-to-Peer Relationship: After First List Merge

$V_{PX_{offset_{P1\,list}}}$	is the offset of parameter PX within the list to be hosted by parameter P1 (the parameter receiving the group list from P2)
V_{P2}	is the current value of P2 (the parameter sending the group list to P1)
$V_{PX_{offset_{P2\,list}}}$	is the offset of parameter PX within the list that is hosted by parameter P2 (the parameter sending the group list)
V_{P1}	is the value of parameter P1 (the parameter receiving the group list)

We now show how the offset of PX in P1's group list (Equation 5.2) was derived. Following the reasoning in Equation 5.1 on page 82

$$V_{PX_{offset_{P1\,list}}} = V_{PX} - V_{P1} \tag{5.3}$$

and by the same reasoning, we know that

$$V_{PX_{offset_{P2\,list}}} = V_{PX} - V_{P2} \tag{5.4}$$

$$\implies V_{PX_{offset_{P2\,list}}} + V_{P2} = V_{PX}$$

Substituting V_{PX}in Equation 5.3 gives Equation 5.2, which is generalised as:

"new offset" = "value of list sender" + "offset in list" - "value of list receiver" (5.5)

(b) **Join type determination:** When a new parameter entry, PX, is added to P1's peer group entry list based on the information received from P2's peer group entry list, the relationship type (either absolute or relative) of PX in relation to P1 is inferred based on the nature of relationship between P1 and P2, and the relationship between P2 and PX. The relationship between P1 and P2 will be known at the time that the new relationship is initiated. The relationship between P2 and PX will be known from the peer group list entry of PX in P2's peer group list. Section 5.5.2 will describe the inference rules that were defined in the course of implementation.

5. By definition, a peer-to-peer relationship is one such that a change in value of one parameter in the group results in a change in value of all other parameters in the group. Thus, when a peer-to-peer relationship is created between two parameters that already have other peer-to-peer relationships, both peer-to-peer groups are merged to form a larger peer-to-peer group, where parameters from each of the two groups share peer-to-peer relationships with each other.

   In the context of our illustration, the addition of P2's peer group list entries to P1's peer group list entries is the first step in merging the two peer-to-peer groups. After all the new peer group list entries within P2's peer group list have been added to P1's peer group list, P1 will have knowledge of the desired final peer-to-peer group. In order to ensure that all other parameters contain exactly the same group description as P1, P1 sends its updated peer group list to all parameters via a SET PTPGRP command. The layout of the SET PTPGRP command is exactly the same as that of the GET PTPGRP command response as shown in Table 5.2. Each parameter that receives the SET PTPGRP command, adds any new peer group list entries to its peer group list as described in part 4 above. Figure 5.5 shows the resultant state of group lists after the creation of the new peer-to-peer relationship is complete, where all parameters are now peers of each other.

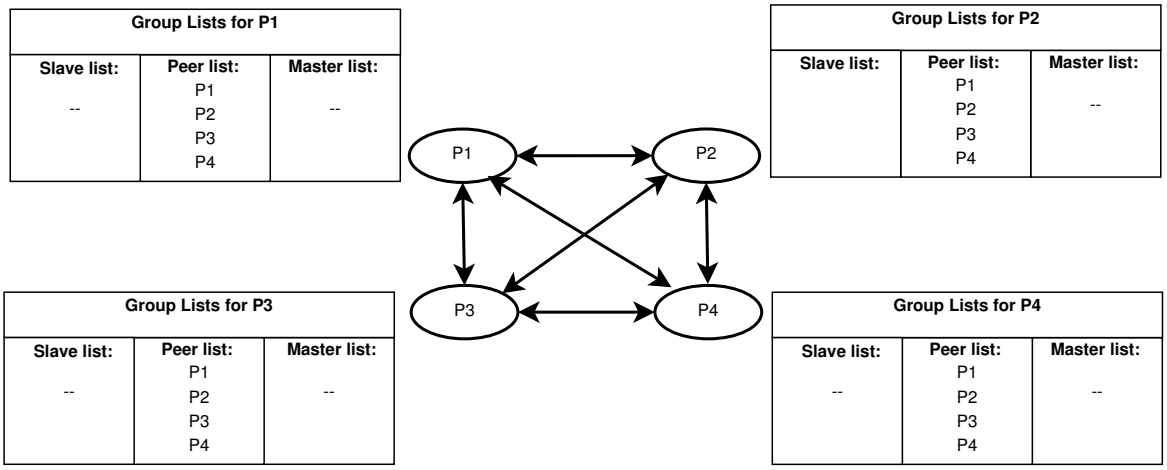


Figure 5.5: Creating a Peer-to-Peer Relationship: Final State

Figure 5.6 provides a diagrammatic summary of the steps required to create a peer-to-peer relationship as described above.

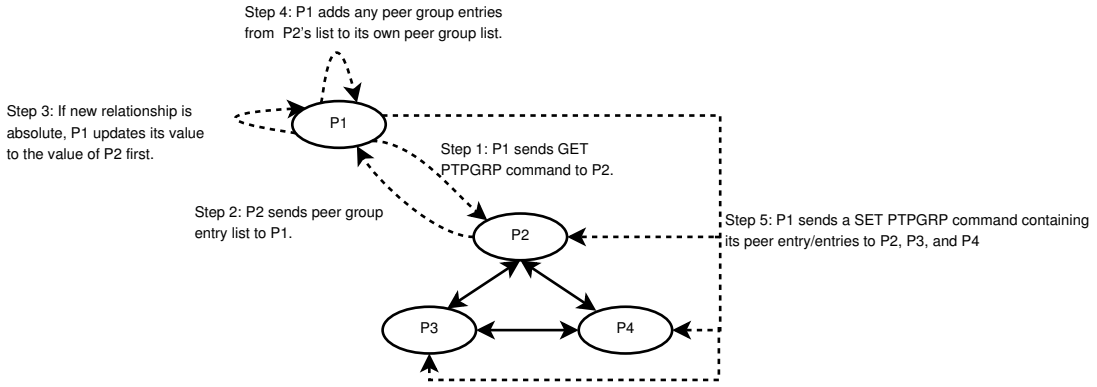


Figure 5.6: Creating a Peer-to-Peer Relationship: Summary of Steps

5.3.2 Breaking Peer-to-Peer Parameter Relationships

Section 5.3.1 has just described the procedure for creating peer-to-peer relationships. We now describe the steps that were implemented in order to break existing peer-to-peer relationships. We continue from the final state of the example described in Section 5.3.1 as shown in Figure 5.5. Assuming that the peer-to-peer relationships between P4 and all other parameters are to be broken, the following steps are necessary:

1. P4 removes the entries for P1, P2, and P3 from its peer group entry list.
2. P4 sends a SET PEER_OFF command to P1, P2, and P3. The SET PEER_OFF command contains information that uniquely identifies the peer group entry for P4 in the peer group entry lists.
3. Upon receiving the SET PEER_OFF command, each parameter removes the peer group entry for P4 from its peer group entry list.
4. Figure 5.7 shows the resulting parameter group lists after the SET PEER_OFF commands have been processed, where P4 has been isolated from the sub-group containing parameters P1, P2, and P3.

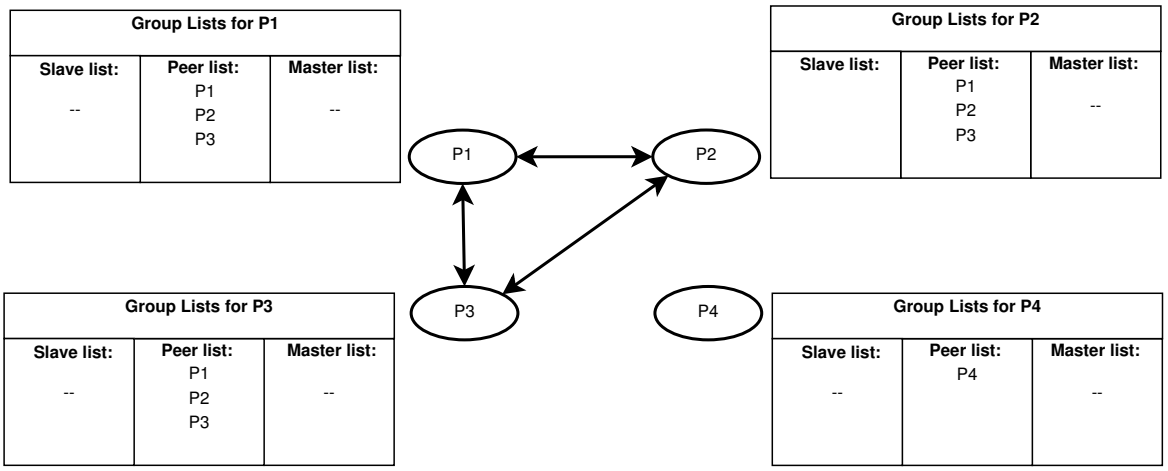


Figure 5.7: Breaking Peer-to-Peer Relationships: Final State

5.3.3 Creating Master-Slave Parameter Relationships

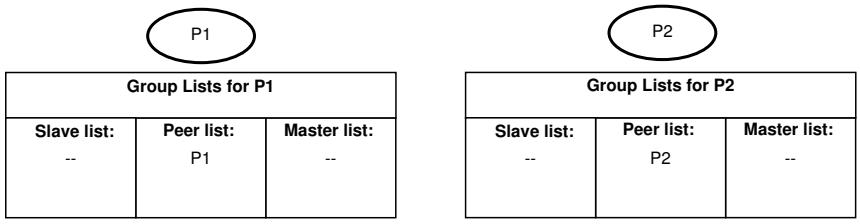


Figure 5.8: Creating a Master-Slave Relationship: Initial State

Figure 5.8 shows the initial state of parameter group lists for two XFN parameters, namely P1 and P2. In order to create a relative/absolute master-slave relationship, where P1 is master of P2, the following steps are necessary:

1. P1 sends a SET MASTERS command containing an entry of itself to P2. The structure of the command is shown in Table 5.3.

Field	Size	Description
Current value	32-bits	The value of the parameter that is sending this command
Number of entries (n)	32-bits	The number of master group entries being sent by this command.
Master group entry 1	152-bits	The master group entry fields for the entry in position 1 of the list. Device identifier, XFN node ID, XFN parameter ID, status, join type, offset, and join flags.
:	:	:
:	:	:
Master group entry n	152-bits	The master group entry fields for the entry in position n of the list.

Table 5.3: SET MASTERS Command Layout

2. Upon receiving the SET MASTERS command,

   (a) If the relationship specified in the master entry is absolute, P2 changes its value to the value of the master entry.

   (b) P2 recalculates the offset of the master entry using the generalised formula shown in Equation 5.5 on page 88.

   (c) P2 adds the master entry to its master group entry list.

3. P1 adds an entry of P2 to its slave group entry list. Figure 5.9 shows the resulting state of the parameter group lists after the master-slave relationship has been created.

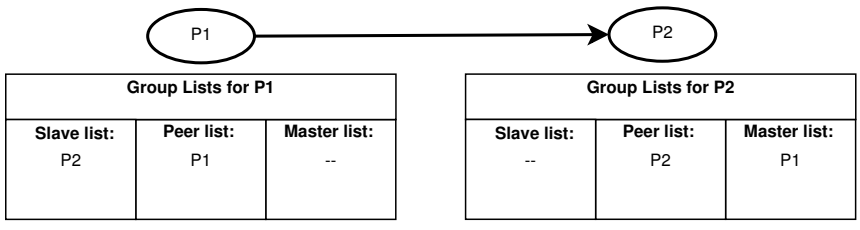


Figure 5.9: Creating a Master-Slave Relationship: Final State

5.3.4 Breaking Master-Slave Parameter Relationships

We now describe the procedure for breaking master-slave relationships. We continue from the final state of the creation of master-slave relationships shown in Figure 5.9. Here, in order to break the master-slave relationship between P1 and P2, the following steps are necessary:

1. P1 sends a SET MASTER_OFF command that contains an entry of itself to P2. The structure of this command is identical to that of the SET MASTERS command as shown Table 5.3.
2. Upon receiving the SET MASTER_OFF command from P1, P2 removes the entry of P1 from its master group entry list.
3. P1 removes the entry of P2 from its slave entry list, effectively breaking the master slave relationship as shown previously in Figure 5.8.

5.4 Handling XFN Parameter Value Updates

The previous section has described how parameter group lists, in their simplest forms, are managed. Based on this parameter group list structure, we now describe how changes in parameter values of grouped XFN parameters are handled. In addition, this section describes the concept of a common value unit that was implemented in order to enable the grouping of parameters without prior knowledge of their underlying value units.

5.4.1 Handling Parameter Value Changes

A number of factors can cause the value of an XFN parameter to change. These factors include, changes in application-specific properties such as DSP parameters, user actions via control surfaces, and changes in values of other grouped parameters. The SET GRPVAL (set group value) command was added for use in value updates of grouped XFN parameters. Here is the typical sequence of events showing what happens when a parameter's value is changed:

1. A parameter receives a command to change its value. This is typically triggered by a SET VAL (set value) command that is directed at the parameter.

2. Upon receiving a SET VAL command the parameter updates its value to the value specified by the SET VAL command.
3. The parameter sends a SET GRPVAL (set group value) command containing its new value to all parameters in its peer and slave group lists.
4. Upon receiving a SET GRPVAL command, a parameter first ensures that the parameter that sent the command is present in either its master or peer group list. If the parameter that sent the SET GRPVAL command is present in either of the lists the following sequence of events takes place:

   (a) If the relationship between the parameter that sent the SET GRPVAL command and the one receiving the command is absolute, the parameter receiving the command updates its value to the value specified in the SET GRPVAL command.

   (b) If the relationship between the parameter that sent the SET GRPVAL command and the one receiving the command is relative, the parameter receiving the command updates its value in a way that takes into account the offset in the group entry list. The new value is calculated as follows

   $$V_{host} = V_{remote} - V_{offset}$$

   This equation is derived from the equation that is used to calculate offsets as shown in Equation 5.1 on page 82.

   (c) After the value of a parameter is updated by a SET GRPVAL command, the parameter sends a SET GRPVAL to parameters that are in its slave group entry list. Slave parameters that receive the SET GRPVAL command repeat step 4 as described above, making it possible to handle updates in a hierarchy of master-slave relationships. Updates are not sent to other peer parameters upon receipt of a SET GRPVAL command since this would result in an infinite loop of peer parameter value updates. Section 5.5.1.2.1 will describe a rule that ensures that peers of slave parameters remain synchronised.

Figure 5.10 shows a graphical illustration of this XFN parameter value update process with six XFN parameters, namely P1, P2, P3, P4, P5, and P6. Here, P1, P2, and P3 are involved in peer-to-peer relationships with each other. P2 is master parameter to slave parameters P4 and P5, while P4 is master parameter to slave parameter P6. Notice that there is hierarchy of master-slave relationships from P2, to P4, to P6, where P2 is referred to as a *grand master* of P6. When

P1's value is changed by a SET VAL command, P1 updates its value and issues a SET GRPVAL command to its peers, P2 and P3. P2 and P3 update their values depending on nature of their relationship with P2, either absolute or relative. Since P2 is a master to P4 and P5, P2 re-issues a SET GRPVAL command containing its latest value to both P4 and P5. P4 and P5 update their values depending on the nature of the relationship, either relative or absolute. Since P4 is a master to P6, P4 re-issues a SET GRPVAL command containing its latest value to P6. Lastly, P6 updates its value depending on the relative or absolute nature of the master-slave relationship with P4.

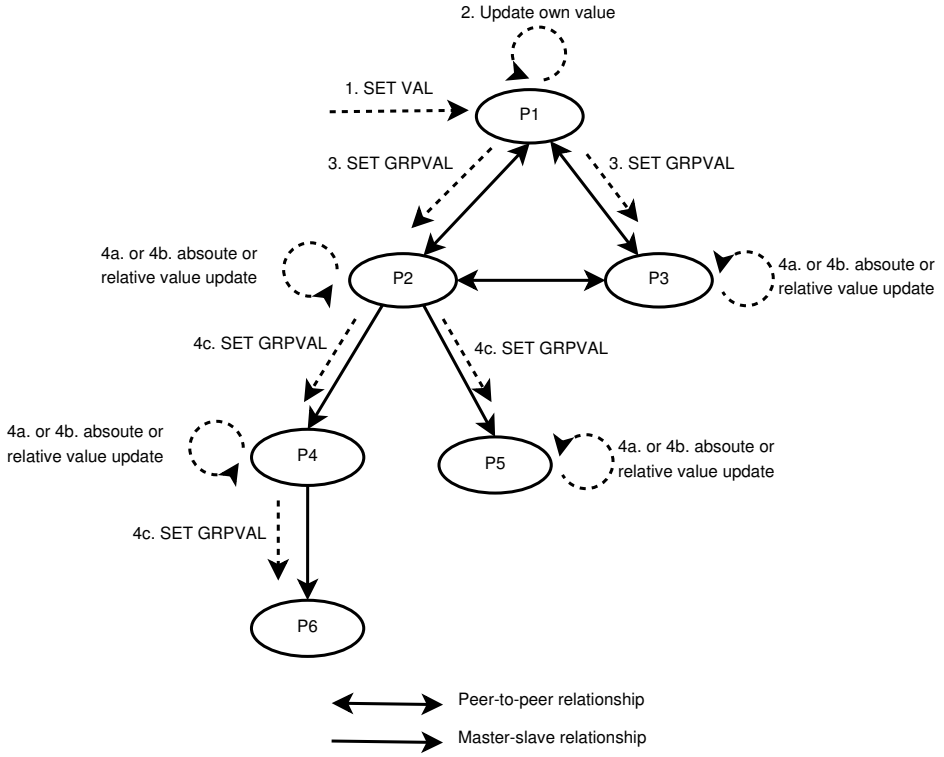


Figure 5.10: XFN Parameter Value Updates

5.4.2 Handling of Non-Additive Relationships

Group value updates are handled in an additive manner, where the following formula is used to calculate the new value of a parameter: $V_{host} = V_{remote} - V_{offset}$, as described in Section 5.4.1. Section 2.2.1.1.3.1 on page 20 introduced the concept of inverse relationships, while Example 2 in Section 2.2.2.1.1.1 on page 22 highlighted the need for multiplicative relationships

as opposed additive relationships. Multiplicative and inverse relationships do not fit into the relationship scheme that was implemented as part of this study. However, we note that extended functionality, referred to as the "value modifier" [Audio Engineering Society, 2012], was added (outside this study) for the XFN protocol to support customised value update handling.

A value modifier is an intermediary module that changes/"modifies" the value of an XFN message targeted at an XFN parameter. Figure 5.11 shows an example of a value modifier that alters values sent from P1 to P2. The modifier contains an input value parameter (IVP) that is joined to P1, and an ouput value parameter (OVP) that is joined to P2. In addition, the value modifier contains a value change function that applies rules, such as inversion or multiplication, to the input value in order to produce a modified output value. When the value of P1 changes:

- P1 sends a value update to the IVP via the additive parameter grouping mechanism proposed in this study.
- The IVP uses the value change function to modify the input value.
- The output value of the modifier is sent from the OVP to P2 also via the additive parameter grouping mechanism proposed in this study.

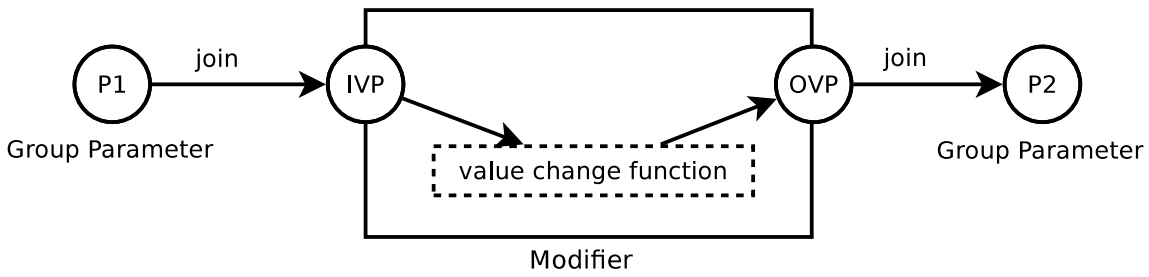


Figure 5.11: Value Modifier

5.4.3 Common Value Unit

Up to now, we have focused on the modelling of XFN parameter relationships using three parameter group lists, namely master, slave, and peer group lists. We have also shown how XFN parameter value updates are handled. However, we have not referred to any value units. Through an analysis of the requirements for grouped parameter relationships, a number of factors were taken into account before a value unit was defined. These factors include:

1. The ability to create relationships between parameters that have different underlying value units as mentioned in Section 4.3. For example, a relationship between light intensity and some audio property, such as volume.
2. The need for a unit that is flexible enough to be implemented in any type of controller ranging from 1-bit controllers, such as on/off switches, to 32-bit controllers, such as computer processors.
3. The ability to cope with the varying needs of relationships described above, where values and offsets need to be calculated dynamically.

During the course of implementation, a value unit that takes these factors into account was defined for grouped XFN parameters. This unit is referred to as a *global unit*. The rest of this section describes the global unit in detail.

5.4.3.1 Global Unit Value Type Description

The global unit was defined as a 32-bit positive integer value, where the most significant bit is always 0. The 32-bit range is allocated as shown in Table 5.4, where the 0% mark of the global unit value range is defined as 20000000_{16}, while the 100% mark of the global unit value range is defined as $3FFFFFFF_{16}$.

Global Unit Value			
Hexadecimal	**Binary**	**Decimal**	**Percentage**
0x7FFFFFFF	$_{msb}$0111 1111 1111 1111 1111 1111 1111 1111$_{lsb}$	2147483647	300%
0x5FFFFFFF	$_{msb}$0101 1111 1111 1111 1111 1111 1111 1110$_{lsb}$	1610612735	200%
0x3FFFFFFF	$_{msb}$0011 1111 1111 1111 1111 1111 1111 1111$_{lsb}$	1073741823	100%
0x20000000	$_{msb}$0010 0000 0000 0000 0000 0000 0000 0000$_{lsb}$	536870912	0%
0x00000000	$_{msb}$0000 0000 0000 0000 0000 0000 0000 0000$_{lsb}$	0	-100%

Table 5.4: Global Unit Range Allocation

When considering relative parameter relationships and the need to maintain an offset between the values of any two parameters, two scenarios were identified where a simple 0 - 100% value

range was found to be insufficient. Figure 5.12 graphically illustrates these two scenarios with two parameters, namely P1 and P2. P1's initial value is 100% of the global unit range, while P2's initial value is 0% of the global unit range. Assume P1 and P2 are joined with a relative peer-to-peer relationship. The first scenario occurs when, for example, the value P1 is reduced to 50%, causing an underflow in the new value of P2, where P2's value is -50%. The second scenario occurs when, for example, the value of P2 is increased to 50%, causing an overflow in the new value of P1, where P1's new value is 150%.

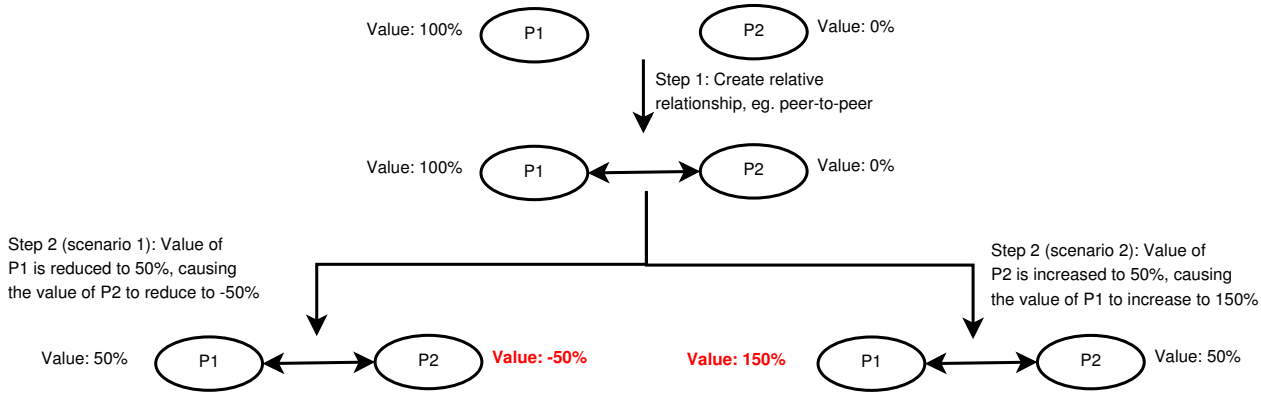


Figure 5.12: Global Unit Value Range Overflow and Underflow

Having identified the possibility of value overflow and underflow, it was appropriate to define a range for the global unit that not only spans from 0 - 100% but also allows for some value overflow and underflow. Value overflow is taken into account by values that are greater than $3FFFFFFF_{16}$ and less than or equal to $7FFFFFFF_{16}$ as shown in Table 5.4. As a proportion of the 0 - 100% global unit value range, $7FFFFFFF_{16}$ marks the 300% mark of the global unit value range. In contrast, value underflow is taken into account by values that are greater than or equal to 0_{16} but less than 20000000_{16}, where 0_{16} marks the -100% mark of the global unit value range.

From Table 5.4, it can be seen that the entire global unit value range is -100% (0_{16}) through to 300% ($7FFFFFFF_{16}$). Such a range allocation ensures that the most significant bit (msb) of the entire 32-bit range is always zero, thus ensuring that all global unit values are positive numbers in any calculations. The factors that led to the asymmetric -100% to 300% global unit value range are described below.

Factors Leading to the Asymmetric Global Unit Value Range

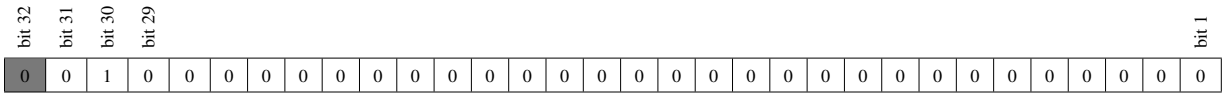


Figure 5.13: Global Unit Value Bits

As indicated above, the global unit value range is asymmetric, spanning from -100% (0_{16}) through to 300% ($7FFFFFFF_{16}$). The asymmetry within the range resulted from the need to make global unit values controllable by variable-bit-sized controllers, ranging from 1-bit to 32-bit controllers, while allowing for value overflow and underflow. Figure 5.13 shows the various bit positions of this value range. Bit 32 is always 0 in order to maintain all numbers as positive integers. In order for the global unit value range to be controllable by a variable-bit-sized controller, the 0% to 100% value marks should correspond to no bits set and all bits set, respectively. Assume the 0% to 100% global unit value is represented by bits 29 through to bit 1, inclusive. Examples of possible values per controller include:

- For a 1-bit controller, the 0% to 100% global unit value is represented by the value of bit 29, where the value is 100% when bit 29 is set and 0% when bit 29 is not set.
- For a 2-bit controller, the 0% to 100% global unit value is represented by the value of bits 29 and 28, where the value is 100% when bits 29 and 28 are both set and 0% when bits 29 and 28 are both not set.
- *... (other values omitted for brevity)*
- For a 29-bit controller, the 0% to 100% global unit value is represented by the value of bits 29 through to 1, where the value is 100% when all of bits 29 through to 1 are set and 0% when all of bits 29 through to 1 are not set.

If the 0% to 100% global unit value range is represented by bits 29 through to 1 alone, it is not possible to allow for value overflow and underflow. In order to allow for overflow and underflow, the defined values for 0% and 100% global unit values were altered such that bit 30 is always set. Therefore, for a 29-bit controller, the 0% global unit value becomes 20000000_{16}, while the 100% global unit value becomes $3FFFFFFF_{16}$. As a proportion of the 0% - 100%, the global unit value where all bits are not set (0_{16}) is -100%, while the global unit value where all but bit 32 are set ($7FFFFFFF_{16}$) is approximately 300%.

5.4.3.2 Global Unit Value Manipulation by Various Controllers

As mentioned previously, one of the considerations taken into account before defining the global unit was the need for ease of use across various types of controllers, including controllers with limited capabilities, such as 1-bit controllers (on/off switch, for example). In order to change the global unit value of a parameter, a controller is required to alter any of the least significant 31 bits of the parameter's global unit value as shown in Figure 5.14. Such bitwise manipulation is desirable, since it allows any type of controller to control any parameter on the network. It should be noted that setting bit 31 to 1 results in an overflow global unit value (greater than 100%) being set. In contrast, whenever bits 30 and 31 are set to 0, the result is an underflow global unit value (less that 0%). Controllers typically function in the 0 - 100% region of global unit values, while overflow and underflow results mainly from relative relationships. Bearing this in mind, we now show how an 8-bit controller, for example, would modify a value within the 0 - 100% global unit value range.

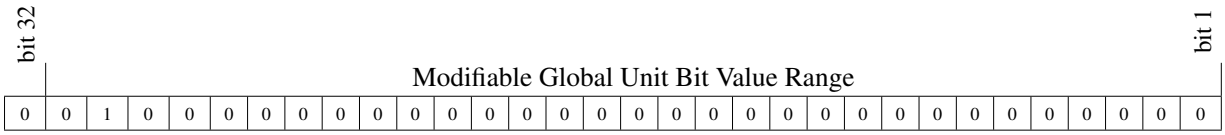


Figure 5.14: Bitwise Global Unit Range Modification

Figure 5.15 shows an 8-bit controller that is positioned to control bits 22 to 29, inclusive. The global unit value being modified in the figure is 20000000_{16}. The global unit value is modified by copying the bit values of the 8-bit controller to bits 22 through to 29 of the global unit value. Here, each increment of the 8-bit controller by a value of one will increment the global unit by a value of $10000000000000000000_2(2^{21} = 2097152_{10})$. Therefore, at this level the 8-bit controller produces global unit values that range from 20000000_{16} to $3FE00000_{16}$.

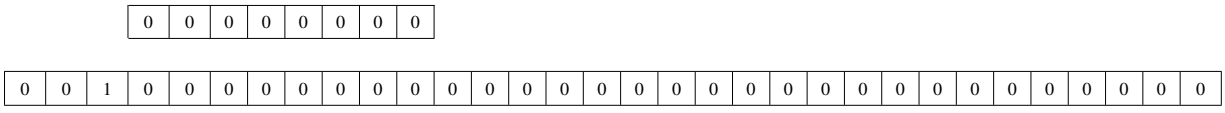


Figure 5.15: Global Unit Value 8-bit Controller: Coarse Granularity

In order to control the global unit value with finer granularity, a controller is required to alter the least significant bits of the global unit value. Continuing from the 8-bit controller example, Figure 5.16 shows a controller with a value of 10101010_2 positioned over bits 1 to 8, inclusive. In

this position, the global unit will be $200000AA_{16}$. The controller is capable of representing values in the range $20000000_{16} - 200000FF_{16}$, where each increment of the 8-bit controller by a value of one will increment the global unit by a value of one $(1_2 = 1_{10})$. This is the finest granularity by which any global unit can be changed. As a proportion of the 0 - 100% global unit range, where 0% is 536870912_{10}, while 100% is 1073741823_{10}, this is equivalent to 0.00000019% as shown in Equation 5.6 below

$$\frac{1}{1073741823 - 536870912} * 100 = 0.00000019\% \tag{5.6}$$

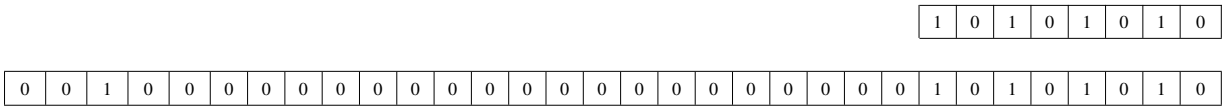


Figure 5.16: Global Unit Value 8-bit Controller: Finest Granularity

Throughout our descriptions in this section, we have made reference to an 8-bit controller. However, it is important to emphasise that the global unit was defined to be sufficiently generic to support variable-bit-sized controllers. The next section describes how global unit values can be mapped to meaningful units/quantities.

5.4.3.3 Global Unit Value to Application-Specific Unit Mapping

The previous section presented a generic description of the global unit that forms the common means of interaction between grouped parameters. XFN parameters are typically associated with application-specific parameters that have their own units, such as gain, light intensity, frequency, time, *et cetera* . All multimedia devices containing parameters that can be grouped are required to implement mapping tables that convert a global unit value to an appropriate application-specific quantity/unit. Each application is required to map the linear global unit value range in whichever way that makes sense for it. For example, some mappings may be linear, while others might be non-linear. Linear and non-linear mappings are described in Sections 5.4.3.3.1 and 5.4.3.3.2, respectively.

5.4.3.3.1 Linear Global Unit Mappings

Linear global unit mappings map a value range from a minimum value to a maximum value, in fixed increments. For example, audio level parameters can be mapped from -144 dB to +60 dB

in steps of 1 dB. Table 5.5 shows the resulting linear dB-to-Global-Unit mapping.

dBu	Global Unit Value		
	Decimal	**Hex**	**Percentage**
-144	536870912	0x20000000	0.00%
-143	539502632	0x20282828	0.49%
-142	542134352	0x20505050	0.98%
:	:	:	:
-123	592137035	0x234B4B4B	10.29%
:	:	:	:
-103	644771438	0x266E6E6E	20.10%
:	:	:	:
-82	700037561	0x29B9B9B9	30.39%
:	:	:	:
-62	752671964	0x2CDCDCDC	40.20%
:	:	:	:
-42	805306368	0x2FFFFFFF	50.00%
:	:	:	:
-21	860572491	0x334B4B4A	60.29%
:	:	:	:
-1	913206894	0x366E6E6D	70.10%
:	:	:	:
20	968473017	0x39B9B9B8	80.39%
:	:	:	:
40	1021107420	0x3CDCDCDB	90.20%
:	:	:	:
60	1073741823	0x3FFFFFFF	100%

Table 5.5: Example Linear dB-to-Global-Unit Mapping Table

5.4.3.3.2 Non-Linear Global Unit Mappings

Table 5.6 shows an example of a non-linear mapping table that can be used by an application to map global unit values to a dBu scale. In this example, the dBu value is derived from the percentage value (as a ratio out of 100) in the following equation: $\text{dBu} = 20\log_{10}\frac{\%}{100}$. This equation is based on the fact that a decibel value is calculated based on a ratio. For example, in terms of voltage, the decibel (dBu) value is calculated relative to 0.7746V using the following equation: $\text{dBu} = 20\log_{10}\frac{\text{V}}{0.7746}$.

dBu	Global Unit Value		
	Decimal	**Hex**	**Percentage**
-∞	536870912	0x20000000	0%
:	:	:	:
-20	590558003	0x23333333	10%
:	:	:	:
-13	644245094	0x26666666	20%
:	:	:	:
-10	697932185	0x29999999	30%
:	:	:	:
-8	751619276	0x2CCCCCCC	40%
:	:	:	:
-6	805306368	0x2FFFFFFF	50%
:	:	:	:
-4	858993459	0x33333332	60%
:	:	:	:
-3	912680550	0x36666665	70%
:	:	:	:
-2	966367641	0x39999998	80%
:	:	:	:
-1	1020054732	0x3CCCCCCB	90%
:	:	:	:
0	1073741823	0x3FFFFFFF	100%

Table 5.6: Example Non-Linear dBu-to-Global-Unit Mapping Table

5.4.3.3.3 Considerations for Standardising the Mapping of Global Units to Universal Units

The definition of global unit value mapping tables is done at the application level. Inconsistencies arise when a global unit has different mappings for a given universal unit, particularly across multiple vendor implementation. Thus, a standard set of global unit value mapping tables must be used by all devices in a network.

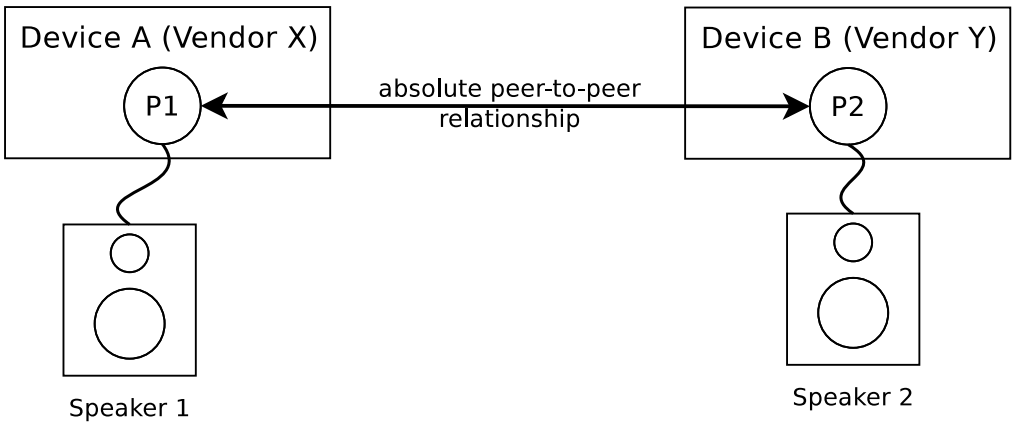


Figure 5.17: Cross-Vendor Parameter Relationships

Figure 5.17 shows an example of a scenario that can result in mapping table inconsistencies due to cross-vendor parameter relationships. In this scenario, the following assumptions have been made:

- Device A from vendor X contains an XFN parameter, P1.
- Device B from vendor Y contains another XFN parameter, P2.
- P1's function is to control the volume of speaker 1.
- P2's function is to control the volume of speaker 2.
- The volume of the speaker 1 and speaker 2 shall always be the same. Therefore, an absolute peer-to-peer relationship exists between P1 and P2.
- Vendor X maps the 0% - 100% global unit range to a universal unit range of -∞ dBu - 0 dBu as shown in Table 5.7a.
- Vendor Y maps the 0% - 100% global unit range to a universal unit range of -13 dBu - 2 dBu as shown in table 5.7b.

Recall from Section 4.1.2.2.1 on page 73 that an absolute peer-to-peer relationship between parameters implies that the associated parameters all share the same value. Thus, when P1 is set to the 80% global unit value, for example, P2 is also set to the 80% global unit value. Since P1 and P2 use different vendor-specific mapping tables, P1 is set to -2 dBu, while P2 is set to 0 dBu. Although P1 and P2 are set to the same global unit value, the speaker 2 is always louder than speaker 1. This discrepancy is due to the use of different mapping tables for the same universal unit.

Global unit value	Universal unit equivalent (dBu)
0%	-∞
:	:
20%	-13
:	:
40%	-8
:	:
60%	-4
:	:
80%	-2
:	:
100%	0

(a) Vendor X

Global unit value	Universal unit equivalent (dBu)
0%	-13
:	:
20%	-8
:	:
40%	-4
:	:
60%	-2
:	:
80%	0
:	:
100%	2

(b) Vendor Y

Table 5.7: Vendor-Specific Global Unit to Decibel Mapping Tables

In order to avoid discrepancies similar to the one mentioned above, standard global-unit-to-universal-unit mapping tables are required. Vendor-specific mapping tables then become permissible only for proprietary application-specific units.

5.5 Parameter Grouping Rules

In the course of implementation, three special cases that required some attention were identified. These cases include the hybridisation of master-slave and peer-to-peer groups, inferring relationship types upon addition of group members, and the retention of permanent relationships. In order to deal with these special cases, some rules were defined to govern parameter grouping. This section describes the scenarios that were identified and the rules that were subsequently defined to deal with the identified scenarios.

5.5.1 Hybridisation of Master-Slave and Peer-to-Peer Groups

5.5.1.1 Scenario

Section 5.3 focused on the creation and breaking of master-slave or peer-to-peer relationships. The procedures described assumed that the peer-to-peer relationships and master-slave relationships were created in isolation. However, in reality these relationships are not necessarily mutually exclusive. For example, a sound engineer may create peer-to-peer relationships among a group of volume parameters and then later decide to have a single master volume parameter controlling at least one parameter that belongs to the initial peer group of volume parameters.

Figure 5.18 illustrates the most basic case of the scenario described above, where parameters P2, P3, and P4 are involved in a peer-to-peer relationship, while P1 is master to P2. In this example, any change in value of P1 would be sent to its slave parameter, P2. However, in order for the peer-to-peer relationship to remain valid, P2 would then be required to update its peer parameters, P3 and P4, with a "set group value" command that contains the new value of P2. It follows that in this scenario, P2 becomes the single parameter that is responsible for propagating all updates from the master parameter, P1, resulting in updates being chained from a master parameter to a slave parameter, and from the slave parameter to its peer parameters. Such chaining of updates was identified as a potential cause for value update delays in any downstream parameters.

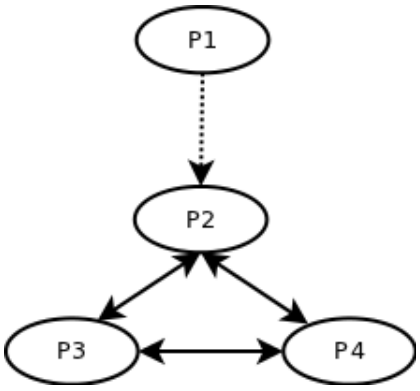


Figure 5.18: Master to Slave in Peer Group

Figure 5.19 illustrates the same scenario, although with parameter P1 as master to more than one slave parameter, where the slave parameters, P2 and P3, belong to the same peer-to-peer group. Here, any change in value of P1 would be sent to the slave parameters, P2 and P3. Upon

receiving each update, P2 and P3 would both update their peers in order to ensure that the peer-to-peer relationships remain valid. Since each of the peer parameters has no knowledge of which peer is under which master parameter's control, P2 would send value updates to peers P3 and P4, while P3 would send value updates to peers P2 and P4. Notice that P4 receives duplicate updates from each of the affected slaves and that each of the slaves receives an update from the other affected slaves. When more complex groups of parameters are created, such unnecessary duplication was identified as a potential source of inconsistency and inefficiency.

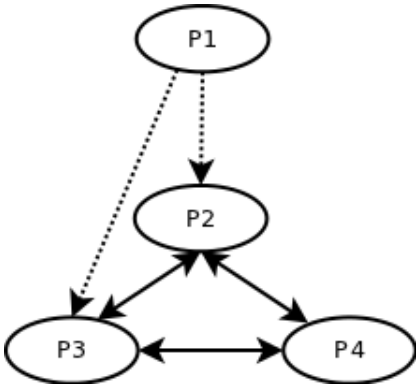


Figure 5.19: Master to Multiple Slaves in Peer Group

5.5.1.2 Rules Defined

Two problems resulting from the hybridisation of master-slave and peer-to-peer relationships have been described above, namely delays due to propagation of updates and the unnecessary duplication of updates. In order to resolve these two problems rules were defined to govern the creation and breaking of master-slave and peer-to-peer relationships. The rules defined are based on the underlying assumption that any parameters that have a peer-to-peer relationship should share the same master parameters, since any changes in value of a peer parameter will affect other peer parameters. Therefore, three rules were defined as follows:

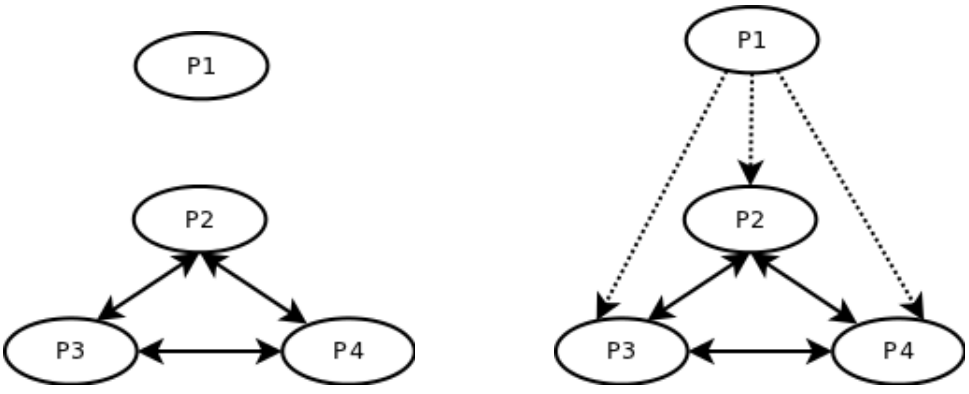
1. When a master-slave relationship is created between a master parameter and a slave parameter that belongs to a peer group, the master becomes master to all the slave's peer parameters. Consequently, whenever a slave parameter receives an update from a master parameter, the slave parameter shall not propagate the change to its peer parameters.

2. When a master-slave relationship is broken between a master parameter and a slave parameter that belongs to a peer group, the master-slave relationship between the master and each of the peer parameters is also broken.
3. When a peer-to-peer relationship is created between two parameters that are under the control of one or more unique master parameters, the masters are merged such that both parameters (including their peers) are controlled by the same set of master parameters.

We now describe each of these three rules in detail with the aid of examples.

5.5.1.2.1 Rule 1: Master Becomes Master of all Peers

Figure 5.20a shows the initial state before a master-slave relationship is created between parameters P1 and P2, where P1 becomes the master of P2. As shown in the diagram, peer-to-peer relationships exist between parameter P2 and parameters P3 and P4. By applying our rule, where the master becomes the master of all peer parameters, P1 first requests the list containing the peer group entries of P2 before creating any master-slave relationship. The peer group entry list of P2 contains entries for P2, P3, and P4. Lastly, P1 then creates a master slave relationship with each of the peers in the peer group list from P2. This results in a state where P1 is effectively a master parameter to P2, P3, and P4 as shown in Figure 5.20b.

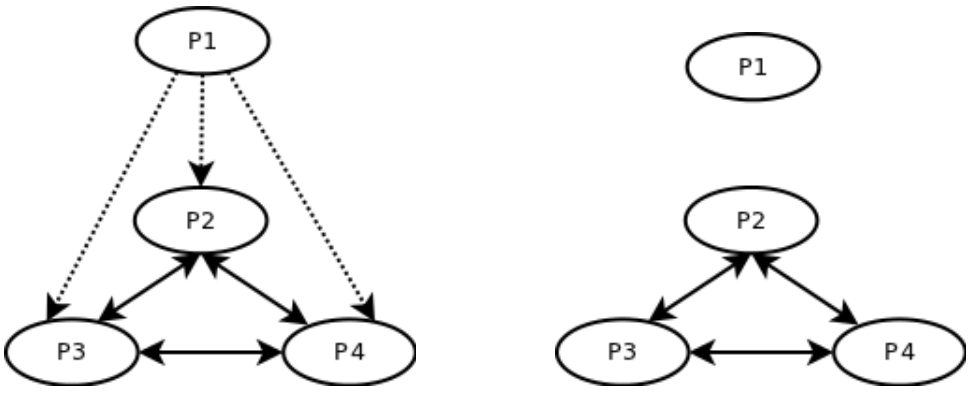


(a) Before (b) After

Figure 5.20: Creating a Master-Slave Relationship with a Peer Group

5.5.1.2.2 Rule 2: Master Removed from all Peers

The second rules follows from the first rule described above. Here, when a master-slave relationship is broken between a master parameter and a slave parameter that is part of a peer group, the other master-slave relationships that exist with the slave's peers are also broken. Figure 5.21a shows an initial state where parameter P1 is master to slave parameters P2, P3, and P4, while P2, P3, and P4 have peer-to-peer relationships between them. Assume that the master-slave relationship between the master and any of the slave parameters, say P4, is to be broken. By applying our rule, P1 first requests the list containing the peer group entries of P4. The peer group entry list of P4 contains entries for P2, P3, and P4. The procedure is completed by breaking the master-slave relationships that exist between the master and each of the entries in the peer group entry list of P4. This results in a state where all the joins between the master parameter and each member of the peer group are broken as shown in Figure 5.21b. Notice how the peer-to-peer relationships remain intact.



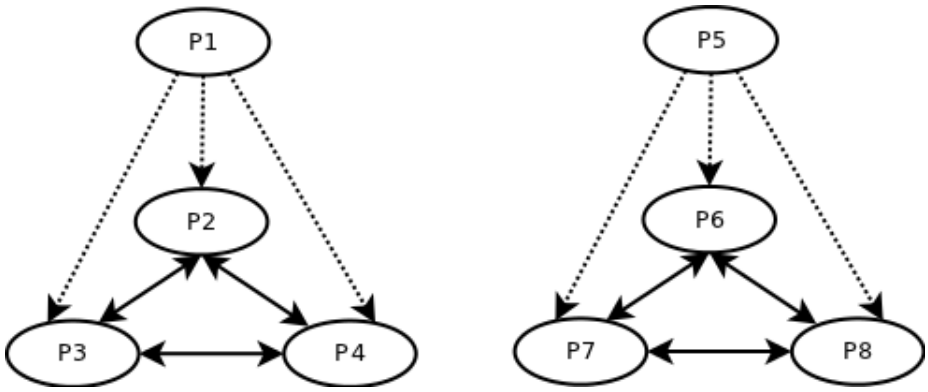
(a) Before (b) After

Figure 5.21: Breaking a Master-Slave Relationship with a Peer Group

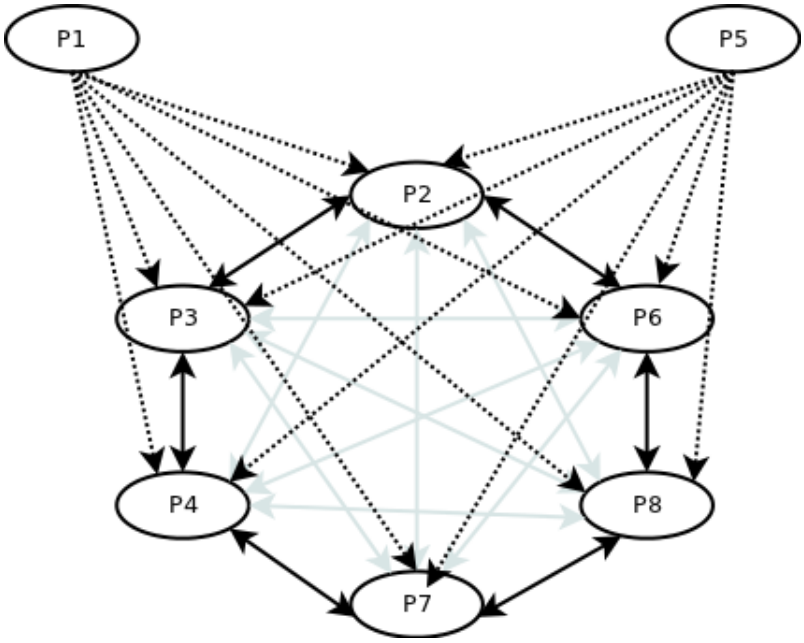
5.5.1.2.3 Rule 3: Masters Merged for all Peers

Figure 5.22a shows the initial state of a scenario with sets of parameters, where each set contains three parameters with peer-to-peer relationships and one master parameter that controls each of the peer parameters. In the first set, parameters P2, P3, and P4 are part of a peer group, while parameter P1 is master parameter to each of these peer parameters. In the second set, parameters P6, P7, and P8 are part of another peer group, while parameter P5 is master to each of these

peers. Assume that a peer-to-peer relationship is to be created between one peer parameter of the first group and another peer parameter from the second set, for example P4 and P7. Applying our rule results in a state shown in Figure 5.22b, where parameters P2, P3, P4, P6, P7, and P8 are merged to form one peer group, while each of the peer parameters is under that control of two master parameters, namely P1 and P5.



(a) Before



(b) After

Figure 5.22: Creating a Peer-to-Peer Relationship across Two Peer Groups

5.5.2 Inferring Relationships Types

5.5.2.1 Scenario

Up to this point, we have described two fundamental relationship types, namely master-slave and peer-to-peer, as well as their relative/absolute forms. Section 5.5.1 describes how hybridisation of master-slave and peer-to-peer relationships is achieved. During the course of implementation, further investigation was carried out to determine whether absolute relationships can coexist with relative relationships. As mentioned in Section 5.5.1, the introduction of one new relationship can result in a number of other relationships between various parameters being created as a side effect. An analysis of the scenarios that result in additional relationships being created revealed a common pattern, where relationships are known between one parameter, say P1, and two other parameters, say P2 and P3, while the relationship between P2 and P3 is unknown. These scenarios result from:

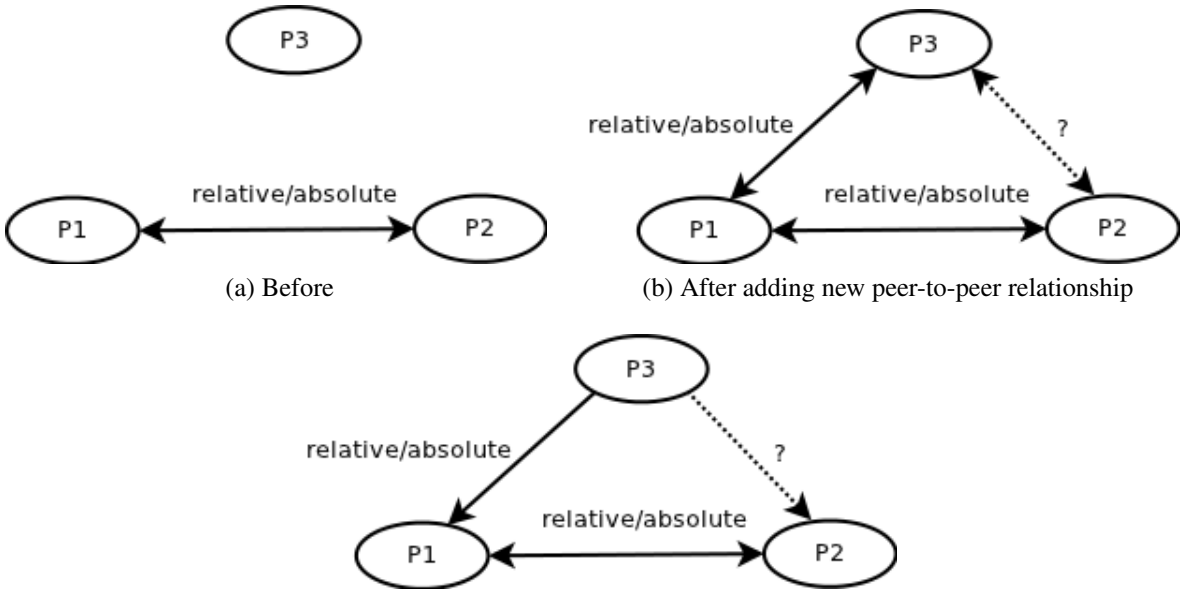
- The addition of a new peer parameter to an existing peer group.
- The addition of a new master parameter to a slave parameter that is part of a peer group.

We now describe these scenarios with the aid of diagrams shown in Figure 5.23.

Figure 5.23a shows an initial state with parameters P1 and P2 sharing a relative or absolute peer-to-peer relationship. In addition, the figure shows another parameter, P3, that has no relationships with any other parameter. In this state, two alternatives exist, namely to create a peer-to-peer relationship or master-slave relationship between either P1 or P2, and P3.

We begin by exploring the first scenario, where a relative or absolute peer-to-peer relationship is created between parameters P3 and P1. Recall from Section 5.3.1 that when a peer-to-peer relationship is created between two parameters, the net result is a merge of the peer groups from both parameters. Figure 5.23b shows the final state after the peer-to-peer relationship between P1 and P3 is made. Notice that at this point, the relative or absolute nature of the relationships between parameters P1 and P2, and between P1 and P3 are known. However, the relative or absolute nature of the relationship between parameters P3 and P2 is unknown (indicated by a "?" in the figure).

We now explore the second scenario, where a relative or absolute master-slave relationship is created between parameter P3 and P1. In this scenario, assume P3 becomes master to P1. Section



(a) Before

(b) After adding new peer-to-peer relationship

(c) After adding new master-slave relationship

Figure 5.23: Addition of a New Parameter Relationship to Existing Peer Group

5.5.1.2.1 introduced a rule which prescribes that when a master-slave relationship is created while the slave is also part of a peer group, the master becomes master to all the slave's peer parameters. Following this rule, P3 becomes master parameter to both parameters P1 and P2. Figure 5.23c shows the final state after the master-slave relationship is created between P3 and P1. Again, notice that the absolute or relative nature of the relationship between parameters P3 and P2 is unknown (indicated by a "?" in the figure), while the relative or absolute nature of the relationships between parameters P1 and P2, and between P1 and P3 are known.

Considering the two scenarios described above a number of rules were defined in order to provide a deterministic mechanism of inferring the unknown relationships based on the known relationships. These rules are described in the next section.

5.5.2.2 Rules Defined

Four inference rules were defined to deal with all possible alternatives that result from the two scenarios described above. It was empirically proven that the rules are applicable regardless of whether the relationship is master-slave or peer-to-peer. Whenever the absolute or relative nature of a relationship between a parameter, P1, and two other parameters, P2 and P3, is known, the

relative or absolute nature of the relationship between P2 and P3 can be inferred based on the known relationships using the following rules:

Rule 1. P1 related to P2 relative
P1 related to P3 relative
Implies:
P2 related to P3 relative

Rule 2. P1 related to P2 relative
P1 related to P3 absolute
Implies:
P2 related to P3 relative

Rule 3. P1 related to P2 absolute
P1 related to P3 relative
Implies:
P2 related to P3 relative

Rule 4. P1 related to P2 absolute
P1 related to P3 absolute
Implies:
P2 related to P3 absolute

In summary, an unknown relationship is absolute if and only if the known relationships from which the inference is made are both absolute. Otherwise, the unknown relationship is relative. The two scenarios where relationships are inferred were analysed in order to prove that the rules defined above are independent of the master-slave or peer-to-peer nature of the relationships. These scenarios pertain to:

1. The creation of a peer-to-peer relationship with a parameter that has another peer-to-peer relationship.
2. The creation of a master-slave relationship, where the slave has a peer-to-peer relationship.

In order to assert the validity of the assertions made by these rules, sixteen permutations of these scenarios were identified as shown in Table 5.8.

Permutation	P1-P2 Relationship	P1-P3 Relationship	Inferred P2-P3 Relationship
1	Relative Peer-to-Peer	Absolute Peer-to-Peer	Relative Peer-to-Peer
2	Relative Peer-to-Peer	Relative Master-Slave	Relative Master-Slave
3	Relative Peer-to-Peer	Absolute Master-Slave	Relative Master-Slave
4	Relative Peer-to-Peer	Relative Peer-to-Peer	Relative Peer-to-Peer
5	Absolute Peer-to-Peer	Relative Peer-to-Peer	Relative Peer-to-Peer
6	Absolute Peer-to-Peer	Relative Master-Slave	Relative Master-Slave
7	Absolute Peer-to-Peer	Absolute Master-Slave	Absolute Master-Slave
8	Absolute Peer-to-Peer	Absolute Peer-to-Peer	Absolute Peer-to-Peer
9	Relative Master-Slave	Relative Peer-to-Peer	Relative Master-Slave
10	Relative Master-Slave	Absolute Peer-to-Peer	Relative Master-Slave
~~11~~	~~Relative Master-Slave~~	~~Absolute Master-Slave~~	~~N/A~~
~~12~~	~~Relative Master-Slave~~	~~Relative Master-Slave~~	~~N/A~~
13	Absolute Master-Slave	Relative Peer-to-Peer	Relative Master-Slave
14	Absolute Master-Slave	Absolute Peer-to-Peer	Absolute Master-Slave
~~15~~	~~Absolute Master-Slave~~	~~Relative Master-Slave~~	~~N/A~~
~~16~~	~~Absolute Master-Slave~~	~~Absolute Master-Slave~~	~~N/A~~

Table 5.8: Permutations of Inference Rules

An analysis of these permutations was done. Permutations where the relationships between P1 and P2 is master-slave, while the relationships between P1 and P3 is also master-slave do not require any inference. Consequently, these permutations do not require any further validation, hence permutations 11, 12, 15, and 16 in Table 5.8 were disregarded. Figure 5.24 shows the master-slave relationship chains that can result from such combination of master-slave relationships, where either P2 is master of P1, while P1 is master of P3, or P3 is master of P1, while P1 is master of P2. As shown in the figure, there is no direct relationship between P2 and P3; therefore, the relationship between P2 and P3 does not need to be inferred.

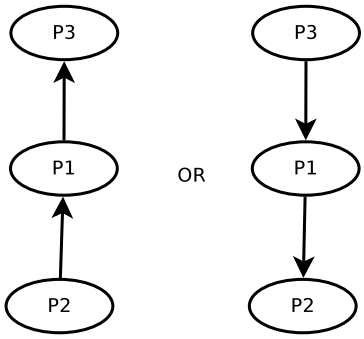


Figure 5.24: Master-Slave Relationship Chains without Inferred Relationships

Further analysis of the remaining 12 permutations under consideration in Table 5.8 reveals some duplicate permutations, namely:

- Permutations 1 and 5, shown in blue, where one relationship is relative peer-to-peer, while the other is absolute peer-to-peer.
- Permutations 2 and 9, shown in red, where one relationship is relative peer-to-peer, while the other is relative master-slave.
- Permutations 3 and 13, shown in yellow, where one relationship is relative peer-to-peer, while the other is absolute master-slave.
- Permutations 6 and 10, shown in green, where one relationship is absolute peer-to-peer, while the other is relative master-slave.
- Permutations 7 and 14, shown in brown, where one relationship is absolute peer-to-peer, while the other is absolute master-slave.

Removing duplicates and disregarded permutations leaves seven unique combinations that require validation as shown in Table 5.9.

Combination	P1-P2 Relationship	P1-P3 Relationship	Inferred P2-P3 Relationship
1	Relative Peer-to-Peer	Absolute Peer-to-Peer	Relative Peer-to-Peer
2	Relative Peer-to-Peer	Relative Master-Slave	Relative Master-Slave
3	Relative Peer-to-Peer	Absolute Master-Slave	Relative Master-Slave
4	Relative Peer-to-Peer	Relative Peer-to-Peer	Relative Peer-to-Peer
5	Absolute Peer-to-Peer	Relative Master-Slave	Relative Master-Slave
6	Absolute Peer-to-Peer	Absolute Master-Slave	Absolute Master-Slave
7	Absolute Peer-to-Peer	Absolute Peer-to-Peer	Absolute Peer-to-Peer

Table 5.9: Combinations of Inference Rules Validated

Detailed descriptions of the validations for each combination are given in Appendix B.

5.5.3 Retention of Permanent Relationships

5.5.3.1 Scenario

In a typical distributed multimedia network, it is normal for one or more devices to assume the role of controllers, thereby providing a means by which users can access and modify device

parameters. Controllers are usually computer workstations, however, any device on the network can be a controller. For illustration purposes, we shall assume that a controller is always a computer workstation containing a number of graphical controls, such as faders, pots, and meters. In the context of XFN, these graphical controls are referred to as *deskitems*. A deskitem is defined as a graphical control that is bound to an XFN parameter.

Figure 5.25 shows a typical controller-device layout, where the controller contains a graphical fader (slider) control that is associated with an XFN parameter, C1, while the device contains an XFN parameter, P1, that is associated with a DSP property of a device, such as gain. In this configuration, parameter C1 assumes the role of a remote control of parameter P1. Any changes in the value of C1 are mirrored in P1, and vice versa. Therefore, a permanent link should be maintained between C1 and P1.

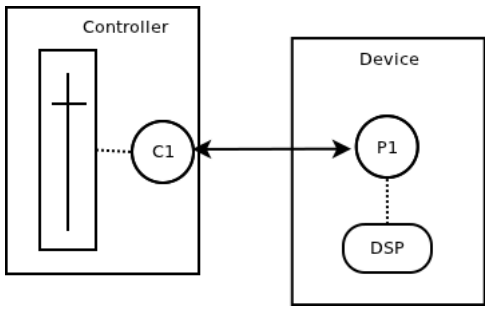


Figure 5.25: Controller-Device Layout

The initial implementation of the above scenario involved the definition of a special type of parameter relationship referred to as a *deskitem join* as shown in Figure 5.26. A deskitem join is a special type of absolute peer-to-peer relationship that can only exist between a control parameter and a device parameter, where control parameters are parameters that are bound to graphical deskitem controls, while device parameters are associated with device properties, such as gain. As shown in the figure, the device parameter can also be involved in relationships with other parameters on the same device or on the network. Having a special one-to-one relationship provides a persistent line of control between control parameters and device parameters. Therefore, this scheme implies a clear separation between control parameters and device parameters. Thus, once a device parameter is updated by a control parameter, the device parameter is responsible for updating other parameters that are affected by its change in value.

With reference to Figure 5.26 whenever the value of the device parameter, P1, is changed by some means other than via C1, the new value of P1 is copied to C1. In contrast, the following

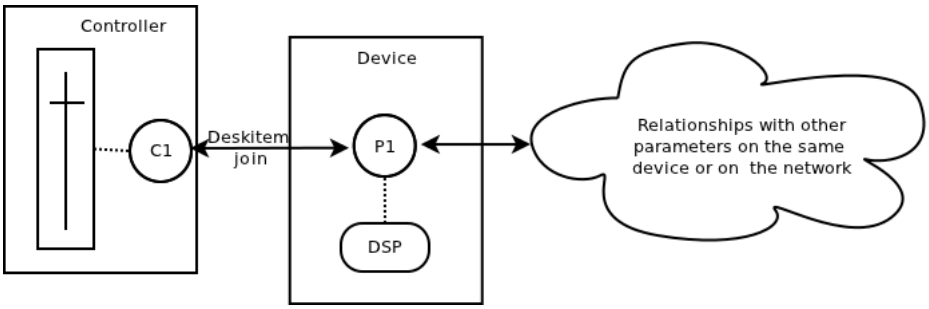


Figure 5.26: Controller-Device Layout with Deskitem Joins

sequence of events typically takes place whenever the controller changes the value of a device parameter:

1. The graphical fader control is moved on controller, causing a change in value of C1.
2. The new value of C1 is copied to P1.
3. P1 then updates parameters that it shares relationships with, namely:
   (a) Other peer parameters.
   (b) Slave parameters.
   (c) Other controller parameters, except for C1.

After some consideration of the update procedures, the propagation of updates through the additional step of copying values from P1 to C1, and vice versa, was identified as a potential source of inefficiency in the system. In particular, the device parameter, P1, becomes more like a server that is required to constantly update other "client" parameters on the network. Such centralisation of updates was identified as a drawback, particularly in resource-constrained devices. Based on this analysis, the need for a mechanism to retain as much decentralisation as possible was identified. The rules that were defined to enable this are described next.

5.5.3.2 Rules Defined

In order to overcome the potential inefficiencies described in the previous section, the special deskitem join relationship was discarded. Instead, rules were added to the peer-to-peer join mechanism to enable it to achieve the desired results in the context of controller and device

parameters. In particular, a flag that stores the permanency state of a relationship was added to each of the entries in the peer group list of a parameter. Thus, "permanent" absolute peer-to-peer relationships are used to model persistent relationships that exist between controller and device parameters. Figure 5.27 shows a layout of the relationships between a controller and device that is based on this scheme.

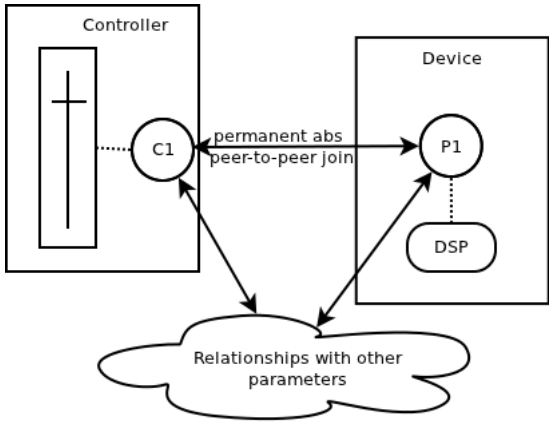


Figure 5.27: Controller-Device Layout with Permanent Absolute Peer-to-Peer Joins

For the remainder of this section, we describe the scheme that uses the permanency flag in more detail. As depicted in Figure 5.28, we assume the following:

- A device containing a parameter, P1, that is associated with a DSP property, such as gain.
- A second device containing a parameter, P2, that is associated with a DSP property, such as gain.
- A controller with a deskitem that is associated with a parameter, C1, where C1 is intended to be a mirror of P1. Thus a permanent absolute peer-to-peer relationship exists between P1 and C1.

We focus mainly on the impact of the permanency flag on peer-to-peer relationships. Thus, only the peer-to-peer group list entries for each of the parameters listed above will be considered in descriptions. Figure 5.29 shows the nature of the peer group lists for the layout with a single controller and the two devices. P2 has no peer-to-peer relationships with other parameters and only contains an entry of itself. By virtue of the absolute peer-to-peer relationship between C1 and P1, the peer group lists of parameters C1 and P1 each contain an entry of the other peer parameter. Notice how none of the flags are set, except for the permanency flag for C1's

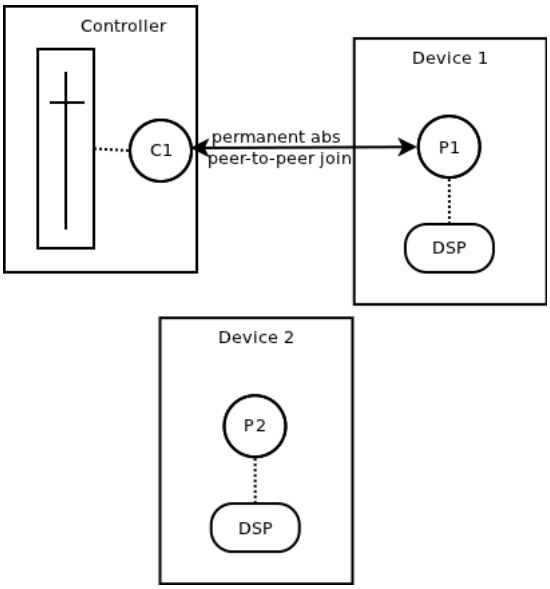


Figure 5.28: Controller with Two Devices

entry in P1's list. The rule that was defined to govern permanency of relationships states that "any parameter that has any of its group members flagged as permanent is required to maintain that relationship until such a time that the permanent group member breaks the relationship" [Chigwamba et al., 2012]. In the context of the example in Figure 5.28, P1 will always ensure that the relationship with C1 is kept, until C1 explicitly terminates the relationship.

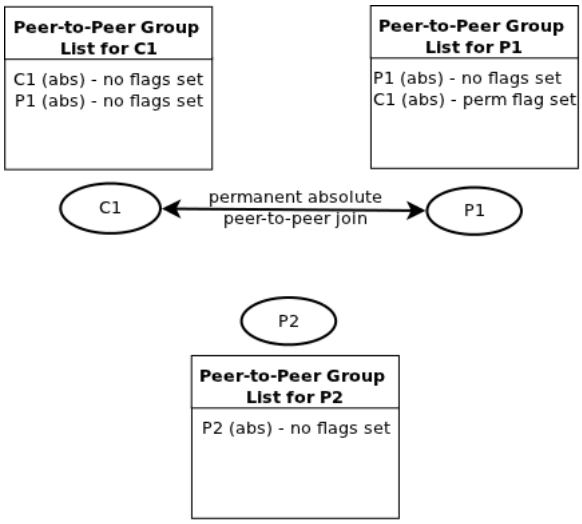


Figure 5.29: Peer Group Lists: Controller with Two Devices

The rule for permanency of relationships does not add any restrictions on the ability of parameters

to participate in additional relationships. For example, in the event that a relative peer-to-peer relationship is created between P1 and P2, the resulting peer-to-peer group list entries will be as shown in Figure 5.30. Notice how the relative or absolute nature of the relationship between C1 and P2 is inferred in accordance with the rules described in Section 5.5.2. In the event that P1 is removed from this resulting peer group containing C1, P1, and P2, P1 remains bound to C1, since there is a permanent relationship between P1 and C1.

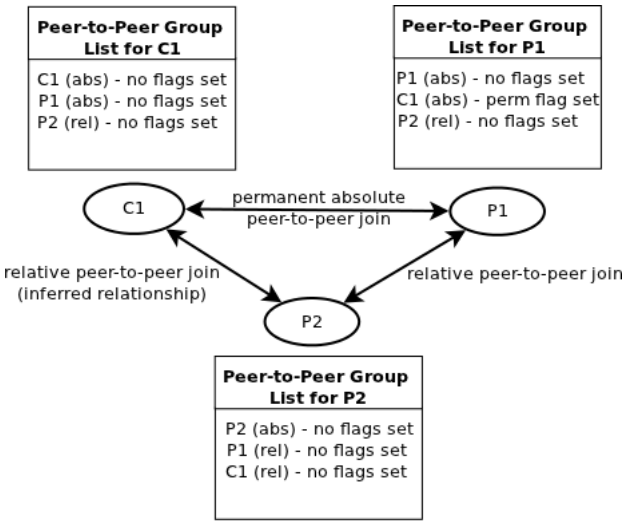


Figure 5.30: Peer Group Lists: Controller and Two Devices Including Extra Relationship

5.6 Future Work

5.6.1 Cyclic Master-Slave Relationships

Figure 5.31 on the next page has been shown in Section 5.4.1 on page 94, where a description of the handling of parameter value updates for grouped parameters is given. The figure shows six parameters, namely P1 - P6. P1, P2, and P3 have peer-to-peer relationships. P2 is the master parameter to slave parameters P4 and P5, while P4 is master parameter to P6. This illustration shows the chaining of relationships, where P2 is a grand master of P6.

The typical parameter value update process is as follows:

1. P1 receives a SET VAL (set value) command that contains a new value.
2. P1 updates its value to the one in the SET VAL command.
3. P1 sends SET GRPVAL (set group value) commands containing its new value to peer parameters, P2 and P3.
4. Upon receiving the SET GRPVAL command, P2 and P3 proceed as follows:
   (a) Change their values in a relative manner if a relative relationship exists between the parameter and P1.
   (b) Change their values in an absolute manner if an absolute relationship exists between the parameter and P1.
   (c) Send a SET GRPVAL command to each of their slave parameters. In this example, P3 has no slave parameters, while P2 has P4 and P5 as a slave parameters. Therefore, P2 sends a SET GRPVAL command to P4 and P5. P4 has P6 as a slave parameter; thus, updates its own value before sending a SET GRPVAL command to P6.

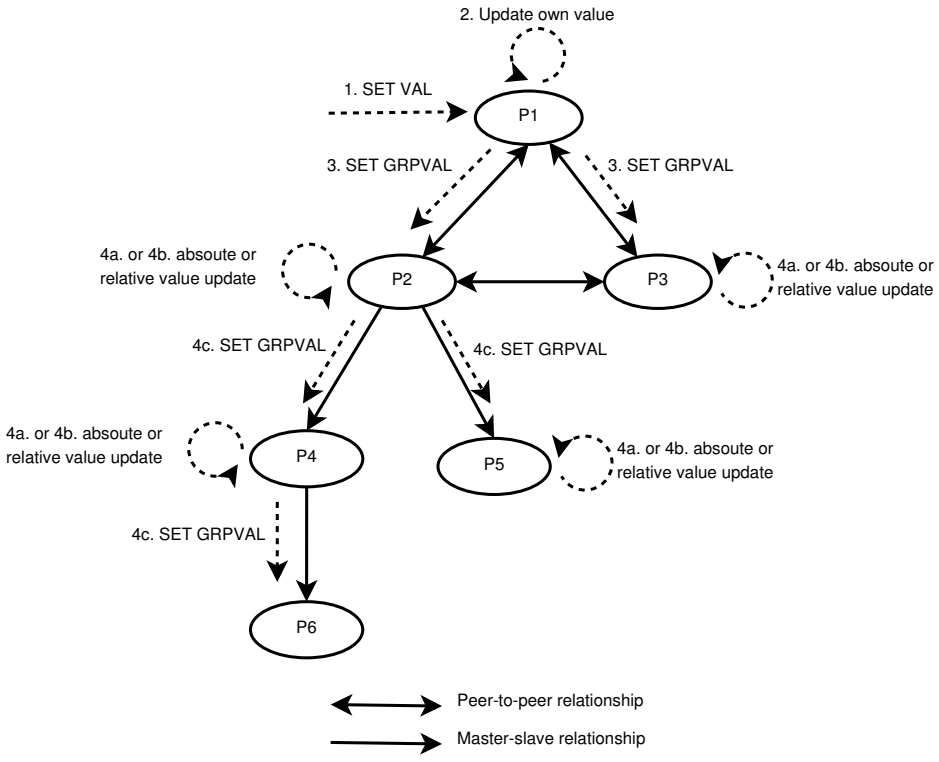


Figure 5.31: XFN Parameter Value Updates

The manner in which updates are done between master-slave relationships requires acyclic relationship chains. Cyclic master-slave relationships chains were identified as leading to undesirable infinite value change loops. Figure 5.32 illustrates an example of an unacceptable cyclic master-slave relationship. The figure builds up on the example shown in Figure 5.31. Here, P2 is master of P4, while P4 is master of P6. A relationship with P6 as master of P2 is not permissible, since it leads to an infinite loop of parameter value updates, where:

- P2 sends a SET GRPVAL command to its slave, P4.
- P4 sends a SET GRPVAL command to its slave P6.
- P6 sends a SET GRPVAL command to its slave P2, and the cycle where P2 sends another SET GRPVAL command to its slave P4 begins.

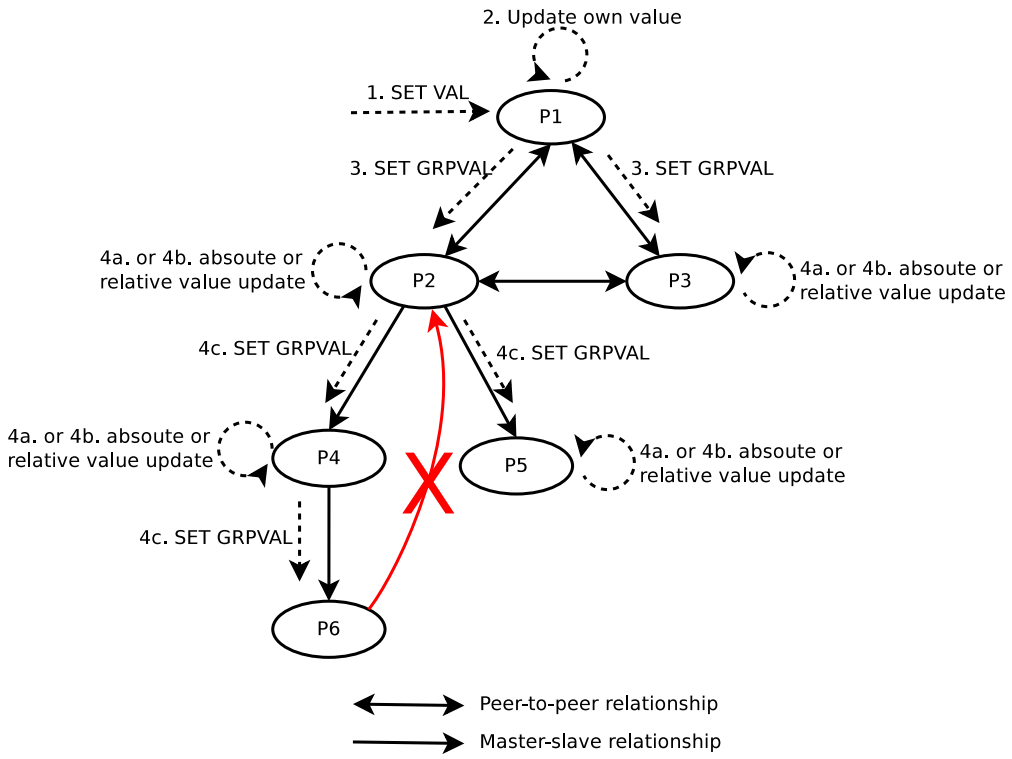


Figure 5.32: Cyclic Master-Slave Relationship Chains

The implementation strategies for parameter grouping relationships described in this chapter do not provide a mechanism to detect and prevent the creation of cyclic master-slave relationships. This leaves XFN networks prone to the undesirable effects of cyclic loops. Further work is required in order to provide an efficient mechanism that alleviates this shortcoming.

5.6.1.1 Possible Solution using Graph Theory

The problem of cycle detection has been solved in other application areas using graph theory. In the context of cyclic master-slave relationships, investigation of loop detection and prevention using *directed acyclic graphs* is recommended. By definition, a directed graph is a set of nodes known as *vertices* that are connected by *edges*, where each edge is associated with an ordered pair of vertices [Thulasiraman and Swamy, 1992]. A directed acyclic graph is a directed graph where it is not possible to start at any vertex, *v*, and after following a sequence of edges, to loop back to the initial vertex, *v*. Figure 5.33 shows an illustration of a directed acyclic graph with vertices *v1* - *v5* and edges *e1* - *e4*, where:

- *e1* is an ordered pair from *v1* to *v3*.
- *e2* is an ordered pair from *v2* to *v3*.
- *e3* is an ordered pair from *v2* to *v4*.
- *e4* is an ordered pair from *v3* to *v5*.

In graph theory, an edge is "incident out of its initial vertex and incident into its terminal vertex" [Thulasiraman and Swamy, 1992]. Using Figure 5.33 as an example, *e1* is incident out of *v1,* and *e1* is incident into *v3*. For a directed graph to be considered acyclic, at least one vertex will have an *in-degree* of zero and at least one vertex will have an *out-degree* of zero [Thulasiraman and Swamy, 1992].

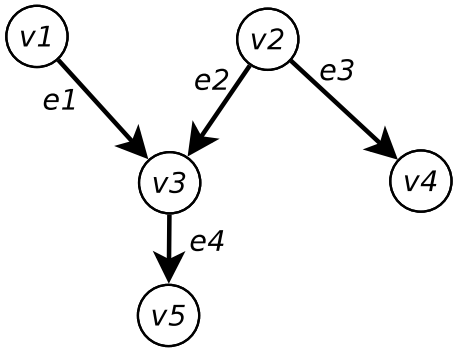


Figure 5.33: Directed Acyclic Graph Illustration

An analogy can be drawn between directed acyclic graphs and the desired master-slave relationship chain, where parameters correspond to vertices, while the unidirectional master-slave

relationships correspond to the directed edges. Some considerations to be made as part of this future investigation include, but are not limited to, the amount of time it takes to discover all relationships between parameters residing in multiple devices and scalability of the solution in large networks.

In other work, Otten [2013] models parameter relationships that have been identified in our study, although in a simulated network environment. Otten [2013] provides an analysis of a number of algorithms that can be used to determine if a graph is acyclic or not. It should be noted these algorithms have not been applied to a live network environment. These algorithms include:

- A topological sort of the nodes in a graph.
- Peeling off the leaves of a graph's tree structure.
- A depth first search of a graph.

For brevity, we give high level descriptions of the ideas behind these algorithms. Detailed descriptions of these algorithms are available in "Network Simulation for Professional Audio Networks" [Otten, 2013].

Topological Sort of Nodes in a Graph

A topological sort is the linear ordering of vertices in a graph, where each node comes before all nodes to which it has outbound edges. Given an acyclic directed graph, G, containing n vertices with integers from a set {1, 2, 3,....,n}, the topological sort of G is such that the presence of an edge (i, j), from vertex i to vertex j, implies that $i < j$ [Thulasiraman and Swamy, 1992].

Peeling off the Leaves of a Graph

This algorithm is based on the theorem that directed acyclic graphs have at least one leaf node. Leaf nodes are nodes that do not have any outbound edges. In this algorithm, leaf nodes are removed (peeled) until there are no more leaf nodes or the resulting graph is empty. If the resulting graph is empty, then the graph is acyclic, otherwise the graph is cyclic by virtue of the existence of nodes that are not leaves.

Depth First Search of a Graph

This algorithm is based on the theorem that directed graphs are acyclic if no back-edges are revealed by a depth first search of a graph. Back-edges are edges in a graph that point from a node to one of its ancestors. In this algorithm, a node is picked and all its connections are traversed, thereby marking the node as travelled. A back-edge is encountered if we travel to a marked node.

5.6.2 Usage of the Global Unit

Section 5.4.2 describes how non-additive relationships can be handled using value modifiers that were created outside this study. The uses of the value modifiers need further research and must be standardised in order to ensure consistency within networks with devices from multiple vendors. For example, the process of combining values for relative relationships by multiplication as opposed to addition must be defined and standardised.

Section 5.4.3.3 describes the use of linear and non-linear mapping tables to translate global unit values to application-specific quantities, such as dB. Further research regarding how to choose appropriate mappings to domain-specific parameters is required.

5.7 Chapter Summary

The XFN protocol was used as the basis for implementation in the course of the investigation. This chapter has given an overview of the XFN protocol, with emphasis on how the parameter grouping requirements described in Chapter 4 were fulfilled. In particular, this chapter described the use of a parameter-centric approach to grouping relationships, where each XFN parameter maintains lists containing entries that describe relationships with other XFN parameters. The chapter described the management of these lists as relationships are created and broken. The manner in which parameter value changes are handled within the groups was also described, including the use of a common value unit, known as a global unit, that can be mapped to application-specific units. Descriptions of the rules that were introduced to deal with special cases that were identified during the course of implementation have also been given. Further

work is required for the detection and prevention of cyclic master-slave relationships in live networks, possibly by employing graph theory techniques.

The next chapter focuses on the implementation strategies for an event-driven parameter monitoring mechanism that was implemented within XFN.

Chapter 6

Implementation Strategies for Parameter Monitoring

Chapter 4 laid out the requirements for parameter relationships in distributed multimedia networks. These requirements are broadly classified as requirements for grouping relationships and requirements for parameter monitoring. This chapter will describe implementation strategies for parameter monitoring within the XFN protocol. These implementation strategies are based on an implementation that was done as part of this study. In particular, a *push* mechanism that provides parameter-oriented, event-driven, single-target or multi-target messaging was implemented. This mechanism involves the transmission of bulk parameter data to "subscribed" remote parameters whenever the value of one or more selected parameters changes. Data is transmitted via unicast, multicast, or broadcast messages.

The push mechanism implementation that was done in this study leverages a mechanism for bulk XFN parameter data retrieval that is known as *Universal Snap Groups* (USG). Although the USG mechanism provides a means for bulk retrieval of parameter data, the mechanism, in its basic form, does not cater for continuous, event-driven, bulk parameter monitoring of device parameters by controllers. In order to fulfil the requirements for parameter monitoring that were identified in this study, the USG mechanism was enhanced to create a push mechanism for event-driven parameter monitoring. This chapter begins by giving an overview of the USG mechanism that exists within the XFN protocol, followed by a detailed description of the push mechanism implementation that resulted from this study. Contributions from this investigations include:

- The implementation of an event-based subscription mechanism for bulk notifications of parameter value changes by enhancing an existing bulk parameter retrieval mechanism.
- The implementation of a mechanism to integrate bulk value change notifications, via *deskitems*, into controllers.

Similar to the parameter grouping mechanisms, the push mechanism was also applied to a graphical control application known as Unos Creator. The implemented mechanisms were tested and proven in a laboratory environment. A description of the tests that were carried out will be given in Section 7.5.3 on page 208.

6.1 The USG Mechanism

The USG mechanism defined by the XFN protocol provides an efficient means to query the values of a large number of parameters. A detailed description of the mechanism can be found in the manual "Enhanced USG Mechanism" [Bradley Klinkradt, 2010]. Recall from Section 5.1 on page 79 that all XFN devices implement an XFN stack that contains a number of XFN nodes, where each XFN node contains a number of XFN parameters. In addition to XFN parameters, each XFN node contains two further components that implement the USG mechanism, namely a *USG buffer pool* and a *USG parameter cache list*, as shown in Figure 6.1. The remainder of this section describes the implementation of the USG mechanism in detail.

6.1.1 USG Mechanism Implementation Details

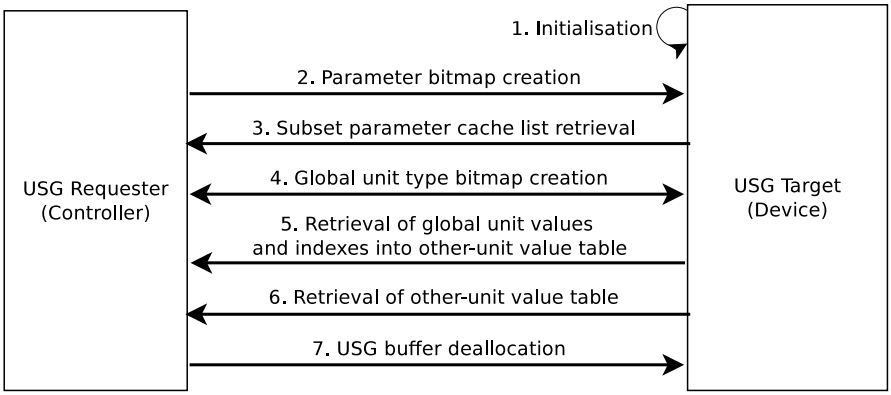


Figure 6.2: USG Mechanism: Summary of Roles and Process

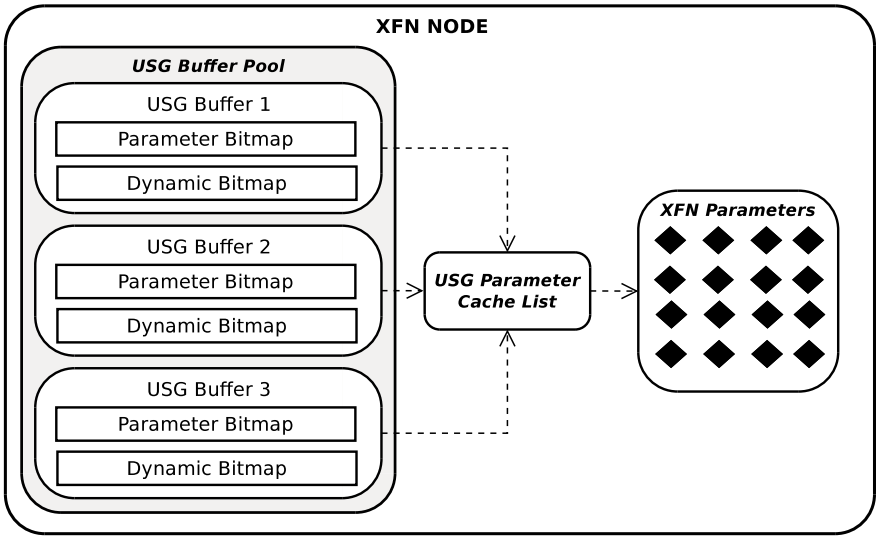


Figure 6.1: USG Mechanism Components

XFN devices that implement the USG mechanism can assume one of two roles, either *USG requester* or *USG target*. A USG requester is a device that queries bulk XFN parameter value data, typically a controller. In contrast, a USG target is a device that contains XFN parameters being queried, typically a multimedia device. It is possible for a single XFN device to assume both roles simultaneously. Figure 6.2 shows a summary of these roles and the phases involved with regards to querying bulk parameter value data, namely:

1. Initialisation
   The USG target creates a parameter cache list of all its XFN parameters and initialises a dedicated USG buffer pool.
2. Parameter bitmap creation
   The USG requester specifies a list of parameters to be queried on the USG target. The USG target allocates a free USG buffer from the pool and creates the appropriate parameter bitmap, then returns the ID of the allocated USG buffer to the USG requester. A bitmap in this context is a position-specific series of ones ('1') and zeros ('0') that corresponds to all the XFN parameters in the parameter cache list, where a '1' in the bitmap indicates that the XFN parameter has been selected as a member of the USG buffer, while a '0' in the bitmap indicates that the XFN parameter is not a part of the USG buffer.
3. Subset parameter cache list retrieval
   The USG requester retrieves the subset of the USG target's parameter cache list that re-

sults from masking the entire parameter cache list with the parameter bitmap. This subset parameter cache list is fixed and governs the rest of the USG interactions.

4. Global unit type bitmap creation
   The USG requester instructs the USG target to create a bitmap for the parameters that use the global unit value format. Based on the subset parameter cache list indicated in part 3, the USG target creates the global unit type bitmap. The created bitmap is sent to the USG requester.
5. Retrieval of global unit values and indexes into *other-unit* value table
   The USG requester retrieves a list that contains global unit values or, for parameters that do not use the global unit value format, an index into the other-unit value table.
6. Retrieval of other-unit value table
   The USG requester retrieves the other-unit value table from the USG target. The other-unit value table contains the values of parameters that do not implement the global unit value format.
7. USG buffer deallocation
   The USG requester instructs the USG target to deallocate the USG buffer for use in other USG queries by the same or other USG requesters.

We now describe each of the seven phases listed above in more detail. These descriptions will build up on an example, where the following assumptions hold:

- There are 10 parameters, namely P1, P2, P3, P4, P5, P6, P7, P8, P9, and P10.
- All odd numbered parameters make use of the global unit value format; thus, parameters P1, P3, P5, P7, and P9 use global units, while the remaining parameters use other types.
- The values of parameters P1, P2, P3, P6, and P7 are to be queried by a USG requester.

6.1.1.1 Initialisation

When the USG target starts up, its application:

- initialises the XFN stack,
- creates an XFN node,

- creates all the XFN parameters that model its application-specific properties, and
- signals the XFN stack to create a USG parameter cache list for the XFN node.

Recall from Section 5.1 on page 79 that all XFN parameters are addressed by a hierarchical address or an XFN-node-unique parameter ID. The USG parameter cache list acts like a directory that is composed of mappings from parameter IDs to hierarchical addresses for all XFN parameters in an XFN node, as shown in Table 6.1 for our example with 10 parameters. In the table, $Px_{hierarchical\ address}$ denotes a memory block that contains the hierarchical address of parameter x, while Px_{ID} denotes a memory block that contains the unique ID of parameter x.

USG Parameter Cache List		
Position index	**Hierarchical parameter address**	**Parameter ID**
0	$P1_{hierarchical\ address}$	$P1_{ID}$
1	$P2_{hierarchical\ address}$	$P2_{ID}$
2	$P3_{hierarchical\ address}$	$P3_{ID}$
3	$P4_{hierarchical\ address}$	$P4_{ID}$
4	$P5_{hierarchical\ address}$	$P5_{ID}$
5	$P6_{hierarchical\ address}$	$P6_{ID}$
6	$P7_{hierarchical\ address}$	$P7_{ID}$
7	$P8_{hierarchical\ address}$	$P8_{ID}$
8	$P9_{hierarchical\ address}$	$P9_{ID}$
9	$P10_{hierarchical\ address}$	$P10_{ID}$

Table 6.1: USG Parameter Cache List

Once the parameter cache list has been created, the USG buffer pool on the XFN node is initialised. The USG buffer pool contains a fixed number of USG buffers that are uniquely identified by IDs. Each USG buffer typically contains two bitmaps, one mandatory bitmap which flags parameters that are part of the USG buffer (the parameter bitmap) and another optional dynamic bitmap which flags parameters that match a particular criterion. These two bitmaps are shown in Figure 6.1 on page 128. For simplicity, we shall only focus on a type of dynamic bitmap that flags parameters using the global unit value format and will refer to this as the *global unit type bitmap*. The purpose of the bitmaps is to mask the parameter cache list in order to get a subset of the parameter cache list that matches various criteria. Section 6.1.1.2 will provide more details regarding the parameter bitmap, and Section 6.1.1.4 will provide more details regarding the global unit type bitmap.

6.1.1.2 Parameter Bitmap Creation

This phase marks the initial interaction between a USG requester and a USG target. The USG requester sends a SET USG_PARAM_LIST command to the USG target. This command contains:

- A valid USG buffer ID or an undefined USG buffer ID.
- One or more XFN parameter hierarchical addresses. These addresses refer to individual XFN parameters or wildcards that refer to more than one matching XFN parameter in the USG target.

The USG target adds the XFN parameters that match the hierarchical addresses to the USG buffer with the supplied USG buffer ID. In the event that the USG buffer ID is undefined, the USG target adds the corresponding XFN parameters to the next free USG buffer in the pool. In order to keep track of parameters that have been added to a USG buffer, the USG target creates a parameter bitmap that masks the parameter cache list described in Section 6.1.1.1. Considering our running example, where the USG requester queries the values of parameters P1, P2, P3, P6, and P7, the resulting parameter bitmap is shown in Figure 6.3. Any further operations on the USG buffer are only applied to parameters where the corresponding bit in the bitmap is set to '1', thereby creating the effect of a subset parameter cache list as shown in Table 6.2. The ID of the USG buffer in use is sent back to the USG requester in the response to the SET USG_PARAM_LIST command. It is important to note that once a USG buffer is allocated it becomes unavailable for other USG queries until it is explicitly freed.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	1	1	0	0	1	1	0	0	0

Figure 6.3: Parameter Bitmap

6.1.1.3 Subset Parameter Cache List Retrieval

After the parameter bitmap is created by the USG target, the USG requester sends a GET USG_DB_INDEXES command to the USG target. The command contains a USG buffer ID that is used to retrieve a copy of the subset USG parameter cache list for the corresponding USG

Subset USG Parameter Cache List		
Position index	**Hierarchical parameter address**	**Parameter ID**
0	$P1_{hierarchical\ address}$	$P1_{ID}$
1	$P2_{hierarchical\ address}$	$P2_{ID}$
2	$P3_{hierarchical\ address}$	$P3_{ID}$
3	$P6_{hierarchical\ address}$	$P6_{ID}$
4	$P7_{hierarchical\ address}$	$P7_{ID}$

Table 6.2: Subset USG Parameter Cache List

buffer. The response to the GET USG_DB_INDEXES command contains a subset parameter cache list that comprises hierarchical-address-to-parameter-ID mappings for only the parameters that are included in the parameter bitmap. Table 6.2 shows the subset parameter cache list that is retrieved for our example.

6.1.1.4 Global Unit Type Bitmap Creation

Once the subset parameter cache list has been retrieved by the USG requester, the USG requester has knowledge of the hierarchical addresses and parameter IDs of the parameters available in the USG buffer on the USG target. At this point there will not be any information that describes the value formats used by these parameters. It is possible for parameters that belong to the same USG buffer to use different value formats. Various value formats have varying storage memory requirements. The USG mechanism defines a mechanism for efficiently retrieving values that implement the same value format. This investigation focuses mostly on parameters that use the global unit value format, thus we will focus on this type to illustrate how values are retrieved.

The USG requester sends a CREATE USG_BITMAP command to the USG target. This command contains the ID of the USG buffer for which a dynamic bitmap is to be created, as well as an indication of the type of bitmap required. A predefined set of USG bitmap types is defined by the XFN protocol. For illustration purposes, we focus on the global unit type bitmap. The global unit type bitmap contains a value of '1' in the case of global unit value formats and a value of '0' otherwise. It should be noted that the bitmap is created only for the parameters that are part of the subset parameter cache list. In the context of our running example, we mentioned the assumption that all odd numbered parameters (P1, P3, P5, P7, and P9) use global unit values. Section 6.1.1.2 has also described how the parameter bitmap is created, resulting in a subset parameter cache list containing only parameters P1, P2, P3, P6, and P7. Figure 6.4 shows the corresponding global

unit type bitmap for this subset parameter cache list. The purpose of the global unit type bitmap will become clearer after reading Sections 6.1.1.5 and 6.1.1.6.

P1	P2	P3	P6	P7
1	0	1	0	1

Figure 6.4: Global Unit Type Bitmap after Applying Parameter Bitmap

After the global unit type bitmap is created, the bitmap is sent to the USG requester within the response message for the CREATE USG_BITMAP command. Most importantly, the order of the bitmap data is the same as the order of the hierarchical-address-to-parameter-ID mappings in the subset parameter cache list. Table 6.3 shows how the USG requester uses the bitmap to mask parameters of the USG buffer that use global unit values. At this point, the value format remains unknown in all cases where the global unit type bitmap value is '0'.

Subset USG Parameter Cache List			**Global unit type bitmap**
Position index	**Hierarchical parameter address**	**Parameter ID**	
0	$P1_{hierarchical\ address}$	$P1_{ID}$	1
1	$P2_{hierarchical\ address}$	$P2_{ID}$	0
2	$P3_{hierarchical\ address}$	$P3_{ID}$	1
3	$P6_{hierarchical\ address}$	$P6_{ID}$	0
4	$P7_{hierarchical\ address}$	$P7_{ID}$	1

Table 6.3: Subset USG Parameter Cache List Masked with Global Unit Type Bitmap

6.1.1.5 Retrieval of Global Unit Values and Indexes into Other-Unit Value Table

After retrieving the global unit type bitmap, the USG requester retrieves another table that contains either a global unit value or an index into the other-unit value table (shown in Table 6.4). This retrieval is done via a GET USG_XFN_UNIT_INDEX_VALS command that is sent to the USG target by the USG requester. The command contains the ID of the USG buffer with the table being retrieved. The global unit type bitmap described in Section 6.1.1.4 is used to determine whether a value in the table is interpreted as a global unit value or an index. The global unit value format and indexes into the other unit value table are both 32-bit values, therefore the response to the GET USG_XFN_UNIT_INDEX_VALS command contains an order-specific list of 32-bit values that represent either a global unit value or index into the other-unit value table.

Table 6.4 shows an illustration of the global unit value or index into other-unit table in the context of our running example, where $Px_{Global\ unit\ value}$denotes the global unit value of parameter x, while $Index_n$ is interpreted as an index into the other-unit value table at position n. The table illustrates how the USG buffer information is transferred from the USG target to the USG requester in a piecemeal manner. This requires the least amount of meta data to be exchanged, since the USG requester first "learns" the structure of the USG buffer layout and then queries the parameter values.

Subset USG Parameter Cache List			**Global unit type bitmap**	**Global unit value or index to other unit table**
Position index	**Hierarchical parameter address**	**Parameter ID**		
0	$P1_{hierarchical\ address}$	$P1_{ID}$	1	$P1_{Global\ unit\ value}$
1	$P2_{hierarchical\ address}$	$P2_{ID}$	0	$Index_0$
2	$P3_{hierarchical\ address}$	$P3_{ID}$	1	$P3_{Global\ unit\ value}$
3	$P6_{hierarchical\ address}$	$P6_{ID}$	0	$Index_1$
4	$P7_{hierarchical\ address}$	$P7_{ID}$	1	$P7_{Global\ unit\ value}$

Table 6.4: Subset USG Parameter Cache List with Values/Indexes

6.1.1.6 Retrieval of Other-Unit Value Table

After retrieving the "global unit value or index into other-unit table", the USG requester retrieves the other-unit value table from the USG target if there are any parameters that have the global unit type bit set to '0' in the global unit type bitmap. This is done by sending a GET USG_OTHER_UNIT_TABLE command to the USG target. The command contains the ID of the USG buffer for which the other-unit value table is required. The response to this command contains a number of value-format/value pairs. The value format dictates the amount of memory used by the value, thus allowing the USG requester to deal with various types of different sizes in a generic manner.

With reference to our running example, Section 6.1.1.5 shows that parameters P2 and P6 are the only parameters that do not use the global unit value format. Table 6.5 shows the other-unit value table that compliments the global unit value or index into other-unit value table that is shown in Table 6.4.

Other unit value table		
Index position	**Value format**	**Value**
$Index_0$	$P2_{valft}$	$P1_{value}$
$Index_1$	$P6_{valft}$	$P2_{value}$

Table 6.5: Other Unit Value Table

6.1.1.7 USG Buffer Deallocation

Recall from Section 6.1.1.2 that once a USG buffer is allocated it becomes unavailable for other USG queries until it is explicitly freed. For the duration of the USG buffer's existence, a USG requester can query the latest values of the parameters in the USG buffer by retrieving the global unit values and indexes into the other-unit value table as well as the other-unit value table. Thus, the USG mechanism makes it possible for a USG requester and a USG target to exchange meta data for parameters only once, while the USG requester can query the latest values of the parameters involved multiple times. This significantly reduces the bandwidth requirements for value updates across networks.

Once a USG buffer is no longer required, a USG requester sends a RESET USG_PARAM_LIST command to the USG target. This command contains the ID of the USG buffer that is to be "recycled" and results in the clearing of the parameter bitmap, hence clearing the subset parameter cache list. The USG buffer is then marked as available for processing other USG queries.

We have given a description of all the phases of the USG mechanism. The push mechanism was implemented using a modified version of the USG mechanism. Section 6.2 will now describe implementation details for the push mechanism, indicating how it leveraged the USG mechanism as well as highlighting enhancements that were made to the USG mechanism.

6.2 Push Mechanism

Recall from Section 4.4 on page 76 that an event-driven parameter monitoring mechanism was identified as requiring the least resources on a multimedia network. The USG mechanism described in the previous section was enhanced to produce the push mechanism. The push mechanism is used for parameter monitoring in the XFN protocol and ensures that data is only transmitted when necessary. In the descriptions that follow, the roles of controller and device have

been split as shown in Figure 6.5. Both, the device and the controller contain two distinct layers, namely an XFN stack layer and an application layer. The application layer interacts with the XFN stack layer via the XFN API (Application Programming Interface).

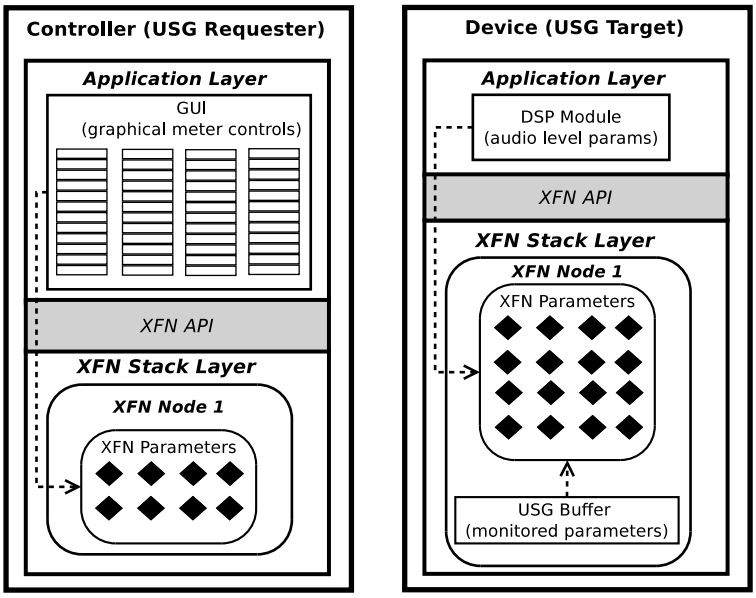


Figure 6.5: Device and Controller Roles

The application layer of a device contains a number of application-specific modules including, but not limited to, a DSP module with audio level parameters that can be monitored. The XFN stack layer of a device contains XFN parameters that model its application-specific module parameters. In addition, the XFN stack layer of a device contains one or more USG buffers that are used to monitor a subset of the device's XFN parameters. In the context of the push mechanism, we refer to parameters that are part of a USG buffer as parameters that belong to a single monitored-parameter list. For simplicity, we will focus on a device that contains one USG buffer, although multiple USG buffers are supported, hence multiple monitored-parameter lists.

In contrast, the application layer of a controller contains a graphical user interface with graphical meter controls that display the values of monitored parameters. These graphical meter controls are associated with XFN parameters in the controller's XFN stack layer. These XFN parameters mirror the values of the XFN parameters that are part of the monitored-parameter list of a device. Thus, any changes in value of monitored parameters within the device are copied to the corresponding XFN parameters in the controller, resulting in an update of the associated graphical meter control in the controller's GUI.

6.2.1 Overview

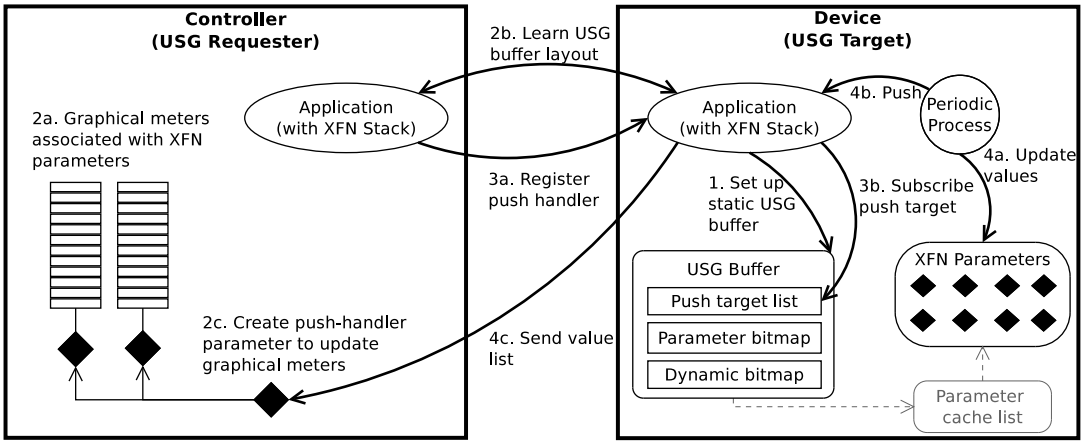


Figure 6.6: Overview of Push Mechanism

Figure 6.6 shows an overview of the push mechanism that was implemented. The diagram shows the four main steps involved in the push mechanism, namely:

1. Device initialisation
   This step involves the set up of a static USG buffer that is used for parameter monitoring. This USG buffer contains a static list of parameters that are available for monitoring across the network. The device's application defines a static list of parameters that are part of the monitored-parameter list.
2. Controller initialisation
   The controller is set up to display the values of monitored parameters. This set up involves the following three steps:
   (a) Association of graphical meter controls with XFN parameters on the controller.
   (b) Learning of a device's monitored-parameter list layout by a controller. This is achieved by learning the device's USG buffer layout.
   (c) Creation of a push-handler XFN parameter on the controller. The push-handler XFN parameter processes value updates from the device, resulting in updates of the graphical meter controls.
3. Push-handler subscription
   The push-handler XFN parameter is subscribed as a listener for value changes of any pa-

rameters that are part of the monitored-parameter list. This subscription is composed of the following two steps:

(a) A request to register the push-handler XFN parameter of the controller as a push target.

(b) The addition of the push-handler XFN parameter to the USG buffer's list of *push targets* that are listening for device parameter updates. The concept of push targets is one of the key enhancements that were made to the USG mechanism for the purpose of the push mechanism.

4. Value change events
   The handling of value changes of any of the monitored device-specific parameters, periodically. This process is composed of the following three steps:

   (a) Updating values of associated XFN parameters.

   (b) Notification of the XFN stack when parameters are updated and require pushing to associated push targets.

   (c) Transmission of a list of updated values to the controllers' push-handler parameters.

Section 6.2.2 below gives detailed descriptions pertaining to the implementation of the four main stages of the push mechanism listed above.

6.2.2 Push Implementation Details

Section 6.1 has already given a detailed description of the USG mechanism, where the following seven steps are carried out:

1. Initialisation (described in Section 6.1.1.1)
2. Parameter bitmap creation (described in Section 6.1.1.2)
3. Subset parameter cache list retrieval (described in Section 6.1.1.3)
4. Global unit type bitmap creation (described in Section 6.1.1.4)
5. Retrieval of global unit values and indexes into other-unit value table (described in Section 6.1.1.5)

6. Retrieval of other-unit value table (described in Section 6.1.1.6)
7. USG buffer deallocation (described in Section 6.1.1.7)

The push mechanism that was implemented leverages the USG mechanism. In addition, some enhancements to the USG mechanism were made in order to fulfil the requirements for parameter monitoring relationships described in Section 4.4. In the context of the push mechanism, we now describe how the USG mechanism was used, as well as highlight the enhancements that were made. The descriptions that follow will make reference to the seven steps of the USG mechanism.

6.2.2.1 Device Initialisation

In order to simplify the implementation of the push mechanism, it was assumed that a device contains one or more static monitored-parameter lists. These monitored-parameter lists include all commonly monitored XFN parameters, such as XFN parameters that are associated with device-specific audio level parameters. In order to accommodate static monitored-parameter lists, step 1 (initialisation) of the USG mechanism was modified such that a device can reserve one or more USG buffers for its sole use. Any USG buffers that are reserved for parameter monitoring cannot be changed by any controllers (USG requesters) on the network.

For each of the reserved USG buffers, the device internally proceeds to step 2 of the USG mechanism, where a parameter bitmap is created based on the corresponding static monitored-parameter list that is stored in the device. This step is typically done by a controller (USG requester). However, in the context of the push mechanism, the device (USG target) provides a single reference point regarding the parameters that are monitored as part of the reserved USG buffers. In particular, the parameters that belong to the USG buffers are statically specified by the USG target and this cannot be altered by any controller on the network.

6.2.2.2 Controller Initialisation

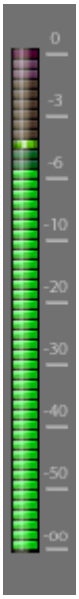
In Section 5.5.3 on page 114 we introduce the concept of deskitems, where a deskitem is defined as a graphical control that is bound to an XFN parameter. During the course of implementation, a class of deskitems known as *meter deskitem* was implemented with the intention of displaying

values of monitored parameters of an XFN device, such as audio level parameters. Figure 6.7a shows a pseudo-XML element that highlights the main XML attributes used to create the graphical audio level meter control shown in Figure 6.7b. A detailed description of the meter deskitem will be given in Section 7.2.1.5 on page 168. The pseudo-XML element in Figure 6.7a shows four main sections that describe:

1. The address of the parameter to be monitored on the remote device.
2. The appearance of the graphical meter control.
3. Mappings from global unit values to other meter-specific units.
4. Meter-specific behaviour, such as ballistics.

```
<METER_DESKITEM

<!– Address of monitored parameter –>
ipAddress="192.168.1.1" xfnNodeId="1" usgId="0" xfnAddressLevel1="ba"
xfnAddressLevel2="d1" xfnAddressLevel3="1" xfnAddressLevel4="d8"
xfnAddressLevel5="1" xfnAddressLevel6="d05c" xfnAddressLevel7="1"

<!– Graphical appearance –>
backImage="meter_back.png" barStripeImage="led_meter_stripe.png"
peakStripeImage=”led_meter_peak_stripe.png" numberOfSteps="51"

<!– Global unit value mappings –>
xfnUnitTableFile="../xfn_unit_tables/diceMeterXfnUnitTable.txt"

<!– Meter-specific behaviour –>
ballisticsEnabled="1" peakHoldMillis="1000" peakFallbackMillis="1000"
peakFallbackValue="20" barHoldMillis="0" barFallbackMillis="1000"
barFallbackValue="20"

/>
```

(a) Pseudo-XML Element



(b) Graphical Control

Figure 6.7: Example Meter Deskitem

With knowledge of the information that is provided by the meter deskitem XML element, we now describe how the controller is initialised in the context of the push mechanism.

6.2.2.2.1 Associating Controls with XFN Parameters

When a meter deskitem is loaded, a new instance of a graphical meter component is created. In order to allow for customisation of graphical meter controls, the appearance of the meter is derived from the XML attributes that pertain to the deskitem's appearance. Once meter components have been created, based on the XML file, each graphical meter component is then associated with a newly created unique instance of an XFN parameter. Recall from Section 5.1 on page 79 that there is a callback mechanism that associates an XFN parameter with application-specific behaviour. Therefore, each XFN parameter created is also associated with a dedicated application-specific callback that handles the XFN parameter's value change events. In order to associate an XFN parameter with an application-specific callback function, an XFN parameter stores the address of the callback function that is invoked upon value changes of the XFN parameter. Figure 6.8 on the following page gives a diagrammatic illustration of the process of associating graphical meter controls with XFN parameters, where:

1. XML elements are parsed to determine the properties of each graphical meter, resulting in the creation of graphical meter controls.
2. New XFN parameters are created and bound to each graphical meter control.
3. Each XFN parameter is associated with an application specific callback that will be used to handle changes in value of the parameter.

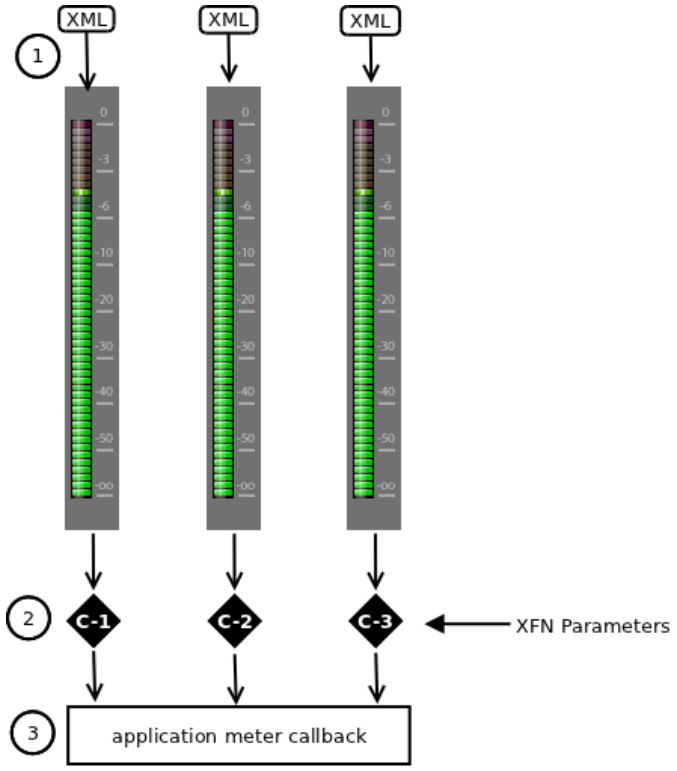


Figure 6.8: Graphical Meters with XFN Parameter Associations

6.2.2.2.2 Learning the Monitored-Parameter List Layout

Section 6.2.2.1 has given a description of a procedure for device initialisation in preparation for parameter monitoring. This device initialisation is composed of steps 1 and 2 of the USG mechanism, where one or more USG buffers are initialised and set up to include parameters on a device's static monitored-parameter lists. For simplicity, we assume that only one USG buffer is used for parameter monitoring on a remote device. All controllers are required to learn the layout of the monitored-parameter list by learning the corresponding USG buffer layout. This learning process commences after the graphical controls have been created and their corresponding XFN parameters created as described in Section 6.2.2.2.1. Each USG buffer on a remote device is identified by a unique USG buffer ID. The USG buffer ID is required by a controller in order to determine the USG buffer to learn about, hence the presence of the "usgId" XML attribute in the pseudo-XML element shown in Figure 6.7a on page 140.

The learning of the USG buffer layout involves steps 3, 4, and 5 of the USG mechanism. Step 3 of the USG mechanism results in the retrieval of an order-specific subset parameter cache list that contains hierarchical-address-to-parameter-ID mappings of all the parameters that are part

of a USG buffer as described in Section 6.1.1.3. Step 4 of the USG mechanism results in the creation of a global unit type bitmap that flags parameters of the USG buffer that use the global unit value format as described in Section 6.1.1.4. Step 5 results in the retrieval of a list of global unit values or, in the case of parameters that do not use the global unit value format, indexes into the other-unit value table as described in Section 6.1.1.5. This step enables a controller to initialise its graphical controls with the current values.

Table 6.6 shows the information that is retrieved by a controller after completing the USG buffer learning process described above. Notice how the hierarchical-address-to-parameter-ID mappings are matched with the global unit type bitmap and the global unit value or index to other-unit value table. The position index has been greyed out in the table, since this is not stored as part of the data that is learnt. However, the position index is shown to emphasise the importance of list order.

Subset USG Parameter Cache List			**Global unit type bitmap**	**Global unit value or index to other unit table**
Position index	**Hierarchical parameter address**	**Parameter ID**		
0	$P1_{hierarchical\ address}$	$P1_{ID}$	1	$P1_{Global\ unit\ value}$
1	$P2_{hierarchical\ address}$	$P2_{ID}$	1	$P2_{Global\ unit\ value}$
2	$P3_{hierarchical\ address}$	$P3_{ID}$	1	$P3_{Global\ unit\ value}$
3	$P4_{hierarchical\ address}$	$P4_{ID}$	0	$Index_0$
4	$P5_{hierarchical\ address}$	$P5_{ID}$	1	$P5_{Global\ unit\ value}$
:	:	:	:	:
:	:	:	:	:
x	$P(x+1)_{hierarchical\ address}$	$P(x+1)_{ID}$	1	$P(x+1)_{Global\ unit\ value}$

Table 6.6: USG Buffer Learning Process Results

6.2.2.2.3 Handler Configuration

Once the ordering of the monitored-parameter list has been learnt by the controller, the controller initialisation process is completed by the creation of an XFN parameter that is responsible for handling monitored-parameter value updates. We refer to this parameter as the *push-handler parameter*. The manner in which value updates are processed by the push-handler parameter will be described in Section 6.2.2.4. As with all XFN parameters, the push-handler parameter is associated with a callback that executes application-specific logic upon receipt of the monitored-parameter value lists. At the time of handler configuration, the following information is available to the controller:

- XML-derived graphical meter deskitem controls that are bound to XFN parameters, as described in Section 6.2.2.2.1.
- An ordered list of parameters that form part of the monitored-parameter list, as described in Section 6.2.2.2.2.

Using the information available, the controller creates another ordered list that associates a monitored-parameter with one or more deskitem parameters. As shown in Figure 6.7a on page 140, the XML element of a meter deskitem contains the hierarchical address of the parameter that it monitors on the remote device, namely XML attributes 'xfnAddressLevel1' up to 'xfnAddressLevel7'. By matching the hierarchical address in the XML element of a meter deskitem with the hierarchical addresses of the monitored-parameter list entries, such as those shown in Table 6.6, it is possible to match controller parameters with the appropriate monitored parameters in the remote device. We illustrate this using an example with:

- Four deskitems with graphical controls, where each graphical control is associated with one of four XFN parameters on the controller, namely C1, C2, C3, and C4.
- Six parameters on the remote device, namely P1, P2, P3, P4, P5, and P6, where:
    - The deskitem bound to parameter C1 monitors parameter P1.
    - The deskitem bound to parameter C2 monitors parameter P2.
    - The deskitems bound to parameters C3 and C4 monitor parameter P3.
    - Parameters P4, P5, and P6 are not monitored.

Table 6.7 shows the resulting ordered list showing how entries of a USG buffer are mapped to controller parameters. Notice how the list contains all parameters in the remote device's monitored-parameter list, including parameters that are not monitored by the controller. This is due to the fact that predefined monitored-parameter lists are defined on the remote devices. Thus, each controller subscribes to receive value change notifications for one or more of the monitored parameters.

Subset USG Parameter Cache List			Global unit type bitmap	Global unit value or index to other unit table	Associated controller parameters
Position index	Hierarchical parameter address	Parameter ID			
0	$P1_{hierarchical\ address}$	$P1_{ID}$	1	$P1_{Global\ unit\ value}$	C1
1	$P2_{hierarchical\ address}$	$P2_{ID}$	1	$P2_{Global\ unit\ value}$	C2
2	$P3_{hierarchical\ address}$	$P3_{ID}$	1	$P3_{Global\ unit\ value}$	C3, C4
3	$P4_{hierarchical\ address}$	$P4_{ID}$	1	$P4_{Global\ unit\ value}$	-
4	$P5_{hierarchical\ address}$	$P5_{ID}$	1	$P5_{Global\ unit\ value}$	-
5	$P6_{hierarchical\ address}$	$P6_{ID}$	1	$P6_{Global\ unit\ value}$	-

Table 6.7: Controller-Parameter to Monitored-Parameter Mapping List

Figure 6.9 shows an example illustration of the states of the controller and device after the handler configuration stage, where:

- There are four controller parameters (C1, C2, C3, and C4) that are bound to graphical meter controls. Each of these XFN parameters is associated with an application-specific meter callback that handles value updates of the graphical meter controls.
- On the device, there are six parameters (P1, P2, P3, P4, P5, and P6) that make up the monitored-parameter list for the USG buffer with an ID of 1.
- The controller has a push-handler parameter, H1, that receives notifications upon value changes of any of the monitored parameters belonging to USG buffer 1 on the device. H1 is bound to the mapping table shown in Table 6.7. The mapping table has knowledge of all parameters on the monitored-parameter list of USG buffer 1 on device, and maps these to the corresponding controller parameters.

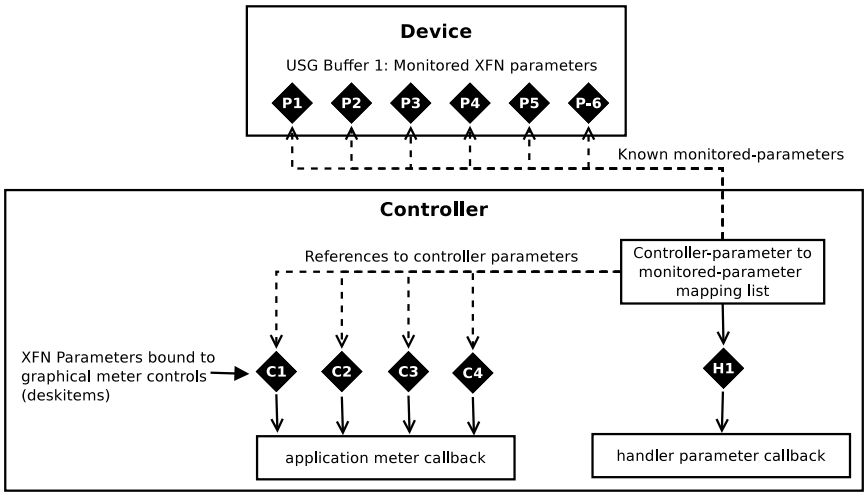


Figure 6.9: Device and Controller State after Handler Configuration

We mention in passing here that each separate USG buffer that is monitored by a controller requires its own dedicated push-handler parameter created in the controller. For instance, if the device in Figure 6.9 had a USG buffer with an ID of 2, then another push-handler parameter (H2) would have been created by the controller to handle value changes for USG buffer 2.

The above descriptions are based on the assumption that there are predefined USG buffers in a device. Once the push-handler parameter and the deskitem parameters have been set up in the controller, the next step involves the controller subscribing the push-handler parameter as a listener for the value changes of any of the parameters in the monitored-parameter list. This will be described in Section 6.2.2.3 below.

6.2.2.3 Subscriber Registration

Section 6.2.2.2 has described how a controller is configured to handle push notifications from a device whenever the value of any of the monitored parameters changes. This configuration process involves learning the USG buffer layout of a device and mapping it to a controller's parameter layout which includes meter deskitems. However, there has been no mention of any configuration on the device side that makes a device aware of the controller. We now look at the configuration of the device in order for push notifications to be "pushed" (sent) to the corresponding controllers.

Figure 6.10 shows the layout of an XFN device that uses USG buffers for parameter monitoring. We focus on the components of a USG buffer, where the USG buffer layout was enhanced by adding a push target list structure. The push target list keeps track of all the push-handler parameters that are listening for changes in value of any of the monitored parameters that are part of the USG buffer. Only targets that are part of the push target list receive push notifications.

A mechanism for the subscription of push targets was implemented. Here, a controller sends a SET PUSH command that targets the appropriate XFN node in a device. Table 6.8 shows each of the fields included in the command. Upon receiving the SET PUSH command, a device adds a new timestamped push target entry to the push target list of the appropriate USG buffer if there is no existing entry. If an entry exists for the push target, its timestamp field is updated to reflect the last time that a SET PUSH command was received for the given push target.

The timestamping of push target entries was introduced after an analysis of a possible scenario, where a target leaves the network without successfully unsubscribing its push-handler parameter

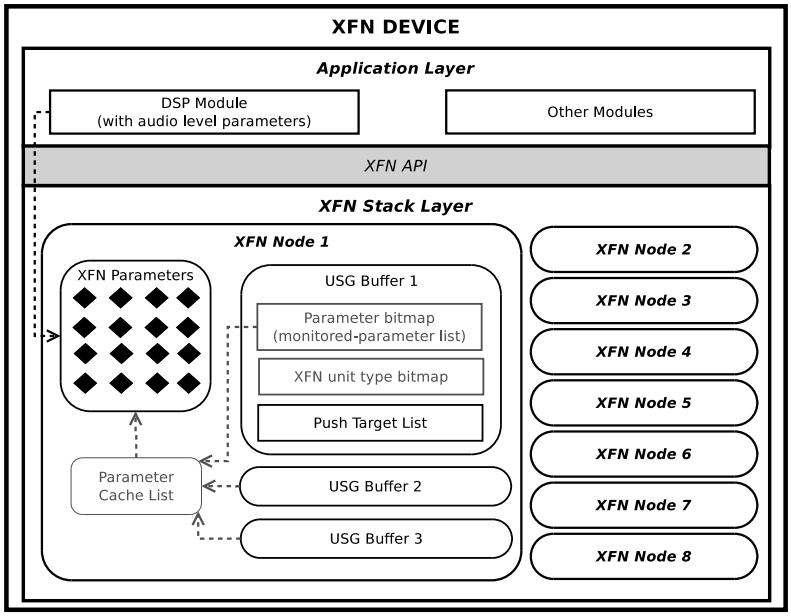


Figure 6.10: XFN Device Layout for Monitored Parameters

Field	Size	Description
USG Buffer Id	32-bit	The unique identifier of the targeted USG buffer.
Target IP Address	32-bit	The IP address of the push-handler XFN parameter. Typically the IP address of the controller.
Target XFN Node ID	32-bit	The unique XFN node identifier for the XFN node that contains the push-handler XFN parameter
Target parameter ID	32-bit	The unique parameter ID of the push-handler XFN parameter.

Table 6.8: SET PUSH Command Structure

from the push target list of a given USG buffer. Such a scenario could potentially result in an unnecessary flood of the network, especially in situations where monitored-parameter value changes are pushed at high rates, such as every 25 ms for audio level meters. In order to minimise the negative effects of controllers failing to unsubscribe their push-handler parameters, each controller is required to periodically send a SET PUSH command that results in an update of its timestamp. The device will only push updates to the subscribed targets on condition that the timestamp was last updated within a specified time interval. It should be noted that, at the time of this writing, this time interval was statically set to 10 seconds. Minimal work is required in

order to make this interval configurable by a controller. In particular, a validity field is required in the SET PUSH command, and this field would be associated with a given push target entry.

In order to unsubscribe a push target, the controller sends a SET PUSH_OFF command that targets the appropriate XFN node on the device. The SET PUSH_OFF command contains all the fields shown in Table 6.8 for the SET PUSH counterpart. Upon receiving the SET PUSH_OFF command, the device removes the matching push target entry from the push target list of the appropriate USG buffer.

Section 6.2.2.4 now describes how value changes are pushed to subscribed push targets.

6.2.2.4 Value Change Events

There is a periodic process that runs in the application layer of a device. This process monitors the values of application-specific properties that affect any of the parameters on the static monitored-parameter lists, hence affecting parameters that belong to one or more USG buffers. The process runs at various intervals depending on what is being monitored and the level of accuracy required for the monitored properties. For example, audio level metering data could require that the process runs at 25 millisecond intervals, while temperature monitoring could require that values are sent at 1 second intervals. Whenever the value of any of the application-specific properties changes, the periodic process first updates the affected XFN parameters, then initiates a push notification for any USG buffers that contain XFN parameters with value changes.

For all buffers with pending push notifications, the XFN stack builds an updated value table and sends the value table to the registered push-handler parameters via SET USG_XFN_UNIT_INDEX_VALS commands. The format of the value table is identical to the value table retrieved by USG requesters in step 5 of the USG mechanism. In particular, the value table is a list of global unit values or, in the case of parameters that do not use the global unit value format, indexes into the other-unit table as described in Section 6.1.1.5; this is the reason why it is important for a controller to learn the layout of USG buffers as described in Section 6.2.2.2.2. The values of all monitored parameters are sent in the value table, regardless of the number of parameters that have actually changed values. Recall from Section 6.2.2.2.3 that each push-handler parameter in a controller contains mappings that allow the push-handler parameter to associate values in the list with the appropriate deskitem parameters, resulting in an update of the associated graphical components. One of the main reasons for sending positional-offset-based value lists for updates

was to reduce the amount of bandwidth required for value updates by avoiding the transmission of all parameter meta data, such as hierarchical addresses, together with the updated values.

Figure 6.11 shows a summary of the process of handling value changes, where:

1. The periodic process on a device reads the new values of the application-specific properties being monitored.
2. For each property with a changed value, the periodic process updates the corresponding XFN parameter via the XFN API.
3. The periodic process initiates a push notification for any USG buffers that contain XFN parameters with value changes.
4. The device's XFN stack, constructs a list that contains the current values of all the monitored parameters in the USG buffer and sends this list to each of the subscribed push-handler parameters.
5. Upon receiving a push update, each push-handler parameter updates any of its deskitem parameters based on the information retrieved when the USG buffer structure is learnt as described in Section 6.2.2.2.2.

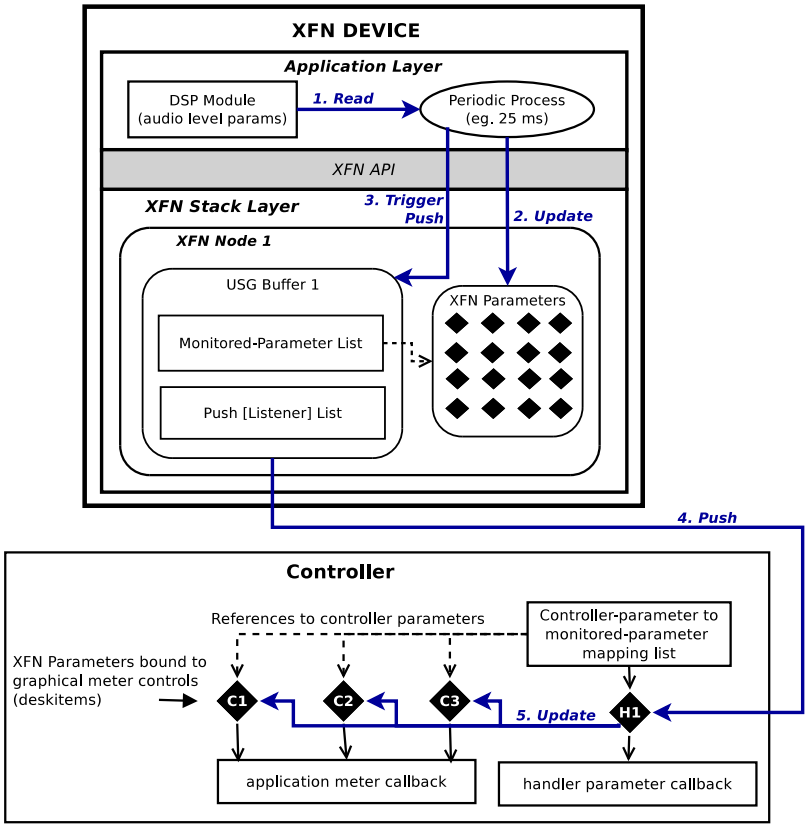


Figure 6.11: Periodic Push Processing

6.3 Fulfilment of Requirements for Parameter Monitoring

The requirements for parameter monitoring described in Section 4.4 emphasised the importance of event-driven monitoring and clustering of multiple value change notifications into a single message. The push mechanism that was implemented in this investigation fulfils these requirements. The mechanism is event-driven and all push notifications are sent to controllers upon changes in values of monitored parameters. When compared to existing polling and semi event-driven monitoring capabilities of the AV/C protocol described in Section 3.3.2.4, such event-driven parameter monitoring bears the least overhead.

The push mechanism enables bulk monitoring of parameters, where the structure of the monitored parameter lists on devices is “learned” dynamically. Section 3.2.4 describes the implementation of status pages in IEC 62379. Although status pages can be viewed as a mechanism for bulk monitoring of parameters, their layout is static and monitoring devices (controllers) must have prior knowledge of the layout of monitored status pages. In contrast, the push mechanism

allows bulk monitoring of parameters, where the layout of the monitored parameter lists (USG buffers) is "learned" dynamically. Such a mechanism for bundling multiple notifications into a single message is currently not defined in OCA (described in Section 3.5.3).

The push mechanism uses global units (defined in Section 5.4.3) as the generic value type for monitored parameters. The use of global units to represent the values of monitored parameters enables client applications to generically monitor any XFN parameters regardless of their underlying value types. This capability resulted in the implementation of a meter deskitem type that is used to display the values of any monitored XFN parameter. None of the existing multimedia technologies described in Chapter 3 provide a mechanism that enables such generic parameter monitoring of networked parameters.

6.4 Future Work

6.4.1 Dynamic Determination of Monitored-Parameter Lists

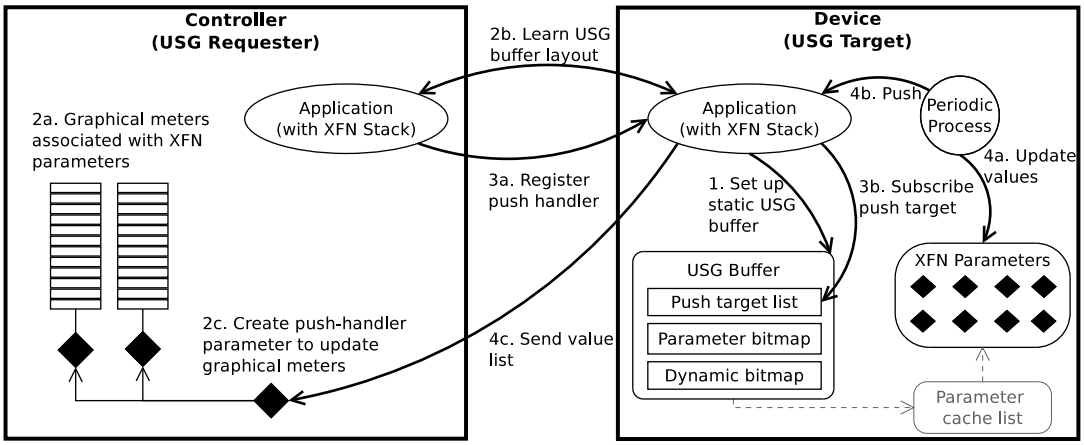


Figure 6.12: Overview of Push Mechanism

Section 6.2.1 on page 137 gives an overview of the implementation of the push mechanism. The overview uses the diagram duplicated in Figure 6.12 to describe the four main steps of the push mechanism. These four steps include:

1. Device initialisation
   Setting up one or more static USG buffers that contain information about all the parameters

that are available for monitoring on a multimedia device. This set up is done when a device starts up and cannot be changed.

2. Controller initialisation
   Loading graphical controls that display the values of the monitored parameters, setting up XFN parameters to handle value updates on the controller, and learning the layout of one or more static USG buffers defined in step 1.
3. Push-handler subscription
   Subscribing an XFN parameter (push-handler parameter) as a listener for all changes in values of the monitored parameters belonging to a particular static USG buffer. The push-handler is the parameter that receives notifications upon value changes of any of the monitored parameters in the given USG buffer.
4. Value change events
   The values of the monitored parameters are periodically monitored and updated. The latest values of the monitored parameters are pushed to the subscribed push-handler parameters on one or more controllers across the network.

Step 1 in the description above involves the static set up of USG buffers containing all information about the monitored parameters. In the implementation of the push mechanism, these lists of monitored parameters were made static in order to make it possible for multiple controllers to monitor a common set of parameters on each device. In particular, multimedia devices define all the parameters that are available in their monitored-parameter lists, while controllers on the network can only choose to monitor parameters from the static set of available parameters. A shortcoming of this approach is that it is not possible for controllers to dynamically specify monitored parameters. We illustrate this next with the aid of an example.

For illustration purposes, Figure 6.13 shows an example of three controllers networked with a single multimedia device, where:

- There are eight XFN parameters on the multimedia device labelled P1 to P8.
- P1 - P4 belong to the monitored-parameter USG buffer.

The current implementation of the push mechanism is such that all three controllers can only monitor parameters that are part of the static USG buffer, namely P1, P2, P3, and P4. Each

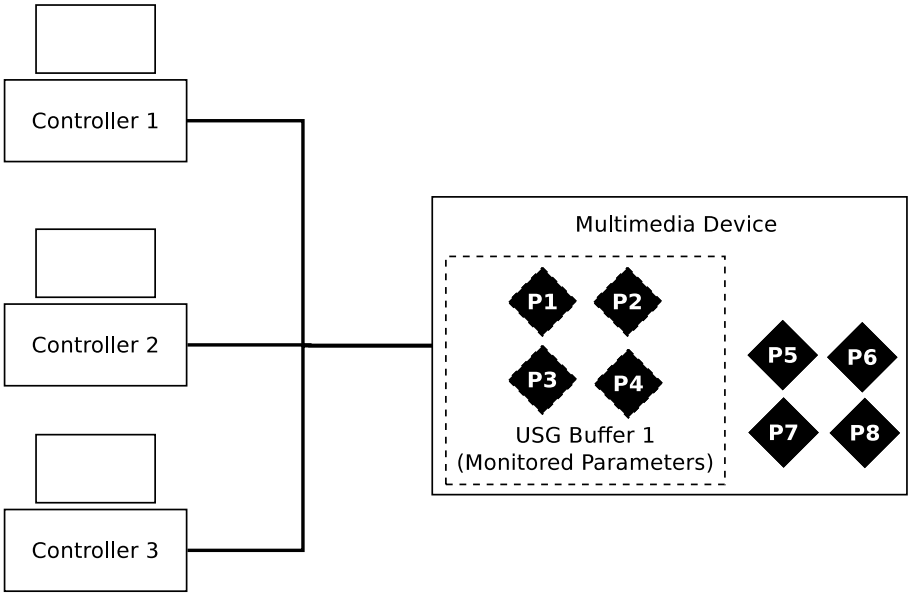


Figure 6.13: Multiple Controllers Monitoring Parameters on a Single Device

controller learns the ordering of parameters in the static USG buffer and stores this ordering as shown in Table 6.9a. The latest values of the monitored parameters are periodically sent to each of the controllers in the order in which they appear in the USG buffer as shown in Table 6.9b.

List position	Parameter
1	P1
2	P2
3	P3
4	P4

(a) Layout Ordering Cached by Controllers

List position	Value
1	Value_{P1}
2	Value_{P2}
3	Value_{P3}
4	Value_{P4}

(b) Value Update Lists Sent to each Controller

Table 6.9: Monitored Parameter USG Buffer

Further work is required in order to make it possible for any controller to add or remove parameters to or from the monitored-parameter USG buffers. This altering of the USG buffer layout implies an update in the ordering of value update lists that are sent to each of the subscribed push-handler parameters in one or more controllers. Therefore, all subscribed controllers will need to update their cached layout information in order to match the new buffer layout. Based on the illustration in Figure 6.13, if controller 3, for example, removes P3 from the USG buffer and adds P5 and P7 to the monitored USG buffer, all controllers will require re-initialisation based on the new layout shown in Table 6.10a. The multimedia device then sends value updates in

the order shown in Table 6.10b. In addition to re-initialisation, rules will be required to guard against conflict scenarios, where one controller removes parameters that are relevant to another controller.

List position	Parameter
1	P1
2	P2
3	P4
4	P5
5	P7

(a) Modified Layout Ordering Cached by Controllers

List position	Value
1	$Value_{P1}$
2	$Value_{P2}$
3	$Value_{P4}$
4	$Value_{P5}$
5	$Value_{P7}$

(b) Modified Value Update Lists Sent to each Controller

Table 6.10: Modified Monitored Parameter USG Buffer

6.5 Chapter Summary

This chapter has shown how the requirements for an event-driven parameter monitoring mechanism have been fulfilled within the XFN protocol. The chapter began by describing the USG mechanism that is defined by the XFN protocol for bulk parameter data retrieval. The USG mechanism was used as the basis for the implementation of an event-driven parameter monitoring mechanism known as the push mechanism.

The push mechanism relies on the existence of static monitored-parameter lists on devices. These monitored-parameter lists are used to initialise USG buffers within a device's XFN stack. Controllers on the network first learn the structure of the USG buffers, then subscribe their designated push-handler parameters as push targets of one or more USG buffers in a device. Upon changes in value of any of the monitored parameters on a device, push notifications are sent to all subscribed push targets. Push notifications contain a list of the latest values for all parameters that are part of a USG buffer. Thus, value lists are only sent to push targets when the values of one or more of the monitored parameters have changed. Such an approach leads to a reduced network load. The push mechanism implementation also took into account the need to keep track of the presence of push targets on the network in order to prevent the unnecessary transmission of push notifications to targets that are no longer on the network.

Further work is required to make the definition of monitored-parameter lists dynamic. This will enable more flexible parameter monitoring, since controllers would add or remove parameters to and from the parameter monitoring USG buffers.

The next chapter will describe the usability of parameter grouping and monitoring relationships in the context of an example graphical workstation application that was enhanced in the course of implementation.

Chapter 7

An XFN Control Application Example

Chapter 4 lays out the requirements for parameter relationships, namely parameter grouping and monitoring relationships. In the context of the XFN protocol, Chapter 5 describes the implementation mechanisms that were created to fulfil the requirements for parameter grouping relationships, while Chapter 6 describes the implementation mechanisms that were created to fulfil parameter monitoring relationships. These implementation mechanisms were applied to an XFN-protocol-compliant graphical control application known as *Unos Creator* [Universal Media Access Networks GmbH, 2010]. Unos Creator is used for multimedia device connection management, command, and control.

Prior to this investigation Unos Creator capabilities focused mainly on the aspects outside the scope of this investigation, namely multimedia device discovery, static network configuration, and stream connection management. There were no mechanisms for parameter grouping and monitoring. The only form of remote control was the ability to set the value of a remote parameter from the Unos Creator GUI in a unidirectional manner, where changes in values of remote parameters, by means other than Unos Creator, were not reflected in Unos Creator. A rudimentary mechanism for parsing XML documents to create customisable graphical fader components existed; this mechanism was limited to the aesthetics of graphical faders and did not take parameter grouping and monitoring into account.

As part of this investigation, the Unos Creator implementation was enhanced to include bidirectional remote control, parameter grouping, and monitoring capabilities. These enhancements were implemented using the parameter grouping and monitoring relationships resulting from this investigation. The main contributions from this investigation include:

- The integration of parameter grouping and monitoring mechanisms into Unos Creator. This integration was implemented by binding graphical controls within Unos Creator to XFN parameters in Unos Creator's XFN stack. Graphical controls that are bound to XFN parameters in this manner are referred to as deskitems. Bi-directional remote control of devices is achieved by creating permanent absolute peer-to-peer relationships between deskitem XFN parameters and the corresponding remote media device parameters. Relationships that are created between associated XFN parameters are mirrored by the corresponding graphical controls.
- The implementation of graphical interfaces to manage relationships between networked XFN parameters via Unos Creator. These interfaces facilitate the creation relationships between deskitems. When relationships are created between deskitems, relationships are subsequently created between the media device parameters that are associated with the deskitems.

Apart from these grouping relationships, four new deskitem types were created, namely pot, button, display, and meter deskitems. Notably, the display deskitem demonstrates the ability to use the values of multiple grouped remote parameters as inputs to a *function* resulting in the display of graphical curves that resemble equalisers on mixing consoles. The meter deskitem was created to demonstrate parameter monitoring. An existing graphical fader implementation was enhanced to enable parameter grouping via graphical faders.

This chapter begins by giving an overview of Unos Creator, followed by a description of the deskitem concept. The implementation of parameter relationships is essential for the deskitem mechanism to work. The chapter concludes by illustrating procedures for creating parameter relationships and for establishing parameter monitoring.

7.1 Overview of Unos Creator

Unos Creator is a graphical control application that reflects the capabilities of the XFN protocol. Figure 7.1 shows an overview of the Unos Creator application architecture in the context of the XFN protocol. As with any XFN device, Unos Creator comprises an application layer and an XFN stack layer that interact via the XFN API (Application Programming Interface). The

application layer of Unos Creator consists of a GUI that fulfils various application-specific functionality. The XFN stack layer is standard across all XFN devices and comprises a number of XFN nodes. These XFN nodes are composed of XFN parameters and, optionally, USG buffers if any of the XFN parameters are retrieved in bulk or monitored. Unos Creator is a classic example of an application with dual roles, namely as controller and controllable device. As a controller, Unos Creator associates each networked multimedia device with its own XFN node that has appropriate mappings between the Unos Creator XFN parameters and the device parameters. As a device, Unos Creator exposes its own application-specific functionality via XFN nodes.

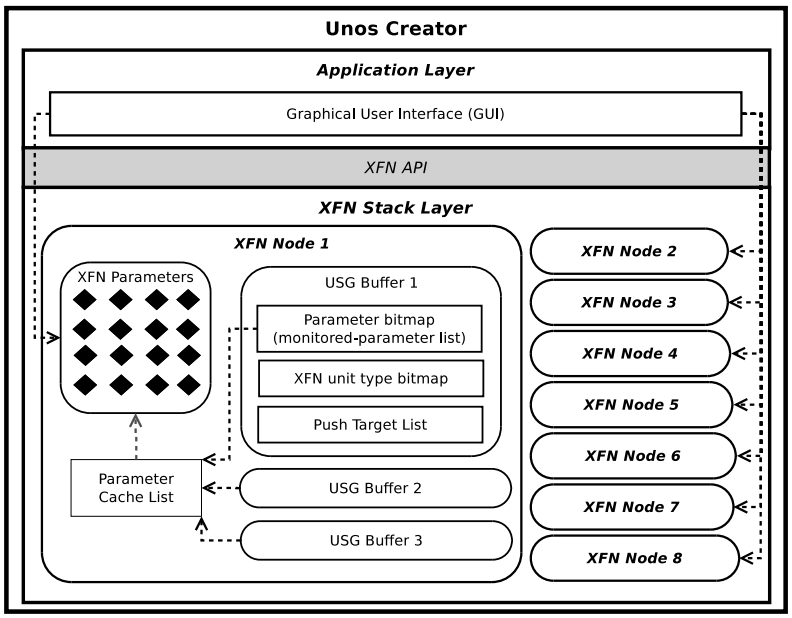


Figure 7.1: Unos Creator Application Overview

Key features of Unos Creator include:

- Multimedia device discovery.
- Static network configuration.
- Stream connection management.

Unos Creator implements a mechanism for the discovery of multimedia devices distributed across a network using the XFN protocol. In some cases, for example networks without a dynamic host configuration protocol (DHCP) server, Unos Creator can be used to assign fixed network addresses (commonly IP addresses) to multimedia devices prior to device discovery.

Once devices have been discovered, Unos Creator is capable of creating multimedia stream connections between devices, such as initiating the streaming of video or audio from one device to another. In other words, it enables connection management.

Section 7.2 begins by describing the implementation of deskitems and how they can be used to control remote parameters. Section 7.3 then describes how grouping relationships are managed using deskitems. Section 7.4 describes how parameter monitoring is achieved using deskitems.

7.2 Deskitems

Figure 7.2 shows an example of a control surface for a media player that is composed of various graphical deskitem controls, including pots, faders, meters, buttons, equaliser displays, and labels. With the exception of faders, that were enhanced from an existing implementation, these controls were implemented as part of this study. Each of these controls is built from an XML element that describes the behaviour and appearance of the control. Most importantly, the XML element contains a hierarchical address of the remote parameter that is controlled. A mechanism was implemented in order to enable storage of compressed packages deskitem packages within multimedia devices. These packages are retrieved and rendered by controllers. Each of the controls rendered is associated with an XFN parameter on the controller. To complete the remote control process, the controller creates a permanent absolute peer-to-peer relationship between each deskitem's XFN parameter on the controller and the corresponding XFN parameter on the remote multimedia device. This provides the controller with read/write access of the remote XFN parameters. Section 5.5.3 on page 114 describes the need for permanent absolute peer-to-peer relationships between a controller and a device, where the relationship between the controller's XFN parameters and the remote device's XFN parameters is fixed and can only be relinquished by the controller itself.

It is worth noting that meter deskitems are an exception to the permanent absolute peer-to-peer relationship rule. Meter deskitems are exclusively read-only controls that are used to monitor the state of remote parameters, such as audio peak level parameters. In some cases, the parameters being monitored change values constantly, for example every 25 ms in our audio level monitoring implementations. Therefore, an efficient bulk parameter monitoring mechanism is most suited to this task. Section 6.2 on page 135 gives a detailed description of the push mechanism that is

Figure 7.2: Media Player Control Surface

used for parameter monitoring. For optimal performance, this push mechanism was chosen for use in conjunction with meter deskitems in Unos Creator.

Figure 7.3 shows a summary of the procedure involved in order to remotely control multimedia devices, where:

1. The deskitem package is downloaded from a multimedia device and rendered in the Unos Creator GUI.
2. Unos Creator creates associations between the graphical controls and XFN parameters within its XFN stack.
3. Permanent absolute peer-to-peer relationships are created between Unos Creator's XFN parameters and the XFN parameters within the devices.

This three step process described above is one of the contributions of this investigation. Section 7.2.1 describes the main deskitem properties and the XML that is used to describe these proper-

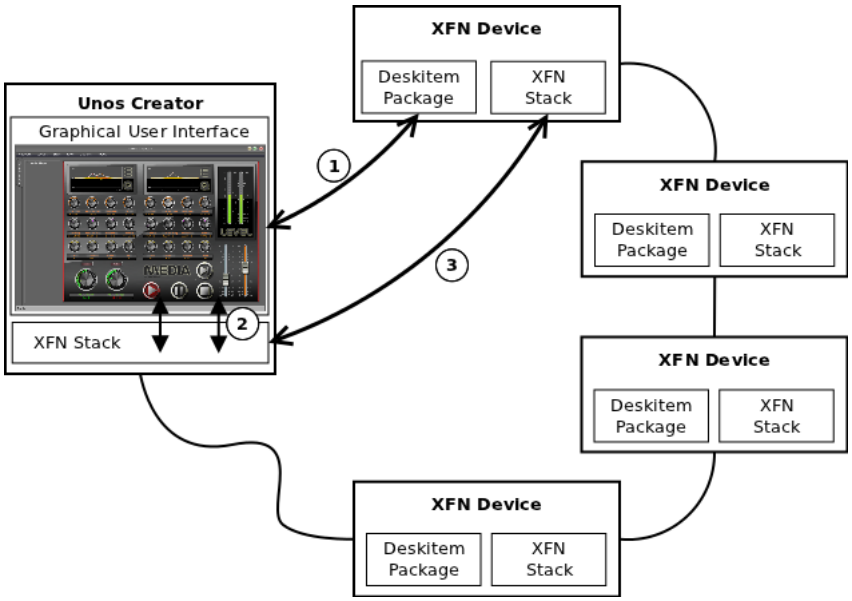


Figure 7.3: Procedure for Device-Specific Deskitem Extraction

ties. Section 7.2.2 will describe a graphical editor that is used to create deskitem packages for devices.

7.2.1 Deskitem Properties

In the course of this investigation, a number of deskitem types were created, including background, fader, pot, meter, multi-I/O, and display deskitem types. Each deskitem contains a number of properties that are described by an XML element that is composed of a number of XML attributes and, in some cases, nested child XML elements. All deskitem types share some common properties in addition to their deskitem-type-specific properties. We shall begin by giving descriptions of the properties that are shared by all deskitems, then describe the type-specific properties for the various deskitem types.

7.2.1.1 Common Deskitem Properties

Table 7.1 lists the properties that are applicable to all deskitem types. These properties are grouped into two categories, namely XFN parameter information, and deskitem naming and positioning. The XFN parameter information properties identify the XFN parameter on the controlled device and specify the nature of the control relationship that will be created between the

deskitem's XFN parameter on the controller and the XFN parameter on the controlled device. A combination of IP address, XFN node ID, and hierarchical parameter address is used to identify the XFN parameter on the controlled device. The nature of the control relationship specifies the use of either a grouping or push relationship. The naming and positioning properties describe the textual name and position of the deskitem on the screen.

Table 7.1: Common Deskitem Properties

XFN PARAMETER INFORMATION		
Property	**XML attribute**	**Description**
IP address	ipAddress	The network address of the XFN device in which the parameter that is controlled by this deskitem resides. Given the dynamic nature of the network address in some networks, this property is set by the controller when the deskitem is rendered.
XFN node ID	xfnNodeId	The ID of the XFN node in which the controlled parameter resides.
Level 1 parameter address	xfnAddressLevel1	The full hierarchical XFN parameter address of the parameter that is controlled by the deskitem.
Level 2 parameter address	xfnAddressLevel2	
Level 3 parameter address	xfnAddressLevel3	
Level 4 parameter address	xfnAddressLevel4	
Level 5 parameter address	xfnAddressLevel5	
Level 6 parameter address	xfnAddressLevel6	
Level 7 parameter address	xfnAddressLevel7	
Join type	joinType	The nature of the relationship to be created between the deskitem XFN parameter and the controlled parameter. A value of '0' means that a permanent absolute peer-to-peer join should be created, while a value of '1' means that the deskitem will be updated via the push mechanism (best suited for meter deskitems).

continued on next page

Common deskitem properties (continued)

Global unit table file	globalUnitTableFile	The path to a file that contains mappings from global units to deskitem-specific units.
Global unit bit range	globalUnitBitRange	The granularity of the control relative to the global unit range. Recall from Section 5.4.3.2 on page 99 that a controller can control up to 31 of bits of global unit value range.
DESKITEM NAMING & POSITIONING		
Property	**XML attribute**	**Description**
Name	name	The name of the deskitem.
Screen coordinates	pos	The x and y screen coordinates, together with width and height of the graphical component, in pixels. Used when rendering the deskitem. The format of the XML attribute value is "<*x*> <*y*> <*width*> <*height*>"

7.2.1.2 Background Deskitem Properties

The background deskitem, as the name suggests, is a form of canvas on top of which other deskitems are placed. This deskitem can have a solid colour or an image as the background. In the case of an image background, the image supplied can contain a series of snapshots of all possible background images, where the selection of the image to be displayed can be linked to the value of an associated XFN parameter. Table 7.2 lists the properties that are specific to the background deskitem, in addition to the common properties described in Section 7.2.1.1. Figure 7.4 shows the structure of the XML element that describes a background deskitem.

BACKGROUND DESKITEM PROPERTIES		
Property	**XML attribute**	**Description**
Background colour	backgroundColour	The solid colour that will be used to fill the background deskitem when no image is supplied.
Background image stripe	backImageStripe	The name of the image that contains a series of snapshots for all possible background images that can be displayed.
Number of background images	numBackStripeImages	The number of snapshots contained in the background image stripe.

Table 7.2: Background Deskitem Properties

```
<BACKGROUND_DESKITEM numBackStripeImages="1" globalUnitTableFile=""
    pos="0 0 870 625" joinType="0" xfnAddressLevel7="" xfnAddressLevel6=""
    xfnAddressLevel5="" xfnAddressLevel4="" xfnAddressLevel3=""
    xfnAddressLevel2="" xfnAddressLevel1="" xfnNodeId="1"
    ipAddress="169.254.141.228" name="ApplicationWindow"
    backImageStripe="background.png" backgroundColour="ff626262"/>
```

Figure 7.4: Background Deskitem XML Element

7.2.1.3 Fader Deskitem Properties

The fader deskitem is a slider input control, commonly found on mixing consoles. Table 7.3 on the following page lists properties that are specific to fader deskitems. These properties pertain to the look and feel of the deskitem. The properties are also annotated in Figure 7.5 on page 166. With the exception of the backgroundImage, trackImage, and thumbImage, the properties listed in Table 7.3 on the following page were added in this investigation in order to make fader deskitems more customisable.

Figure 7.6 shows an example of a complete fader deskitem XML element.

```
<FADER_DESKITEM globalUnitTableFile="" pos="699 359 44 264" joinType="0"
    xfnAddressLevel7="1" xfnAddressLevel6="201" xfnAddressLevel5="1"
    xfnAddressLevel4="11" xfnAddressLevel3="1" xfnAddressLevel2="d1"
    xfnAddressLevel1="1" xfnNodeId="1" ipAddress="169.254.141.228"
    name="Fader left" thumbNeedleRangeLength="222" thumbNeedRangeOffset="23"
    thumbNeedleOffset="32" thumbX="8" trackY="11" trackX="16"
    thumbimage="faderButton.png" trackimage="faderTrack.png"
    backgroundImage="faderBack.png" globalUnitBitRange="8"/>
```

Figure 7.6: Fader Deskitem XML Element

FADER DESKITEM PROPERTIES		
Property	**XML attribute**	**Description**
Background image	backgroundImage	The file name of the image that is drawn as the background of the deskitem. This image can, for example, contain the gradation of the fader.
Track image	trackImage	The file name of the image that is drawn in the region that the fader thumb slides over. This is superimposed onto the background image.
Thumb image	thumbImage	The file name of the image that is drawn as the slider of the fader. This image contains a needle that represents a marker to identify the current value of the fader.
Track x coordinate	trackX	The x coordinate, in pixels, from the leftmost side of the fader deskitem graphic, where the fader track image is drawn.
Track y coordinate	trackY	The y coordinate, in pixels, from the topmost side of the fader deskitem graphic, where the fader track image is drawn.
Thumb x coordinate	thumbX	The x coordinate, in pixels, from the leftmost side of the fader deskitem graphic, where the thumb image is drawn.
Thumb needle offset	thumbNeedleOffset	The y coordinate, in pixels, from the top of the thumb image, where the thumb needle is displayed. This is used to align the current value of the fader with the gradation on the fader background.
Thumb needle range offset	thumbNeedRangeOffset	The y coordinate, in pixels, from the topmost side of the fader deskitem graphic, representing the highest point the thumb needle can be dragged to.
Thumb needle range length	thumbNeedleRangeLength	The length, in pixels, of the path along which the thumb needle can be dragged.

Table 7.3: Fader Deskitem Properties

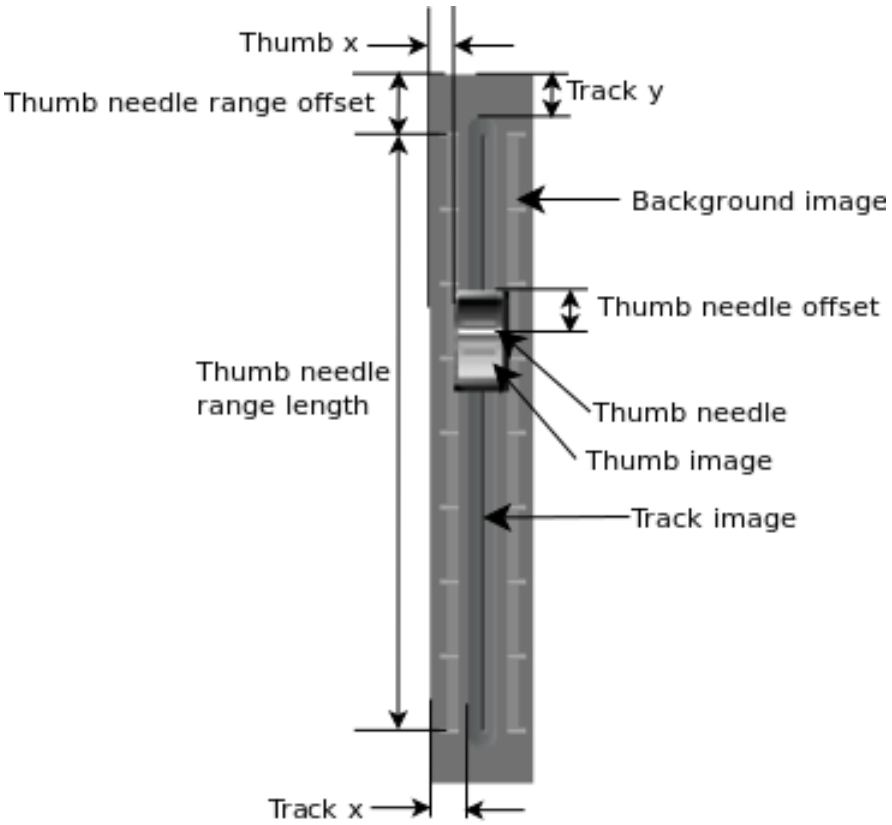


Figure 7.5: Annotated Fader Deskitem

7.2.1.4 Pot Deskitem Properties

The pot deskitem control is a rotary value input control with a needle that points to the current selected value. Table 7.4 lists properties that are specific to pot deskitems. These properties pertain to the look and feel of the deskitem graphic control. Some of these properties are annotated in Figure 7.7 on the next page.

POT DESKITEM PROPERTIES		
Property	**XML attribute**	**Description**
Background image	backImage	The file name of the image that is drawn as the background of the deskitem. This image can, for example, contain the gradation of the pot.
Needle origin x coordinate	rectNeedleOriginX	The x coordinate, in pixels, of the axis of rotation of the pot.
Needle origin y coordinate	rectNeedleOriginY	The y coordinate, in pixels, of the axis of rotation of the pot.
Needle origin offset	rectNeedleOriginOffset	The length, in pixels, from the pot's axis of rotation to the point where the needle will be drawn from.
Needle length	rectNeedleLength	The length of the pot's needle.
Needle thickness	rectNeedleDotThickness	The thickness of the pot needle line.
Needle range offset	rectNeedleRangeOffset	The angle, in degrees, from the rotary origin (show in Figure 7.7 on the following page) to the minimum needle value point.
Needle range	rectNeedleRange	The angle, in degrees, over which the needle can be drawn. This is the angle from the minimum needle value to point to the maximum needle value point as shown in Figure 7.7 on the next page.
Needle colour	needleDotColour	The colour in which the needle is drawn over the background image.

Table 7.4: Pot Deskitem Properties

Figure 7.8 shows a complete XML element for a pot deskitem.

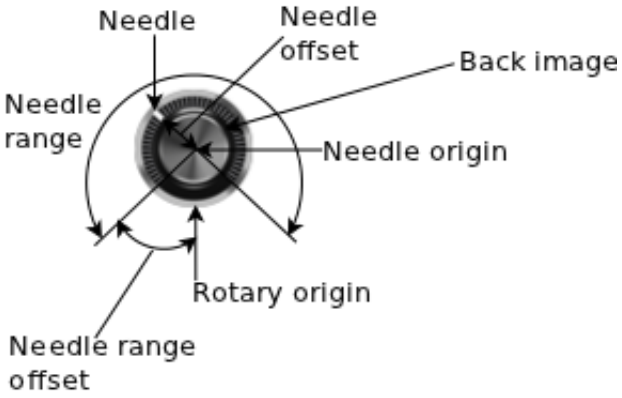


Figure 7.7: Annotated Pot Deskitem

```
<POT_DESKITEM pos="96 64 55 55" joinType="0" xfnAddressLevel7="1" xfnAddressLevel6="201"
    xfnAddressLevel5="1" xfnAddressLevel4="31" xfnAddressLevel3="1" xfnAddressLevel2="d1"
    xfnAddressLevel1="1" xfnNodeId="" name="EQ left low Gain" globalUnitBitRange="8"
    mode="1" rectNeedleDotThickness="2" rectNeedleRange="270" rectNeedleRangeOffset="45"
    rectNeedleLength="5" rectNeedleOriginOffset="20" rectNeedleOriginY="27.5"
    rectNeedleOriginX="27.5" backImage="potBack.png" needleDotColour="ffffffff"/>
```

Figure 7.8: Pot Deskitem XML Element

7.2.1.5 Meter Deskitem Properties

The meter deskitem is a display-only deskitem type that is typically used to monitor audio levels, temperature, voltage, *et cetera*. Section 6.2.2.2 describes the initialisation of meter deskitems as well as the deskitem structure. Table 7.5 lists the properties that are specific to meter deskitems. These properties pertain to the look and feel of the meter, as well as the meter ballistics. Figure 7.9 on page 171 shows an annotated graphical meter control.

Table 7.5: Meter Deskitem Properties

METER DESKITEM PROPERTIES		
Property	**XML attribute**	**Description**
Background image	backImage	The file name of the image that is drawn as the background of the deskitem. This image can, for example, contain the gradation of the meter deskitem graphic control.
Number of steps	numberOfSteps	A count of all values that can be displayed by the meter.

continued on next page

METER DESKITEM PROPERTIES (CONTINUED)		
Property	**XML attribute**	**Description**
Bar image	barStripeImage	The file name of the image that is used to draw the bar section of the meter (see Figure 7.9 on page 171). This image contains snapshots of all possible values.
Peak image	peakStripeImage	The file name of the image that is used to draw the peak section of the meter (see Figure 7.9 on page 171). This image contains snapshots of all possible peak values. The image is superimposed above the bar image.
Bar x coordinate	barX	The x coordinate, in pixels, from the leftmost side of the graphical meter, where the bar will be drawn.
Bar y coordinate	barY	The y coordinate, in pixels, from the topmost side of the graphical meter, where the bar will be drawn.
Bar width	barWidth	The width of the meter bar in pixels.
Bar height	barHeight	The height of the meter bar in pixels.
Ballistics enabled	ballisticsEnabled	Indicates whether or not the meter is using ballistics. A value of 1 implies that ballistics are enabled, while 0 implies that ballistics are disabled. In professional audio, meter ballistic standards have been defined for peak programme meters (PPMs).
Peak hold time	peakHoldMillis	The millisecond time interval within which the peak value is displayed before falling back to the minimum meter value. This property is used with meter ballistics enabled.
Peak fall back time	peakFallbackMillis	The millisecond time interval between each fall of the peak value. This property is used with meter ballistics enabled.

continued on next page

METER DESKITEM PROPERTIES (CONTINUED)		
Property	**XML attribute**	**Description**
Peak fall back steps	peakFallbackValue	The number of steps that the peak values falls after every peak fall back time interval. This property is used with meter ballistics enabled.
Bar hold time	barHoldMillis	The millisecond time interval within which the bar value is displayed before falling back to the minimum meter value. This property is used with meter ballistics enabled.
Bar fall back time	barFallbackMillis	The millisecond time interval between each fall of the bar value. This property is used with meter ballistics enabled.
Bar fall back steps	barFallbackValue	The number of steps that the bar values falls after every bar fall back time interval. This property is used with meter ballistics enabled.

Figure 7.10 on the next page shows an example of a complete meter deskitem XML element.

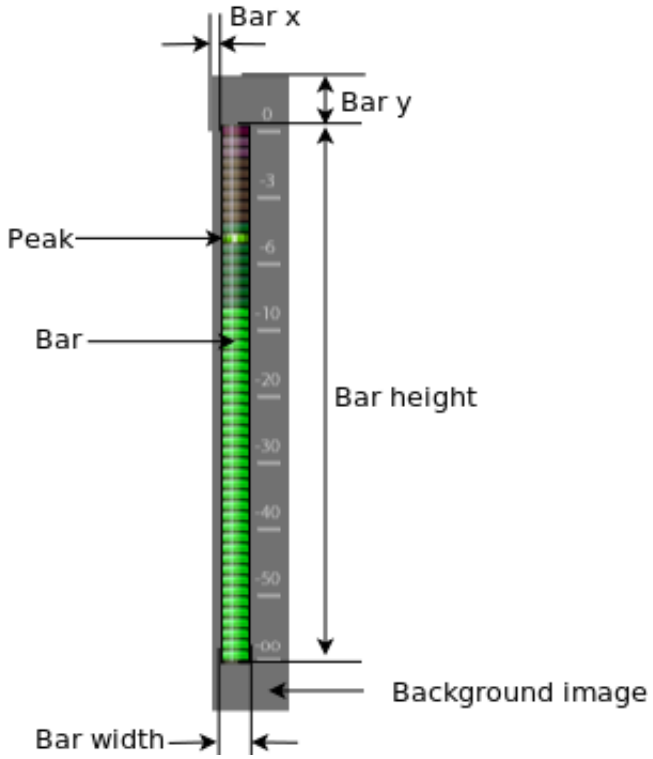


Figure 7.9: Annotated Meter Deskitem

```
<METER_DESKITEM globalUnitTableFile="units.txt" pos="729 47 18 234" joinType="1"
    xfnAddressLevel7="1" xfnAddressLevel6="d05c" xfnAddressLevel5="1" xfnAddressLevel4="a1"
    xfnAddressLevel3="1" xfnAddressLevel2="d1" xfnAddressLevel1="11" xfnNodeId="1"
    ipAddress="169.254.141.228" name="Meter left" backImage="background.png"
    ballisticsEnabled="1" barFallbackValue="20" barFallbackMillis="1000" barHoldMillis="0"
    peakFallbackValue="20" peakFallbackMillis="1000" peakHoldMillis="1000" numberOfSteps="50"
    barHeight="234" barWidth="18" barY="0" barX="0" peakStripeImage="peakImage.png"
    barStripeImage="barImage.png"/>
```

Figure 7.10: Meter Deskitem XML Element

7.2.1.6 Multi-I/O Deskitem Properties

The multi-I/O deskitem, as the name suggests, is a multiple function input-output deskitem that is capable of functioning as a static label, LED, on/off button, increment button, or decrement button. Table 7.6 lists the main properties of the multi-I/O deskitem. Figure 7.11 on page 174 shows an annotated illustration of an on/off mute button graphical component.

Table 7.6: Multi-I/O Deskitem Properties

MULTI-I/O DESKITEM PROPERTIES		
Property	**XML attribute**	**Description**
Style	style	This property specifies the mode of operation of the graphical component. Modes include, but are not limited to, on/off button, increment/decrement button, and text label.
"On" global unit value	onValueGlobalUnitPercent	For on/off buttons, this is the global unit value that the deskitem sends when the button transitions to the "on" state.
"Off" global unit value	offValueGlobalUnitPercent	For on/off buttons, this is the global unit value that the deskitem sends when the button transitions to the "off" state.
Threshold global unit value	thresholdValueGlobalUnitPercent	For on/off buttons, this is the global unit value at which a changeover between the "on' and the "off" states happens. All values at the threshold or below are considered as "off", otherwise considered as "on".
Background image	backimage	The file name of the image that is drawn as the background of the deskitem.
Active part on image	activePartOnImage	The file name of the image that is drawn as the active part when the deskitem is in the "on" state.
Active part normal image	activePartNormalImage	The file name of the image that is drawn as the active part when the deskitem is **not** in the "on" state.

continued on next page

MULTI-I/O DESKITEM PROPERTIES (CONTINUED)		
Property	**XML attribute**	**Description**
Active part x	activePartX	The x coordinate, in pixels, from the leftmost side of the graphic component to the start of the active part image drawing area.
Active part y	activePartY	The y coordinate, in pixels, from the topmost side of the graphic component to the start of the active part image drawing area.
Active part width	activePartWidth	The width, in pixels, of the active part image drawing area.
Active part height	activePartHeight	The height, in pixels, of the active part image drawing area.
Active part mouse handler	activePartMouseHandler	A flag that enables or disables the mouse event listener for the active part. For example, an on/off button with the mouse handler disabled does not toggle the button state upon a mouse click.
Number of text regions	numTextRegions	The count of all text regions on the graphical component
Text property	text (child element)	For each of the text areas on the deskitem, various properties can be set, including justification, font, font size, and drawing area.
Inc/Dec button properties	incDecProperties (child element)	For increment/decrement buttons, properties such as the delay interval when the button is continuously pressed and the delta for the increment/decrement are configurable.

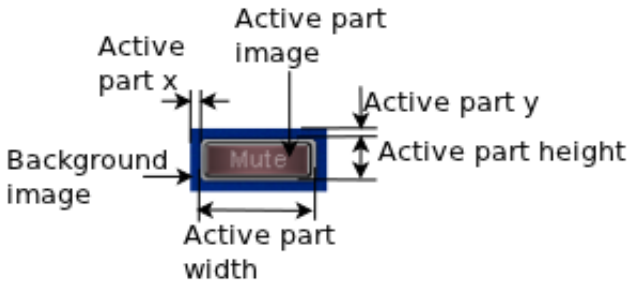


Figure 7.11: Annotated Multi-I/O Deskitem

Figure 7.12 shows an example of a complete XML element for the multi-I/O deskitem.

```
- <MULTI_IO_DESKITEM pos="434 322 63 29" joinType="0" xfnAddressLevel7=""
  xfnAddressLevel6="" xfnAddressLevel5="" xfnAddressLevel4="" xfnAddressLevel3=""
  xfnAddressLevel2="" xfnAddressLevel1="" xfnNodeId="1" ipAddress="169.254.141.228"
  name="multi i/o desk item" numTextRegions="3" activePartMouseHandler="1"
  activePartHeight="23" activePartWidth="57" activePartY="3" activePartX="3"
  thresholdValueGlobalUnitPercent="50" offValueGlobalUnitPercent="0"
  onValueGlobalUnitPercent="100" style="OnOff" globalUnitTableFileFile=""
  activePartOnImage="Button Mute on.png" activePartNormalImage="Button Mute off.png"
  backimage="backImage.jpg" xfnUnitBitRange="8">
      <text colour="ff000000" justification="33" italic="0" bold="0" fontsize="15"
          fontname="Default font" labelText="" height="0" width="0" y="0" x="0" id="0"/>
      <text colour="ff000000" justification="33" italic="0" bold="0" fontsize="8.9"
          fontname="Default font" labelText="" height="0" width="0" y="0" x="0" id="1"/>
      <text colour="ff000000" justification="33" italic="0" bold="0" fontsize="15"
          fontname="Default font" labelText="" height="0" width="0" y="0" x="0" id="2"/>
      <incDecProperties minRepeatDelay="-1" repeatDelay="0" initialRepeatDelay="-1"
          unitDelta="1"/>
  </MULTI_IO_DESKITEM>
```

Figure 7.12: Multi-I/O Deskitem XML Element

7.2.1.7 Display Deskitem Properties

The display deskitem is a special type of composite deskitem that uses values from one or more remote XFN parameters as input arguments to a function that outputs a graphical representation of the input. Figure 7.13 on the next page shows the operation of this deskitem type, where:

1. Permanent absolute peer-to-peer relationships exist between one or more remote XFN parameters and their dedicated deskitem XFN parameters.

2. The global unit values of the remote parameters are converted to their application-specific unit value equivalents via mapping tables.
3. The application-specific unit values are fed into a function (algorithm) as input argument values.
4. The function generates some output in the form of, for example, a graph that is displayed appropriately. Two functions were implemented for this investigation, namely a function to plot the graph for a noise gate and a function to plot the graphs for an equaliser (EQ).

Remote XFN parameters attached to a display deskitem are also controllable, simultaneously, via other deskitem types, such as fader, pot, and button deskitems. This makes it possible to create fully customisable control surfaces with graph-based value manipulation capabilities typically found on digital mixing consoles.

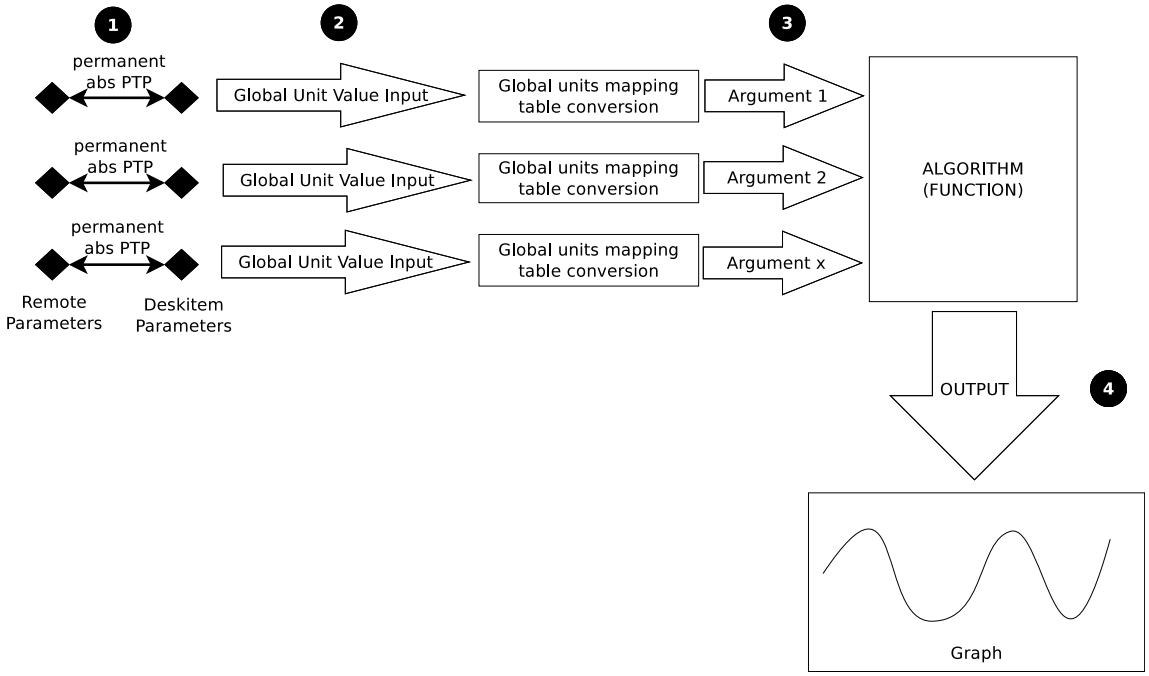


Figure 7.13: Display Deskitem Overview

Figure 7.14 on the following page shows the XML element for an EQ display deskitem. Here, there are four algorithm child elements that result in four equaliser graphs that are superimposed over each other. Each algorithm contains a name attribute, type attribute, and one or more argument child elements. The name attribute is the algorithm's textual descriptor, while the type attribute specifies a numerical ID that identifies the algorithm in use. Each argument child element is composed of hierarchical XFN parameter addresses of the remote XFN parameters that

input into the algorithm, together with their corresponding mapping tables. Table 7.7 lists the properties of each algorithm argument.

```
- <DISPLAY_DESKITEM pos="372 20 232 112" xfnNodeId="1" ipAddress="169.254.141.228" name="EQ
  Display">
    - <algorithm name="" type="3">
        <argument globalUnitTableFile="../xfn_unit_tables/eqQualityglobalUnitTableFile.txt"
            xfnAddressLevel7="1" xfnAddressLevel6="702" xfnAddressLevel5="1" xfnAddressLevel4="31"
            xfnAddressLevel3="2" xfnAddressLevel2="d1" xfnAddressLevel1="1" index="0"/>
        <argument globalUnitTableFile="../xfn_unit_tables/eqFrequencyglobalUnitTableFile.txt"
            xfnAddressLevel7="1" xfnAddressLevel6="701" xfnAddressLevel5="1" xfnAddressLevel4="31"
            xfnAddressLevel3="2" xfnAddressLevel2="d1" xfnAddressLevel1="1" index="1"/>
        <argument globalUnitTableFile="../xfn_unit_tables/eqGainglobalUnitTableFile.txt"
            xfnAddressLevel7="1" xfnAddressLevel6="201" xfnAddressLevel5="1" xfnAddressLevel4="31"
            xfnAddressLevel3="2" xfnAddressLevel2="d1" xfnAddressLevel1="1" index="2"/>
      </algorithm>
    + <algorithm name="" type="3">
    + <algorithm name="" type="3">
    + <algorithm name="" type="3">
  </DISPLAY_DESKITEM>
```

Figure 7.14: Display Deskitem XML Element

Table 7.7: Display Deskitem Algorithm Argument Properties

DISPLAY DESKITEM ALGORITHM ARGUMENT PROPERTIES		
Property	**XML attribute**	**Description**
Index	index	The zero-based offset of the argument input position into the algorithm. These offsets have various meanings depending on the type of algorithm being referenced. For the EQ algorithm, the input at index zero is for the quality parameter, while the input at index one is for the frequency parameter, and the input at index two is for the gain parameter.
Global unit table file	globalUnitTableFile	The path to a file that contains mappings from global units to application-specific units.
Level 1 parameter address	xfnAddressLevel1	The full hierarchical XFN parameter address of the remote parameter that inputs into the algorithm.
Level 2 parameter address	xfnAddressLevel2	

continued on next page

DISPLAY DESKITEM ALGORITHM ARGUMENT PROPERTIES (CONTINUED)		
Property	**XML attribute**	**Description**
Level 3 parameter address	xfnAddressLevel3	The full hierarchical XFN parameter address of the remote parameter that inputs into the algorithm.
Level 4 parameter address	xfnAddressLevel4	
Level 5 parameter address	xfnAddressLevel5	
Level 6 parameter address	xfnAddressLevel6	
Level 7 parameter address	xfnAddressLevel7	

7.2.2 A GUI Editor for Deskitems

Section 7.2.1 describes the properties of the various deskitem types that were created in Unos Creator. The XML elements that describe properties of each deskitem type have also been shown. In order to simplify the deskitem creation process, a graphical editor tool was created for Unos Creator. This tool provides an intuitive interface for building control surfaces. Using this tool, it is possible to add, modify, and delete deskitems on a control surface. Recall from the beginning of Section 7.2 on page 159 that multimedia devices store compressed packages of their deskitems. These compressed packages are downloaded and rendered by controllers. The GUI editor tool also provides functionality to create compressed packages and save them in a format that is loadable by controllers, such as zip files.

Figure 7.15 shows a screenshot of the GUI editor tool for Unos Creator. As shown in the screenshot, there is a canvas on the left-hand-side over which a number of deskitems are placed. Properties for each selected deskitem are editable via the property value inputs on the right-hand-side of the screen. For brevity, we shall only describe the procedures for creating a new deskitem, associating the new deskitem with a remote parameter, and packaging the deskitem into a zip file.

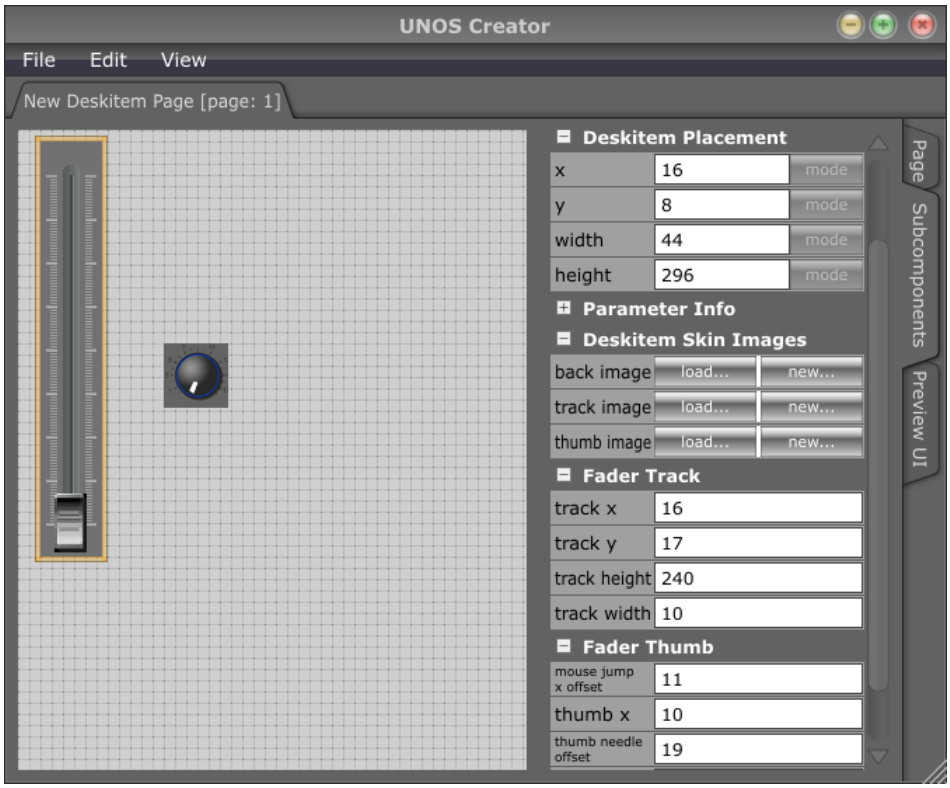


Figure 7.15: Unos Creator GUI Editor Tool

7.2.2.1 Procedure for Creating New Deskitems

In order to create new deskitems using the GUI editor tool, the following steps are required:

1. Addition of new deskitem pages
   A new deskitem page is created by navigating to `File | New Deskitem Page` as shown in Figure 7.16 on the next page. This creates a new canvas on which new deskitems can be placed.

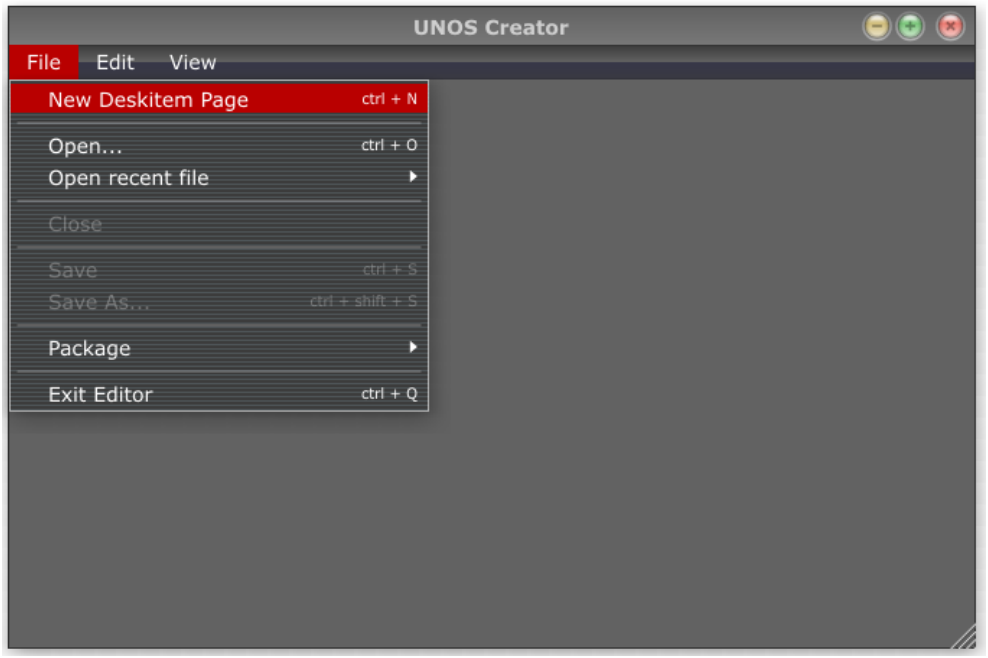


Figure 7.16: Adding a New Deskitem Page

2. Addition of new deskitem controls
   New deskitems are added to the new deskitem page by navigating to `Edit | Add new deskitem | <deskitem type>`, where `<deskitem type>` corresponds to the type of the new deskitem to be added. Figure 7.17 on the following page shows an example of adding a new fader deskitem, resulting in the addition of a new "skinnable[1]" fader control to the deskitem page canvas. Figure 7.18 on the next page shows the unskinned fader deskitem control.

[1] Skinning is the changing of the appearance of a deskitem using one or more pictures.

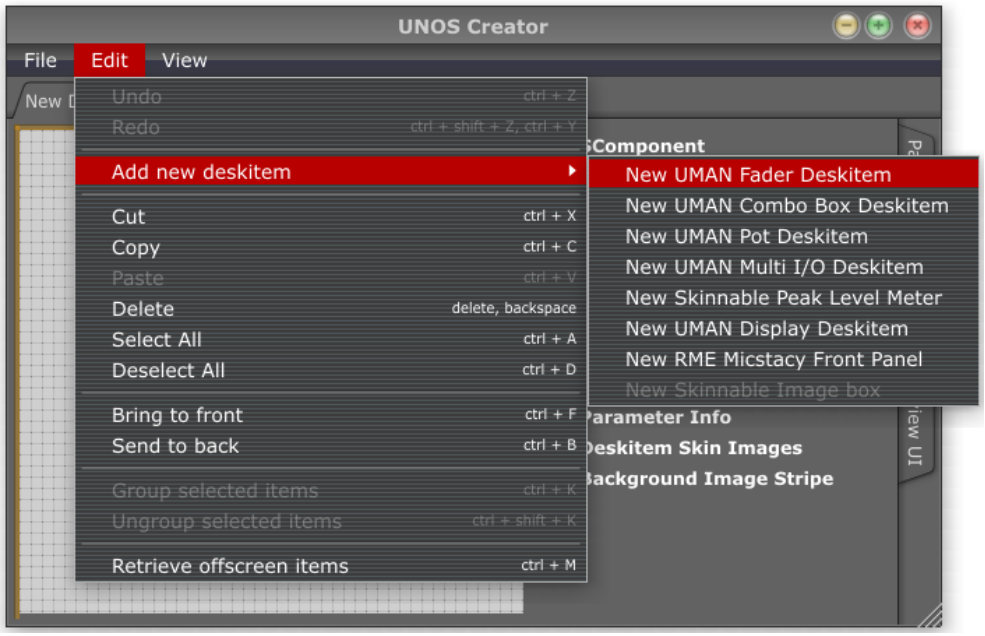


Figure 7.17: Adding a Fader Deskitem

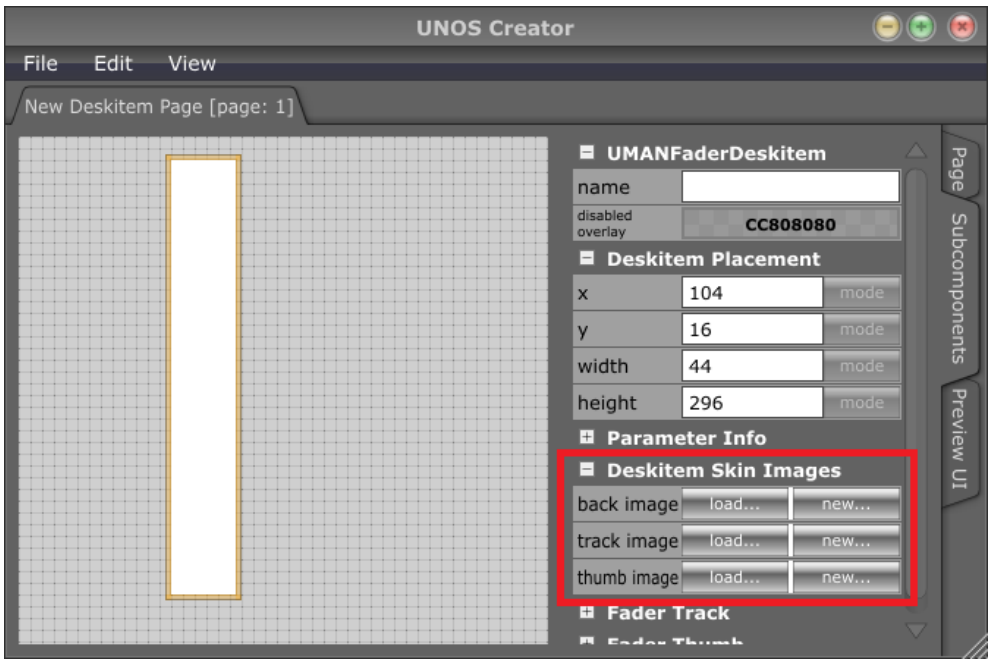


Figure 7.18: Uncustomised Fader Deskitem

3. Skinning new deskitem controls
   The new deskitem controls in part 2 are customised by specifying one or more images to be used for rendering various sections of the deskitem control. For fader deskitems, for exam-

ple, the background, track, and thumb of the fader are skinnable as shown by the "Deskitem Skin Images" section in Figure 7.18 on the preceding page. The background, track, and thumb regions of a fader have been displayed graphically in Figure 7.5 on page 166. In order to skin sections of a deskitem control:

(a) The "load..." button (see Figure 7.18) of the section to be skinned is pressed.

(b) An "Image Properties" window is displayed as shown in Figure 7.19, where various properties of the image can be edited.

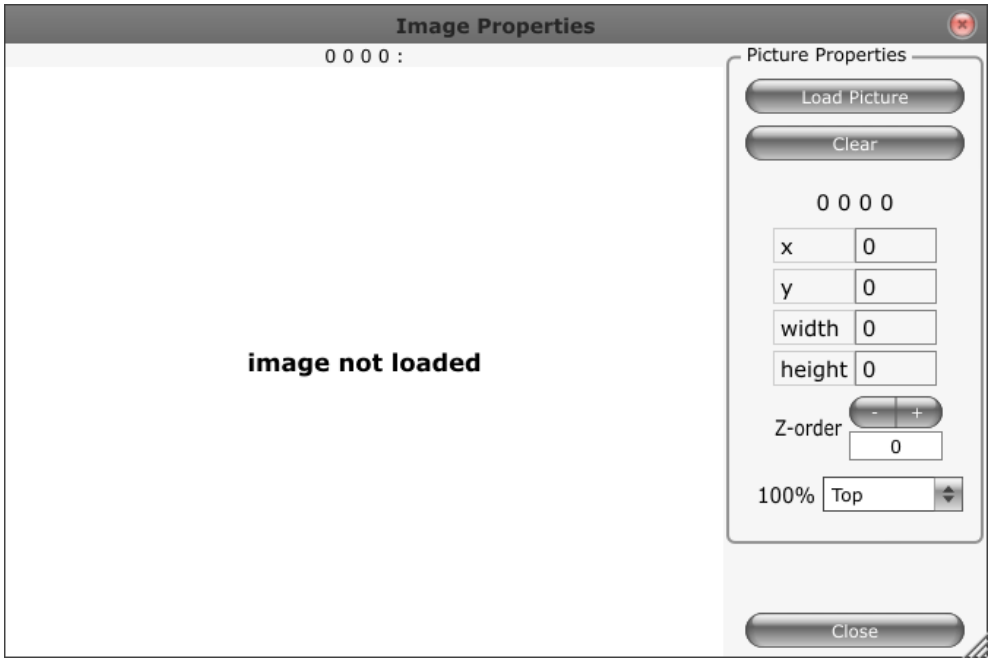


Figure 7.19: Image Property Editor

(c) The "Load Picture" button is pressed, resulting in the display of a file browser window as shown in Figure 7.20 on the next page, where the file containing the image for the section being skinned is chosen. For completeness, Figure 7.21 on the following page shows a fully skinned fader deskitem control.

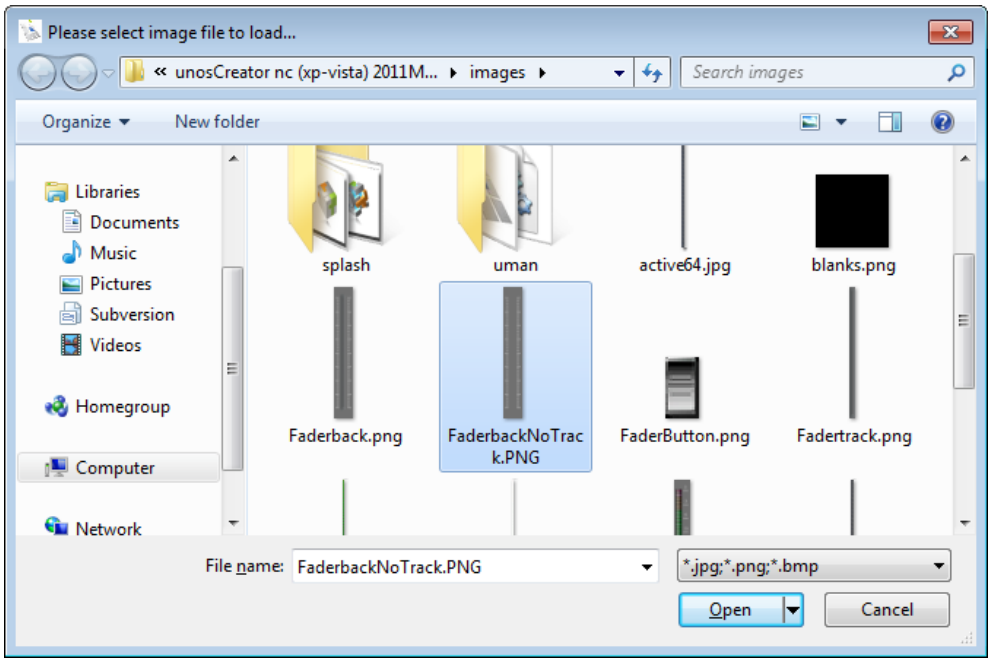


Figure 7.20: Image File Browser

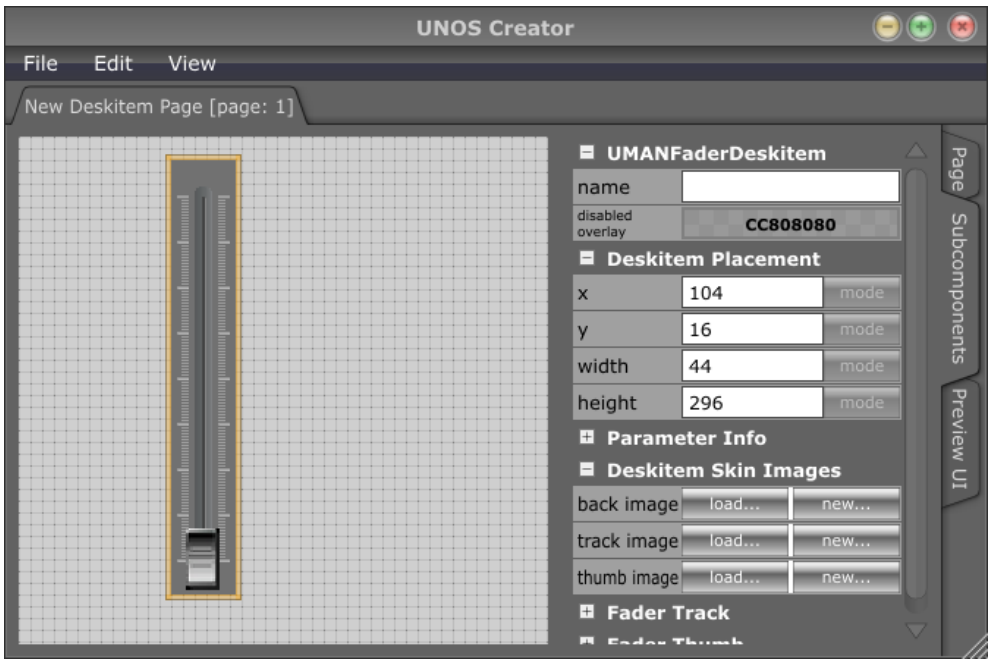


Figure 7.21: Fully Skinned Fader Deskitem

4. Associating deskitem controls with remote XFN parameters

   Recall from step 3 of the deskitem remote control procedure illustrated in Figure 7.3 on page 161 that a permanent absolute peer-to-peer relationship is created between the XFN

parameter that is bound to a deskitem control in Unos Creator and the corresponding XFN parameter on a remote multimedia device. To specify the remote XFN parameter for a deskitem:

(a) The “choose parameter...” button shown in Figure 7.22 is pressed.

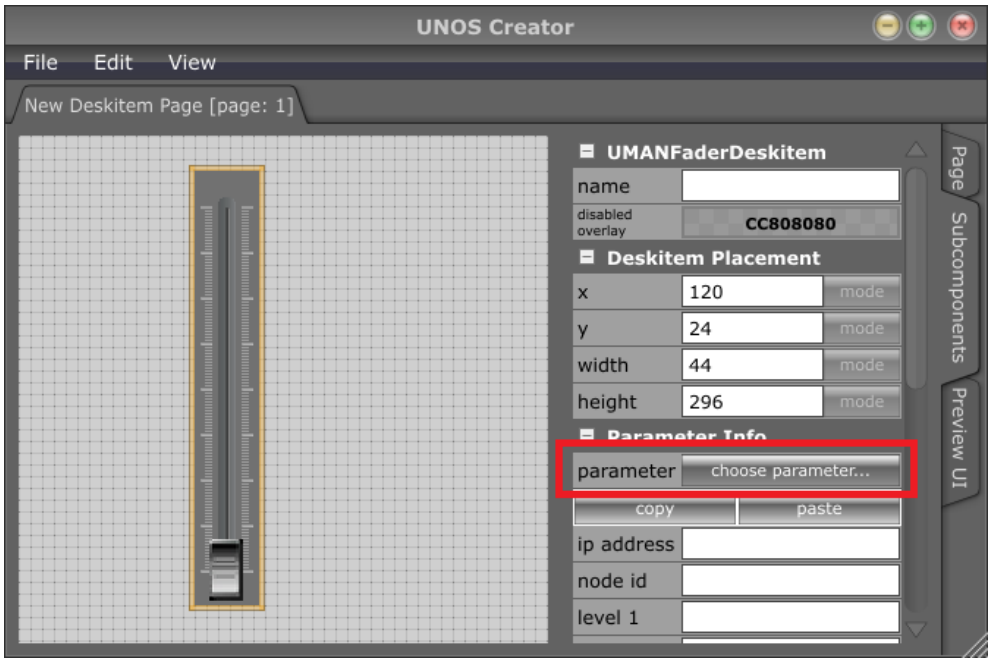


Figure 7.22: Deskitem Remote Parameter Information

(b) A parameter selection window is displayed as shown in Figure 7.23. The parameter selection window has three main sections, namely “XFN Parameter Levels”, “Device Selection”, and “Selected XFN Parameter”. The “Device Selection” section, as the name suggests, enables the selection of a subnet and IP address of the targeted remote device. Upon selection of the remote device’s IP address, a hierarchical tree of the device’s XFN parameters is displayed in the “XFN Parameter Levels” section. Once a parameter is selected in the tree, the hierarchical address of the chosen parameter is displayed in the “Selected XFN Parameter” section and the “Apply” button is pressed to save the selection.

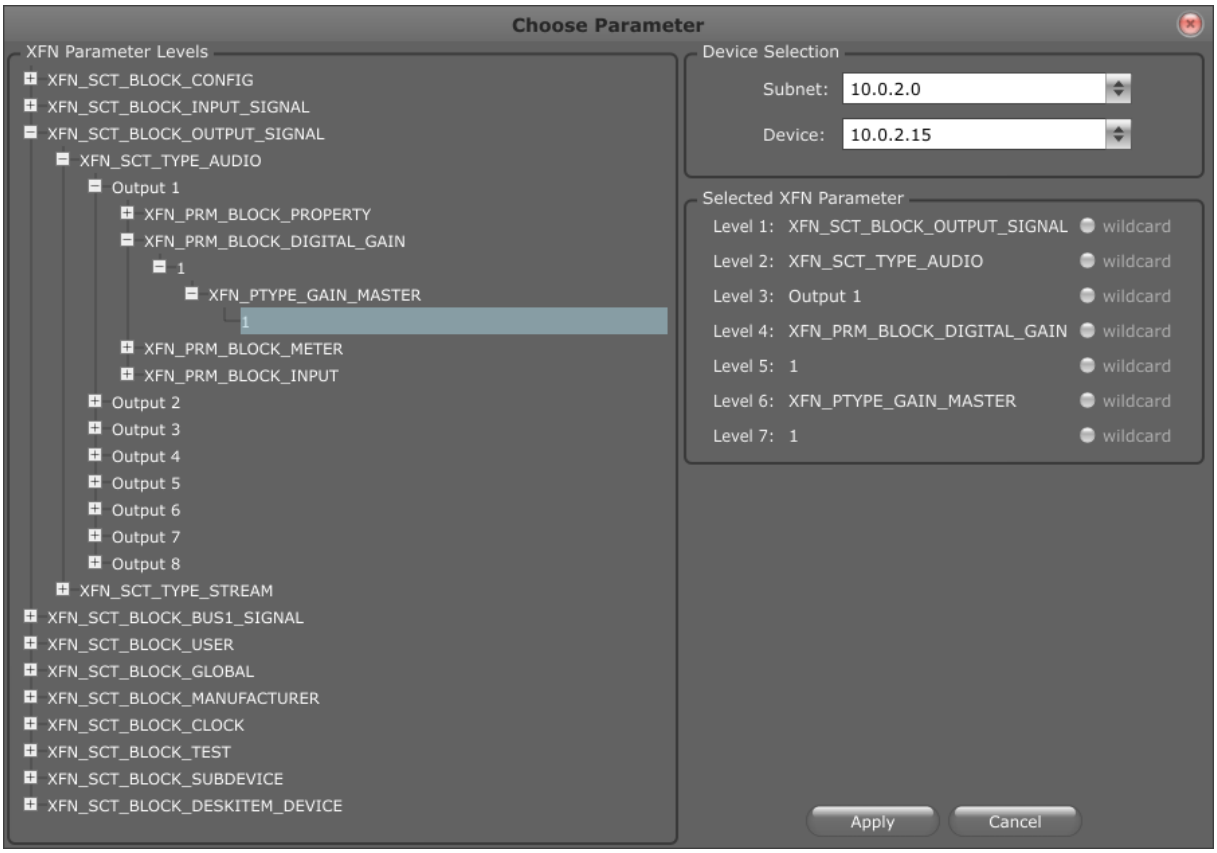


Figure 7.23: Parameter Selection Window

5. Packaging deskitem pages

   Recall from step 1 of the deskitem remote control procedure illustrated in Figure 7.3 on page 161 that deskitem packages, in the form of zip files saved on remote multimedia devices, are downloaded and rendered in Unos Creator. These deskitem packages are created by navigating to the `File | Package | Device deskitem` menu of the editing tool as shown in Figure 7.24 on the next page. Packaging deskitem pages creates a zip archive that can be uploaded to a multimedia device. This zip archive comprises the following folders:

   - xml
   - images
   - global unit tables

   The "xml" folder contains one or more XML files that describe all the deskitem controls

on one or more deskitem pages. The “images” folder contains all the files for the pictures that are used to skin deskitem controls on the deskitem pages. The “global unit tables” folder contains all the mappings from global units to universal units that are required by the deskitems.

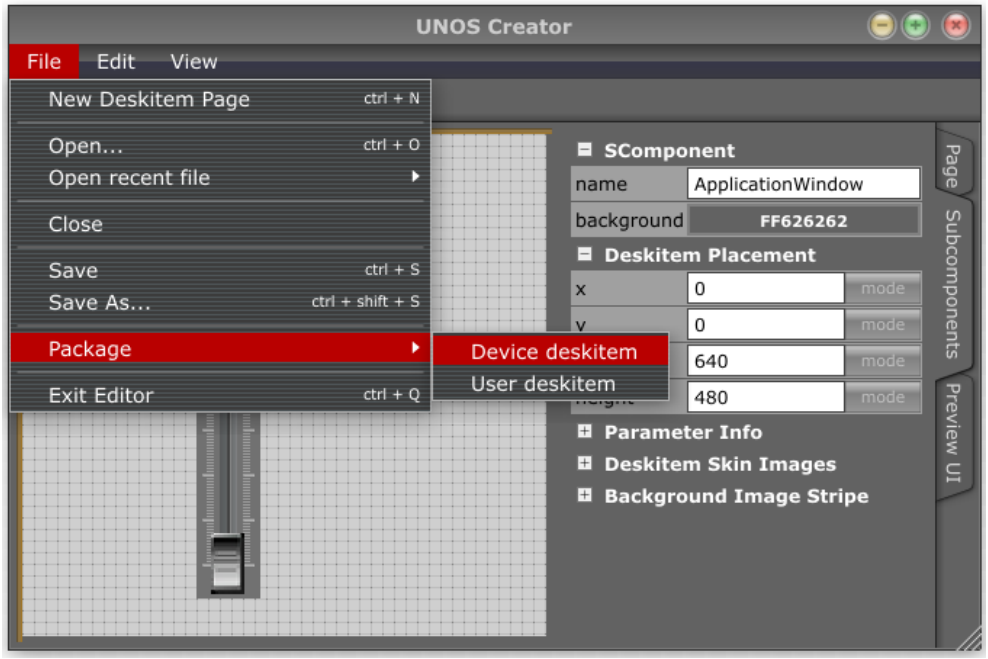


Figure 7.24: Packaging Device Deskitems

7.3 Managing Grouping Relationships

Up to this point, we have been describing how deskitems are constructed and how permanent absolute peer-to-peer relationships are created between deskitem parameters in Unos Creator and the remote parameters on multimedia devices. This section describes how Unos Creator uses deskitems to manage the core relationships described in Section 4.1 on page 68, namely:

- Relative peer-to-peer relationships.
- Absolute peer-to-peer relationships.
- Relative master-slave relationships.
- Absolute master-slave relationships.

The creation of relationships between a group of parameters is a four step process involving the following:

1. Relationship type selection
   The type of relationship to be created is specified as one of relative peer-to-peer, absolute peer-to-peer, relative master-slave, or absolute master-slave.
2. Join parameter selection
   A join parameter is selected; this parameter is the one that is used to determine the manner in which the relationships are created when the joins are applied in step 4.
3. Group member selection
   The parameters to be grouped with the join parameter are selected.
4. Parameter joining
   Relationships between each of the group members and the join parameter are effected by an exchange of the various XFN commands described in Section 5.3.1 on page 84 in the case of peer-to-peer relationships, and Section 5.3.3 on page 90 in the case of master-slave relationships.

The breaking of relationships between a group of parameters is also a four step process comprising:

1. Relationship type selection
   The type of relationship that is to be broken is specified as one of master-slave or peer-to-peer. In contrast to creation, the absolute or relative nature of relationships is not required when breaking relationships.
2. Unjoin parameter selection
   An unjoin parameter is selected; this parameter is used to determine the manner in which relationships are broken in step 4.
3. Group member deselection
   Group members to be removed from the group are deselected.
4. Parameter unjoining
   Relationships between each of the deselected group members and the unjoin parameter are effected by an exchange of the various XFN commands described in Section 5.3.2 on page 89 in the case of peer-to-peer relationships, and Section 5.3.4 on page 92 in the case of master-slave relationships.

Figure 7.25 shows an example of a control surface that is composed of eight fader deskitems labelled Ch1, Ch2, Ch3, Ch4, Master, L1, L2, and M.Gain. There is a control panel to the right of the control surface with buttons labelled ABS (absolute), REL (relative), PTP (peer-to-peer), M-S (master-slave), and edit. The ABS, REL, PTP, and M-S buttons are used when setting up grouping relationships, while the edit button, although it falls outside the scope of this investigation, is used to dynamically change deskitem properties. Examples for the creation/breaking of peer-to-peer and master-slave relationships in Sections 7.3.1 and 7.3.2, respectively, will be based on this example control surface. A mechanism that was implemented in order to enable the creation/breaking of these relationships between parameters that reside in different devices will also be described in Section 7.3.3.

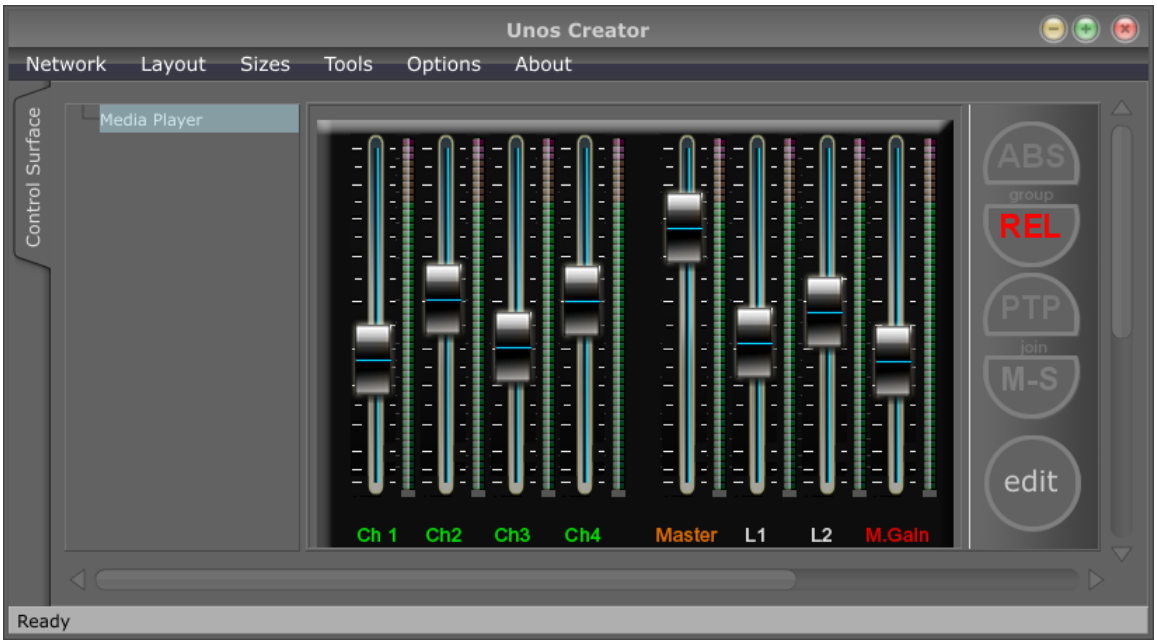


Figure 7.25: Unos Creator - Fader Control Surface

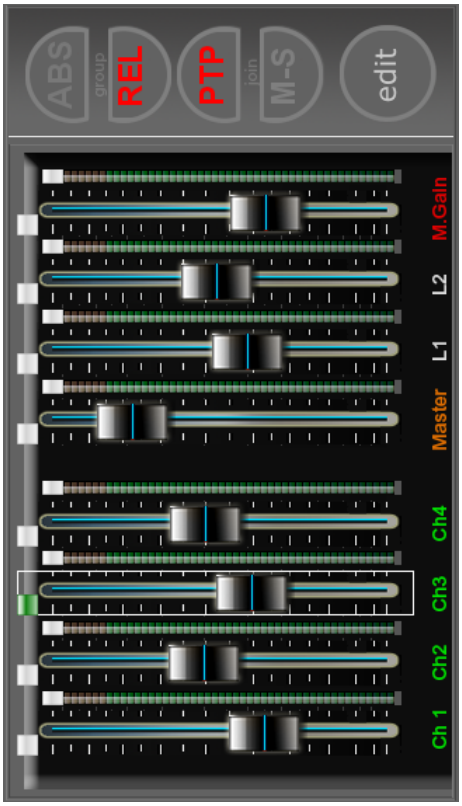
7.3.1 Peer-to-Peer Relationships

With the aid of examples, this section gives step-by-step descriptions of how relative peer-to-peer relationships are created and broken using Unos Creator. While the examples focus on relative peer-to-peer relationships, similar steps are followed when creating absolute peer-to-peer relationships.

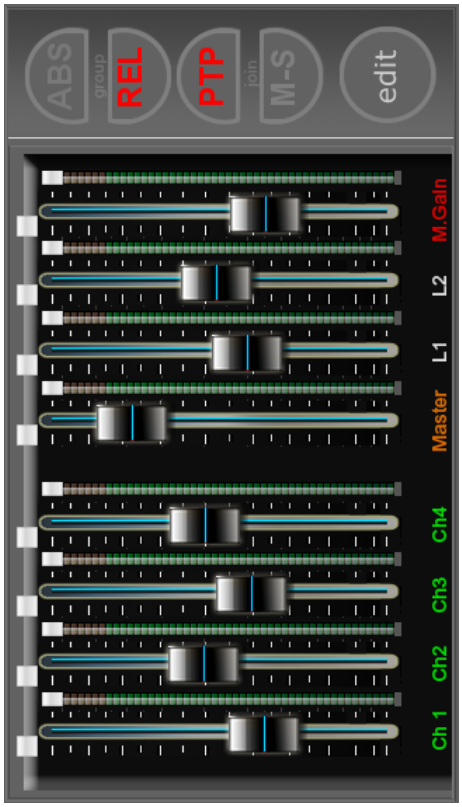
7.3.1.1 Example of Creating Peer-to-Peer Relationships

Using the example control surface shown in Figure 7.25, we now describe the four steps that are followed when creating relative peer-to-peer relationships between parameters associated with the deskitems Ch2, Ch3, and Ch4, where:

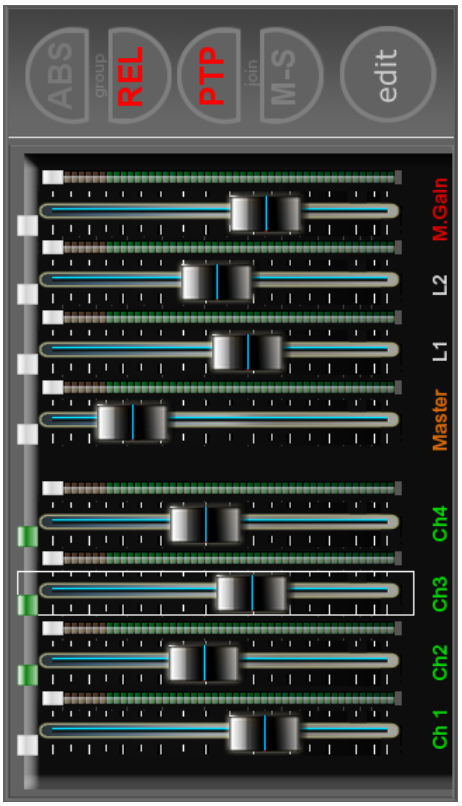
1. Relationship type selection
   As shown in Figure 7.26a, the relative nature of the new relationship is specified by switching on the REL button. In addition, the peer-to-peer relationship type is selected by switching on the PTP button. Once the PTP button is switched on, the control surface transitions into join mode, where white group indicator buttons are displayed on the top left corner of each deskitem on the control surface as shown in the figure.
2. Join parameter selection
   The Ch3 deskitem is selected as the join parameter (by clicking over the deskitem). Figure 7.26b shows the Ch3 deskitem with a white border, resulting from this selection. In addition the group indicator for the Ch3 deskitem is green, indicating that it is the only parameter in its peer parameter group.
3. Group member selection
   The group indicators for deskitems Ch2 and Ch4 are selected, thereby switching them to green as shown in Figure 7.26c. This indicates that peer-to-peer relationships will be created between parameters associated with deskitems Ch2, Ch3, and Ch4.
4. Parameter joining
   After all group members are selected in step 3, the relationships are applied by clicking on the PTP button again. Relative peer-to-peer relationships are created between the join parameter (specified in step 2) and each of the selected group members, namely between Ch3 and Ch2, and between Ch3 and Ch4. By implication, peer-to-peer relationships are also created between parameters Ch4 and Ch2 together with their associated deskitem parameters. The PTP button switches off and the control panel transitions out of join mode, where the group indicators are hidden as shown in Figure 7.26d.



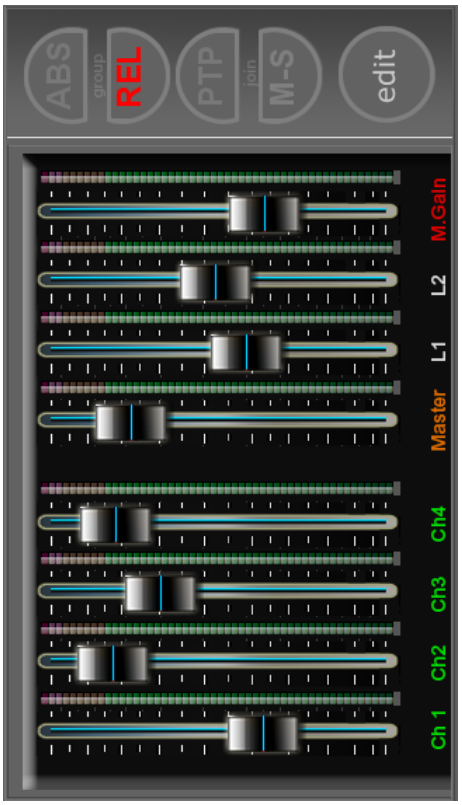
(a) Step 1: Relationship Type Selection



(b) Step 2: Join Parameter Selection



(c) Step 3: Group Member Selection



(d) Step 4: Apply Joins

Figure 7.26: Creating Relative Peer-to-Peer Relationships in Unos Creator

Figure 7.27 shows the state of relationships after the parameters Ch2, Ch3, and Ch4 are joined. The diagram shows a multimedia device that hosts parameters Ch2, Ch3, and Ch4. An instance of Unos Creator is also shown with three parameters, D1, D2, and D3 that are bound to deskitems for Ch2, Ch3, and Ch4, respectively. Recall from step 3 of the deskitem remote control procedure illustrated in Figure 7.3 on page 161 that absolute peer-to-peer relationships are created between each deskitem parameter and the remote parameter in order to achieve remote control. These relationships are shown in the diagram between D1 and Ch2, D2 and Ch3, and D3 and Ch4. A peer-to-peer group comprising parameters Ch2, Ch3, Ch4, D1, D2, and D3 is formed after joining Ch2, Ch3, and Ch4.

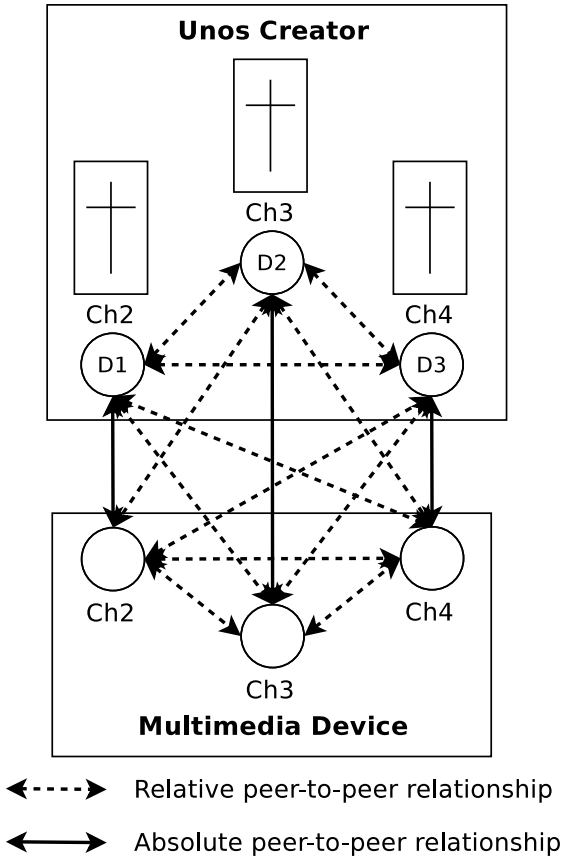


Figure 7.27: Peer-to-Peer Relationship State After Parameter Joining

7.3.1.2 Example of Breaking Peer-to-Peer Relationships

Based on the results of the example described in section 7.3.1.1, where relative peer-to-peer relationships exist between parameters bound to deskitems Ch2, Ch3, and Ch4, we now describe

the four steps that are followed when breaking these relative peer-to-peer relationships.

1. Relationship type selection
   The peer-to-peer relationship type is selected by switching on the PTP button as shown in Figure 7.28a. The value of REL button is ignored when breaking relationships. Once the PTP button is switched on, the control surface transitions into join mode, where white group indicator buttons are displayed on the top left corner of each deskitem on the control surface as shown in the figure.
2. Unjoin parameter selection
   Any of the grouped deskitems is selected as the unjoin parameter (by clicking over the deskitem). Figure 7.28b shows the Ch2 deskitem with a white border, resulting from this selection. The group indicators for the Ch2, Ch3, and Ch4 deskitems become illuminated in green, indicating that all three parameters have peer-to-peer relationships between them.
3. Group member deselection
   The group indicators for deskitems Ch2, Ch3, and Ch4 are deselected, and thus switched off as shown in Figure 7.28c. This indicates that peer-to-peer relationships are to be broken between parameters associated with deskitems Ch2, Ch3, and Ch4.
4. Parameter unjoining
   After all desired group members are deselected in step 3, the relationships are broken by clicking on the PTP button again. Parameters bound to the deskitems that are deselected in step 3 are removed from the peer-to-peer group shown in step 2, namely Ch2, Ch3, and Ch4. The PTP button switches off and the control panel transitions out of join mode, where the group indicators are hidden as shown in Figure 7.28d.

(a) Step 1: Relationship Type Selection

(b) Step 2: Unjoin Parameter Selection

(c) Step 3: Group Member Deselection

(d) Step 4: Unjoin Parameters

Figure 7.28: Breaking Peer-to-Peer Relationships in Unos Creator

Recall from the rules defined in Section 5.5.3.2 on page 116 that whenever relationships are broken, all permanent relationships between controller parameters and device parameters are preserved until they are explicitly broken by the controller. Following from Figure 7.27 on page 190, Figure 7.29 shows the state of parameter relationships after the relative peer-to-peer relationships between parameters Ch2, Ch3, and Ch4 are broken. The diagram shows that all relationships have been broken except for the permanent absolute peer-to-peer relationships between parameters D1 and Ch2, D2 and Ch3, and between D3 and Ch4.

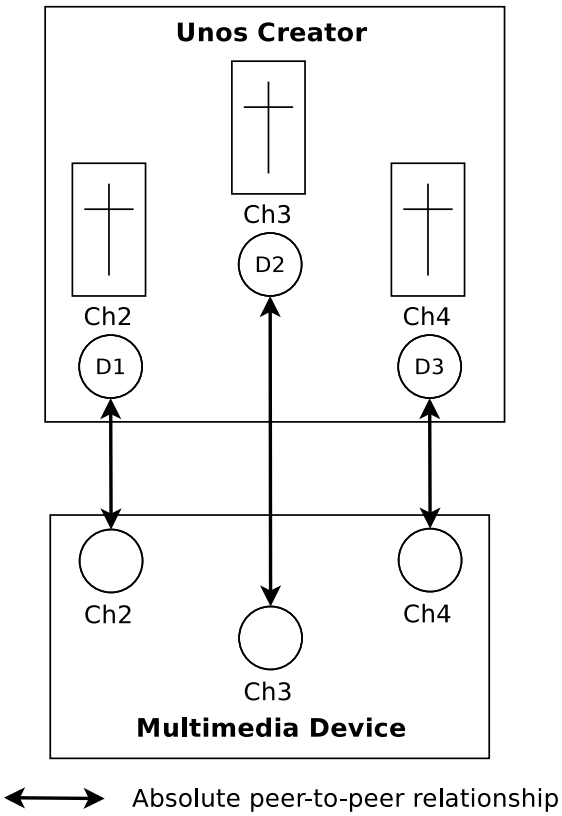


Figure 7.29: Peer-to-Peer Relationship State After Parameter Unjoining

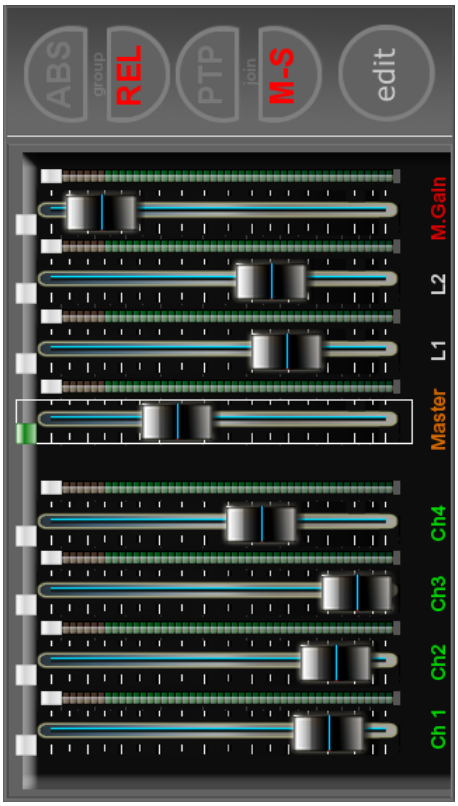
7.3.2 Master-Slave Relationships

This section gives step-by-step examples illustrating how master-slave relationships are created and broken using Unos Creator. Although the examples focus on relative master-slave relationships, similar steps are followed when creating absolute master-slave relationships.

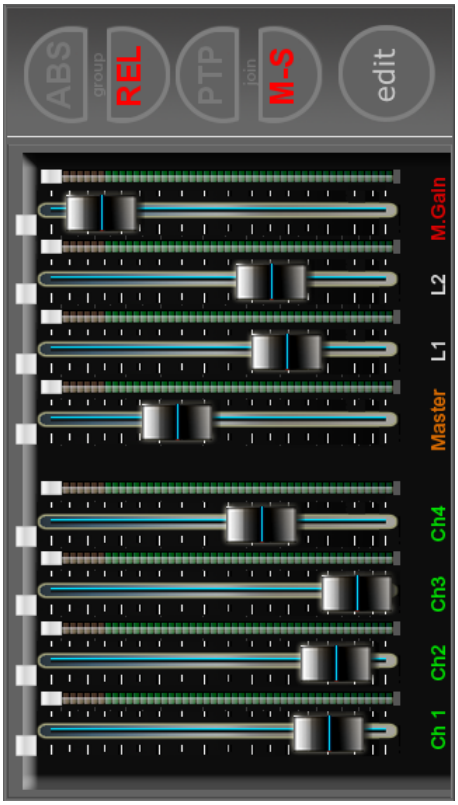
7.3.2.1 Example of Creating Master-Slave Relationships

Using the example control surface shown in Figure 7.25, we now describe the four steps that are followed when creating relative master-slave relationships between parameters associated with the deskitems Ch4, Master, and M.Gain, where M.Gain is master parameter to Master, while Master is master parameter to Ch4. This is a typical example of a cascaded master-slave relationship chain, where M.Gain is a grand-master to Ch4.

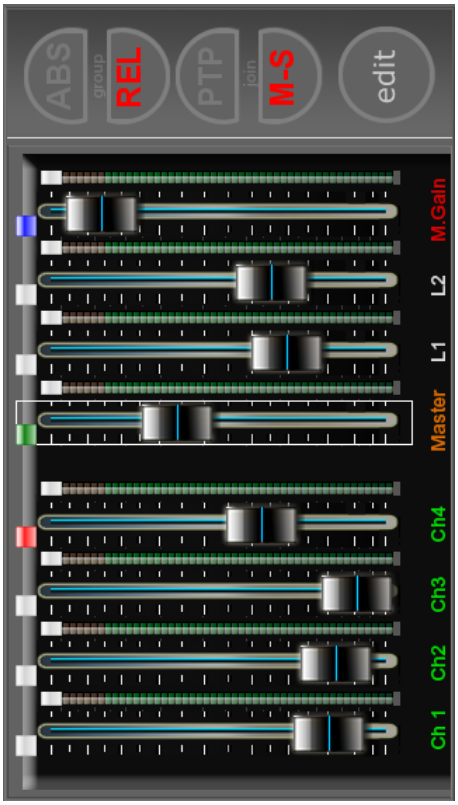
1. Relationship type selection
   As shown in Figure 7.30a, the relative nature of the new relationship is specified by switching on REL button. In addition, the master-slave relationship type is selected by switching on the M-S button. Once the M-S button is switched on, the control surface transitions into join mode, where white group indicator buttons are displayed on the top left corner of each deskitem on the control surface as shown in the figure.
2. Join parameter selection
   The Master deskitem is selected as the join parameter (by clicking over the deskitem). Figure 7.30b shows the Master deskitem with a white border, resulting from this selection. In addition the group indicator for the Master deskitem is green, indicating that it is the only parameter in its peer parameter group.
3. Group member selection
   The group indicator for the Ch4 deskitem is toggled until it is red as shown in Figure 7.30c. This indicates that a master-slave relationship will be created between parameters associated with the selected join parameter (Master) and Ch4, where Ch4 is a slave parameter. In contrast, the group indicator for the M.Gain deskitem is toggled until it is blue as shown in the figure. This indicates that a master-slave relationship will be created between parameters associated with the selected join parameter deskitem (Master) and M.Gain, where M.Gain is the master.
4. Join parameters
   After all group members are selected in step 3, the relationships are applied by clicking on the M-S button again. Relative master-slave relationships are created between the join parameter (specified in step 2) and each of the selected master or slave group members, namely between Ch4 (slave role) and Master (master role), and between Master (slave role) and M.Gain (master role). The M-S button switches off and the control panel transitions out of join mode, where the group indicators are hidden as shown in Figure 7.30d.



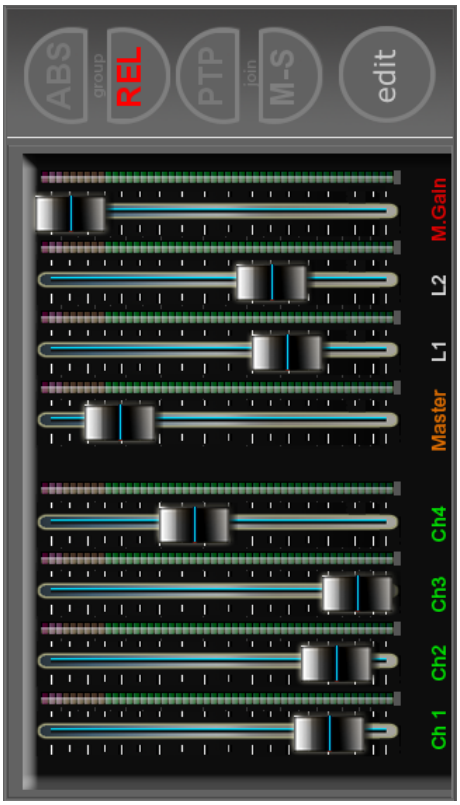
(a) Step 1: Relationship Type Selection



(b) Step 2: Join Parameter Selection



(c) Step 3: Group Member Selection



(d) Step 4: Apply Joins

Figure 7.30: Creating Relative Master-Slave Relationships in Unos Creator

Figure 7.31 shows the state of relationships after parameters Ch4, Master, and M.Gain have been joined, where Ch4 is slave of Master, while Master is slave of M.Gain. The diagram shows a multimedia device that hosts parameters Ch4, Master, and M.Gain. An instance of Unos Creator is also shown with three parameters, D3, D4, and D5 that are bound to deskitem controls for Ch4, Master, and M.Gain, respectively. Notice that absolute peer-to-peer relationships exist between each deskitem parameter and its remote counterpart, namely between D3 and Ch4, D4 and Master, and D5 and M.Gain. Recall from Section 5.5.1.2.1 on page 107 that when a parameter is made master of a member of a peer-to-peer group, the parameter becomes master of all other peer-to-peer group members. Thus, when Master becomes master of Ch4, Master also becomes master of D3 by virtue of the absolute peer-to-peer relationship between Ch4 and D3. Similarly, when M.Gain becomes master of Master, M.Gain becomes master of D4 by virtue of the absolute peer-to-peer relationships between Master and D4.

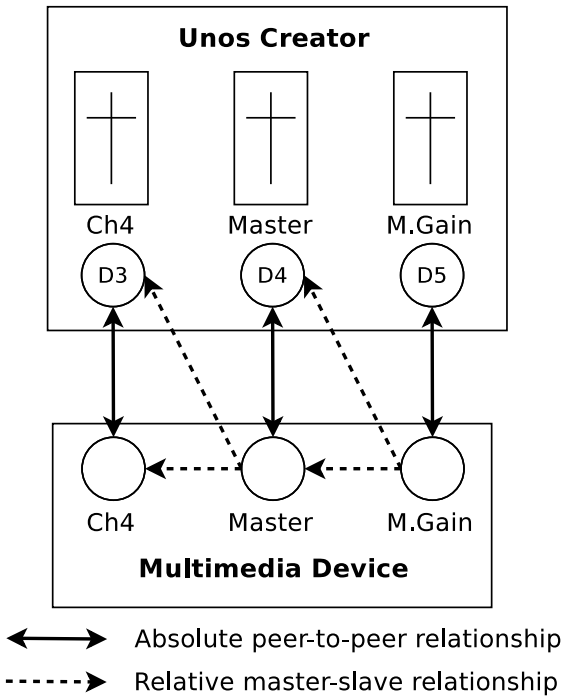


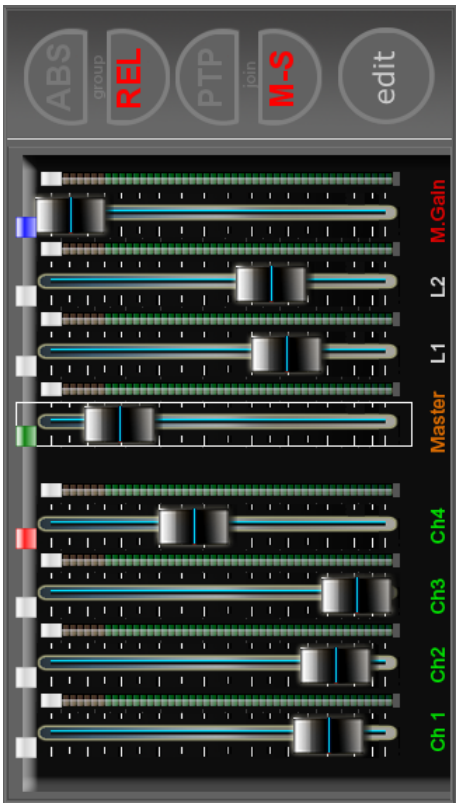
Figure 7.31: Master-Slave Relationship State After Parameter Joining

7.3.2.2 Example of Breaking Master-Slave Relationships

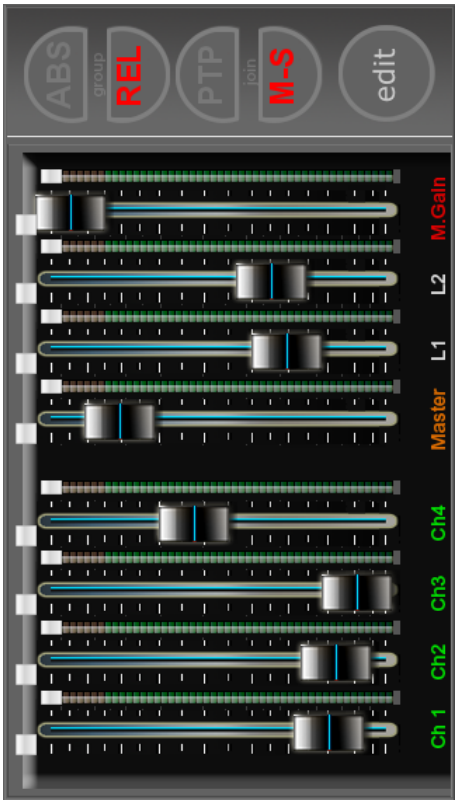
Based on the results of the example described in Section 7.3.2.1, where relative master-slave relationships exist between parameters bound to deskitems Master (master role) and Ch4 (slave

role), and between M.Gain (master role) and Master (slave role), we now describe the four steps that are followed when breaking these relative master-slave relationships.

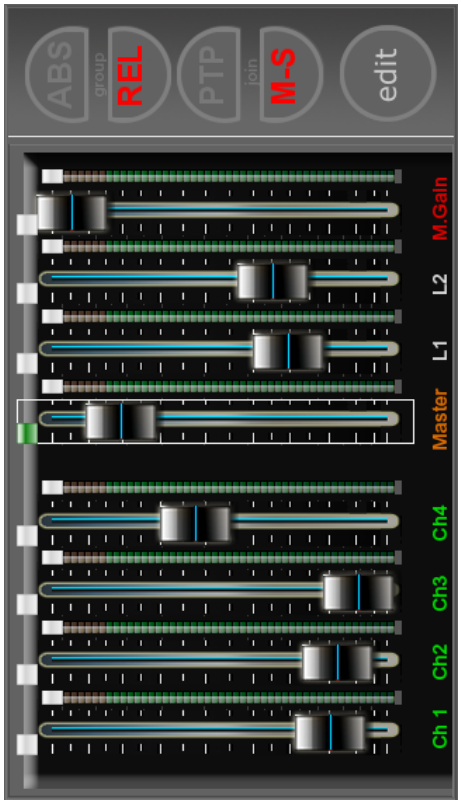
1. Relationship type selection
   The master-slave relationship type is selected by switching on the M-S button as shown in Figure 7.32a. The values of REL and ABS buttons are ignored when breaking relationships. Once the M-S button is switched on, the control surface transitions into join mode, where white group indicator buttons are displayed on the top left corner of each deskitem on the control surface as shown in the figure.
2. Unjoin parameter selection
   The deskitem with the relationships to be broken is selected as the unjoin parameter (by clicking over the deskitem). Figure 7.32b shows the Master deskitem with a white border, resulting from this selection. The group indicator for the Ch4 deskitem becomes illuminated in red, indicating Ch4 is a slave of Master, while the group indicator for M.Gain is illuminated in blue, indicating that M.Gain is a master of Master.
3. Group member deselection
   The group indicator for the Ch4 deskitem is deselected, and thus switched off as shown in Figure 7.32c. This indicates that the master-slave relationship between the selected unjoin parameter and Ch4 will be broken. Similarly, the group indicator for the M.Gain deskitem is deselected, indicating that the master-slave relationship between the selected unjoin parameter and M.Gain will be broken.
4. Unjoin parameters
   After all desired group members are deselected in step 3, the relationships are broken by clicking on the M-S button again. The relationships between the parameters bound to the deskitems that are deselected in step 3 and the unjoin parameter that is selected in step 2 are broken. The M-S button switches off and the control panel transitions out of join mode, where the group indicators are hidden as shown in Figure 7.32d.



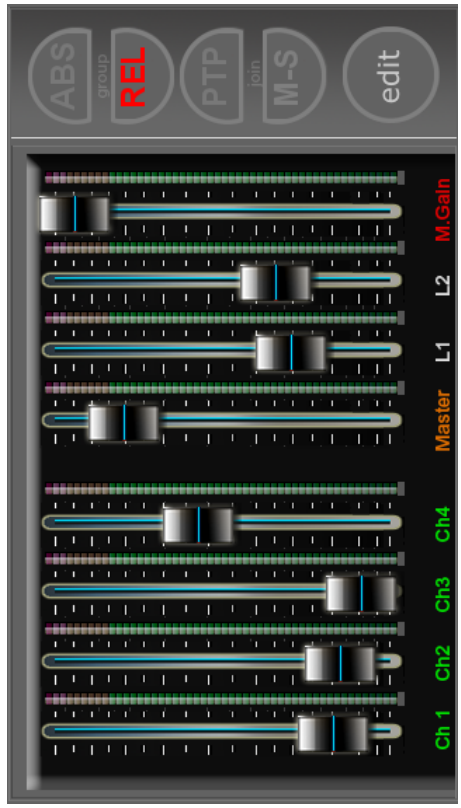
(a) Step 1: Relationship Type Selection



(b) Step 2: Unjoin Parameter Selection



(c) Step 3: Group Member Deselection



(d) Step 4: Unjoin Parameters

Figure 7.32: Breaking Master-Slave Relationships in Unos Creator

Following from Figure 7.31 on page 196, Figure 7.33 shows the state of parameter relationships after the master-slave relationships between parameters Ch4 and Master, and between Master and M.Gain are broken. Recall from Section 5.5.1.2.2 on page 108 that when a master-slave relationship is broken between a master parameter and a slave that is part of a peer-to-peer relationship group, the other master-slave relationships that exist with the slave's peers are also broken. Thus, when the master-slave relationship between Master and Ch4 is broken, the master-slave relationship between Master and D3 is also broken. Similarly, when the master-slave relationship between M.Gain and Master is broken, the master-slave relationship between M.Gain and D4 is also broken. As shown in the figure, only the absolute peer-to-peer relationships between the deskitem parameters and their corresponding remote counterparts remain, namely between D3 and Ch4, between D4 and Master, and between D5 and M.Gain.

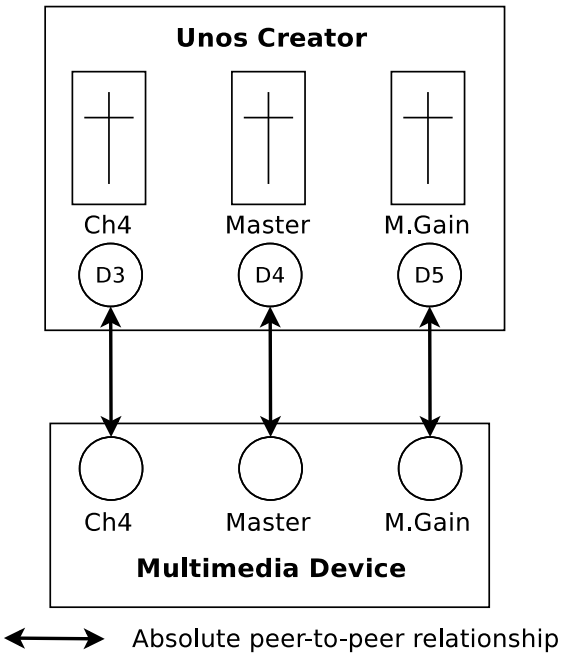


Figure 7.33: Relationship State After Master-Slave Parameter Unjoining

7.3.3 Creating Relationships Between Parameters on Different Devices

Sections 7.3.1 and 7.3.2 have described the steps that are followed when creating and breaking relationships between parameters via a control surface. Control surfaces in this context comprise deskitems that are associated with parameters in a single device. In order to make relationships between parameters residing in different devices, Unos Creator provides a mechanism where

custom control surfaces can be built from deskitems that are associated with parameters of different devices. In particular, a drag-and-drop mechanism has been implemented, where deskitems from device-specific control surfaces are dragged and dropped onto a custom control surface. Figure 7.34 on the next page illustrates this mechanism with three control surfaces, where:

- The top-left fader control surface comprises fader and meter deskitems.
- The larger media player control surface on the right comprises fader, display, multi-I/O, pot, and meter deskitems.
- The bottom-left custom control surface comprises two fader deskitems from the fader control surface as well as a pot deskitem from the media player control surface. The deskitems on the custom control surface are associated as follows:
    - The fader deskitem on the left is a copy of M.Gain from the fader control surface.
    - The fader deskitem on the right of the copy of M.Gain is a copy of Ch1 from the fader control surface.
    - The pot deskitem is a copy of the Low Gain (L.Gain-EQ1) pot deskitem of the left-most EQ from the media player control surface.

Using the steps described in Sections 7.3.1 and 7.3.2 it is possible to create or break relationships between parameters on different devices via their corresponding deskitems on the custom control surface. The custom control surface in Figure 7.34 shows an example where there is a peer-to-peer relationship between one of the fader parameters, Ch1, and the pot parameter, L.Gain-EQ1, shown by the green join indicators. In addition, master-slave relationships (master shown by the blue join indicator) exist between one of the fader parameters, M.Gain, and each of parameters Ch1 and L.Gain-EQ1 in the peer-to-peer group. Recall from Section 5.5.1.2.1 on page 107 that when a master-slave relationship is made with one member of the peer-to-peer group, the master relationship is with all peer-to-peer group members.

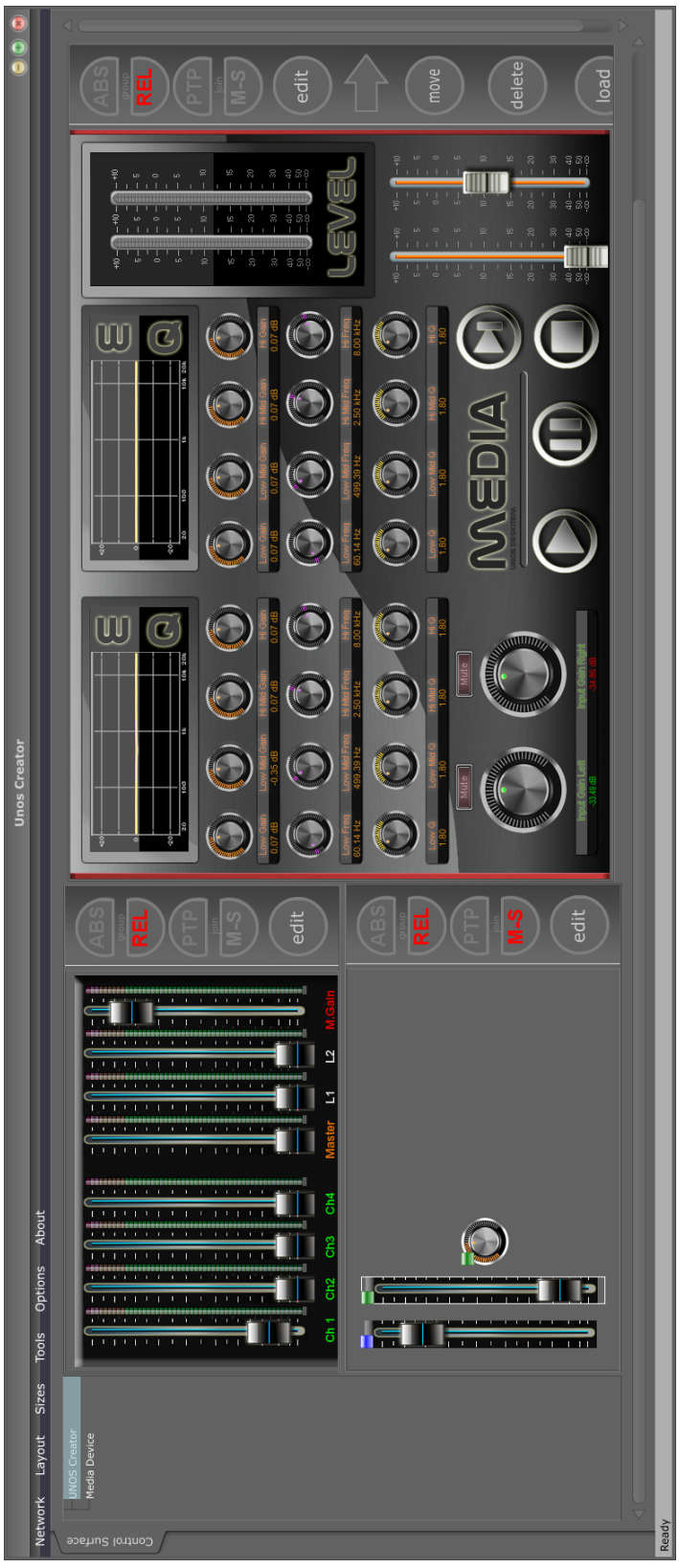


Figure 7.34: Custom Control Surface in Unos Creator

Figure 7.35 shows a representation of relationships between the deskitem parameters of the custom control surface described above and their corresponding remote parameters in two multimedia devices. Parameters M.Gain, Ch1, and L.Gain-EQ1 are associated with deskitem parameters D6, D7, and D8, respectively. M.Gain and Ch1 reside in one multimedia device, namely "Multimedia Device 1", while L.Gain-EQ1 resides in another multimedia device, namely "Multimedia Device 2". As with all deskitems, absolute peer-to-peer relationships exist between the deskitem parameters and their remote counterparts, namely between D6 and M.Gain, between D7 and Ch1, and between D8 and L.Gain-EQ1. A peer-to-peer relationship exists between parameters Ch1 and L.Gain-EQ1, resulting in a peer-to-peer parameter group comprising D7, D8, Ch1, and L.Gain-EQ1. Consequently, parameter M.Gain becomes master of all the parameters in the peer-to-peer relationship group. Notice how these relationships span across the three different devices.

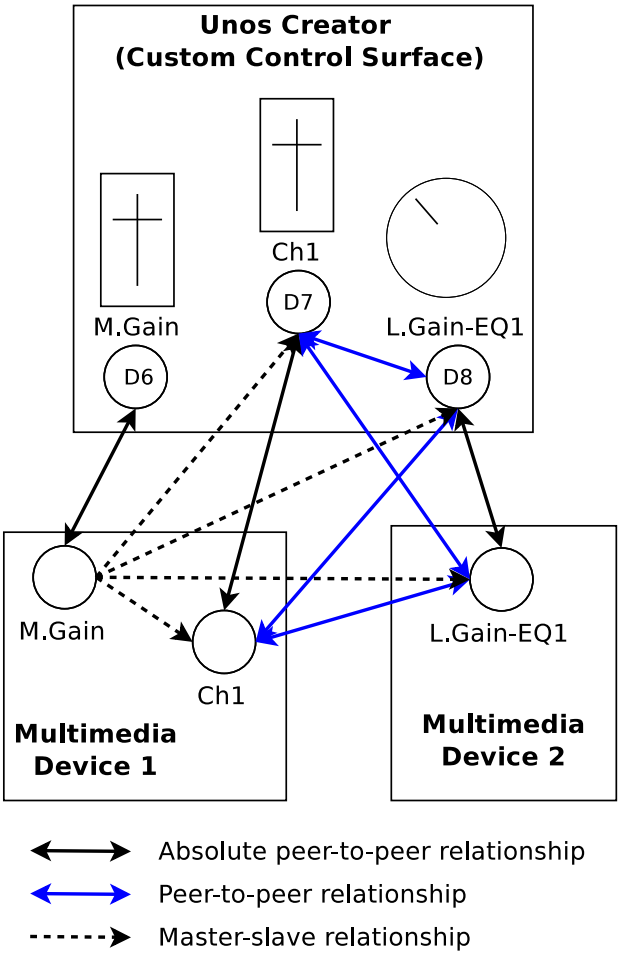


Figure 7.35: Joining Parameters in Different Devices

7.4 Parameter Monitoring

Section 6.2 on page 135 has already given a detailed description of the implementation strategies for the push mechanism that is used for parameter monitoring within the XFN protocol. These implementation strategies are the basis of parameter monitoring within Unos Creator. For brevity, these descriptions will not be repeated in this section. All the information necessary for parameter monitoring to function is contained in the deskitem XML files. Unlike parameter grouping, where user intervention is required to group parameters, there is no user intervention required in order to initiate parameter monitoring after the deskitems have been loaded by Unos Creator. From a Unos Creator perspective, the following steps are carried out:

1. Meter deskitems are loaded and each deskitem is associated with a local XFN parameter.
2. A push-handler XFN parameter is created within Unos Creator. The push-handler parameter keeps mappings between Unos Creator's meter deskitem XFN parameters and the corresponding remote device's monitored XFN parameters.
3. The push-handler parameter is subscribed as a "listener" for value changes in any of the monitored parameters in the remote device.
4. Upon changes in value of any of the monitored parameters in the remote device, the push-handler parameter receives a list containing the latest values of all the monitored parameters. The push-handler parameter then updates each of the meter deskitem parameters based on the mappings in step 2, resulting in an update of the associated graphical meter components.

7.5 Test Environment

The parameter grouping and monitoring capabilities that were implemented in this study were tested in a laboratory set up with evaluation hardware supplied by Universal Media Access Networks GmbH (UMAN). At the time of this investigation, UMAN was the only company with an XFN protocol stack implementation. This XFN protocol stack implementation was enhanced to support parameter grouping and monitoring as part of this study.

Figure 7.36 shows the nature of the laboratory set up with the following:

- Two IP subnets, each with
    - Two firewire-enabled evaluation hardware boards that implement IP-over-firewire. These evaluation boards perform functions of an audio source or an audio receiver.
    - A workstation running the enhanced Unos Creator application.
- A router that was custom-designed by UMAN to support IP-over-firewire.

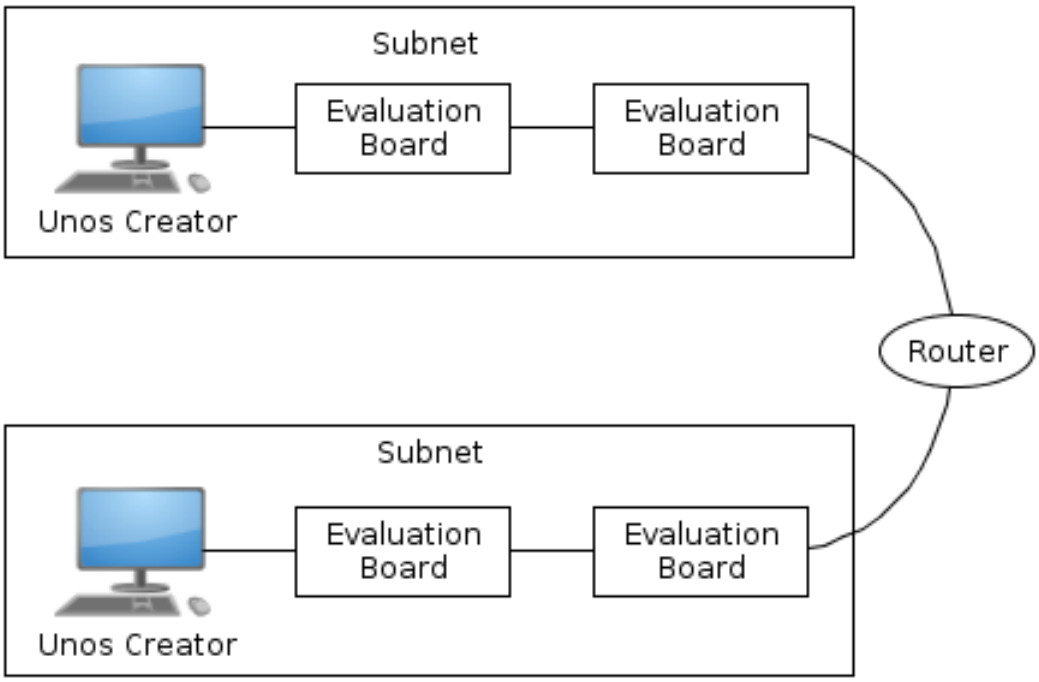


Figure 7.36: Test Environment

The evaluation boards tested had a Digital Interface Communications Engine (DICE) [TC Applied Technologies, Ltd., 2007] chip with the following features:

- AES/EBU receivers capable of receiving up to eight audio channels.
- AES/EBU transmitters capable of transmitting up to eight audio channels.
- Alesis ADAT compatible receivers capable of receiving up to eight audio channels.
- Alesis ADAT compatible transmitters capable of transmitting up to eight audio channels.

- 1394 Audio Video System (AVS) capable of receiving/transmitting up to 16 audio channels over the firewire network.
- A mixer matrix with 18 inputs and 16 outputs, where any input audio channel can be routed to a mixer input, while any mixer output can be routed to any output audio channel. Figure 7.37 shows the layout of the mixer matrix. Each crosspoint in the figure is labelled (x, y), where x is the mixer input number, while y is the corresponding mixer output number. Each crosspoint has a gain and on/off parameter. The gain is controlled by a 16 bit coefficient. The on/off parameter indicates if the associated mixer input audio signal is included in the associated mixer output sub-mix, while the gain parameter determines the magnitude of amplification or attenuation of the mixer input signal. XFN parameters that are associated with these gain and on/off parameters were controlled by deskitems.
- A router module that processes all audio transfers from audio inputs to audio outputs. For example, when audio is routed from an AES/EBU receiver to an input of the mixer matrix, a router entry is created in the router table. Each router entry is associated with a peak value that keeps track of the maximum amplitude of every audio sample. XFN parameters associated with these peak values were monitored using deskitems.

		To audio transmitter → Mixer Out 1	To audio transmitter → Mixer Out 2	..	To audio transmitter → Mixer Out 16
From audio receiver →	Mixer In 1	(1, 1)	(1, 2)	..	(1, 16)
From audio receiver →	Mixer In 2	(2, 1)	(2, 2)	..	(2, 16)
:	:	:	:	:	:
From audio receiver →	Mixer In 18	(18, 1)	(18, 2)	..	(18, 16)

Figure 7.37: DICE Mixer Matrix

7.5.1 Deskitem Control Panel

Figure 7.38 shows a deskitem control panel that was used to control the DICE mixer matrix.

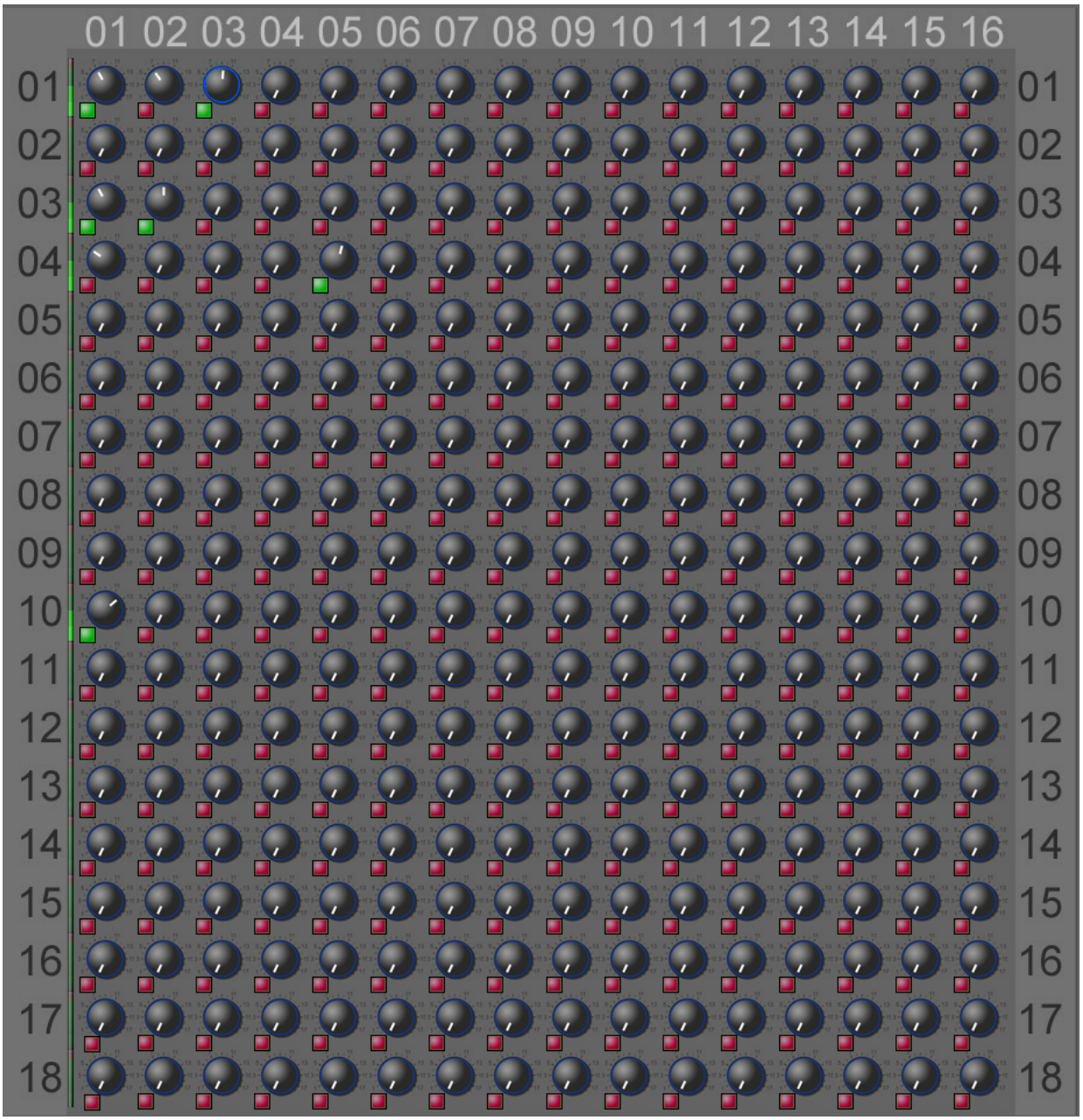


Figure 7.38: DICE Mixer Matrix Deskitem

The deskitem control panel has 18 rows (numbered 01 up to 18) and 16 columns (numbered 01 up to 16), corresponding to the 18 inputs and 16 outputs of the mixer, respectively. Each cross-

point in the deskitem control panel has a pot deskitem and a button deskitem. Buttons are red when in the "off" state and green when in the "on" state. In the figure, buttons for crosspoints (1,1), (1,3), (3,1), (3,2), (4,5) and (10,1) are in the "on" state. The pot deskitem was used to control the gain for a given crosspoint, while the button deskitem was used to control the on/off parameter of the crosspoint. Meter deskitems, the green bars next to each row label, were used to monitor the peak values of audio signals that were routed from audio receivers to each mixer input. In the figure, meters for inputs 01, 03, 04, and 10 are illuminated, indicating the presence of audio signals at the inputs.

7.5.2 Global Unit Value Mapping Tables

On/off values were mapped directly to the global unit value range, since they are not associated with any SI units; values less than 50% of the global unit value range correspond to the "off" state, while values greater than or equal to 50% of the global unit value range correspond to the "on" state. It was necessary for gain coefficients and peak values to be interpreted in terms of global unit values. Recall from Section 5.4.3.3 on page 100 that mapping tables are required to map, for example, decibel values to global unit values. Thus, gain coefficients and peak values were first converted to their decibel equivalent. The conversions that were made for the evaluation boards tested are described below.

7.5.2.1 Gain Coefficients

The gain coefficients use a 2:14 binary fix point notation [TC Applied Technologies, Ltd., 2007]. The coefficients determine the factor by which mixer input audio samples are amplified. Table 7.8 shows the conversions from coefficients to their decibel equivalent, where the minimum gain is -84 dB, while the maximum gain is +12 dB. These decibel values were mapped to global unit values as described in Section 5.4.3.3.1 on page 100.

Coefficient (hex)	**Coefficient (2:14 Binary Fix Point)**	**Multiplication Factor (derived from coefficient)**	**dB** $20 * log_{10}\, multiplication\ factor$
0x0001	00.00000000000001	2^{-14}	-84
:	:	:	:
0x4000	01.00000000000000	1	0 (unity gain)
:	:	:	:
0x8000	10.00000000000000	2	+6
:	:	:	:
0xFFFF	11.11111111111111	≈ 4	+12

Table 7.8: Gain Coefficient to dB Mapping

7.5.2.2 Peak Values

The peak values for router entries are derived from the most significant 12 bits of the 24-bit audio data. Thus, corresponding dBFS value for a peak value, *x*, is calculated using the following formula:

$$20 * log_{10} \frac{x}{2^{12}}$$

Table 7.9 shows this conversion from peak value to dBFS. These dBFS values were mapped to global unit values as described in Section 5.4.3.3.2 on page 101.

Peak Value (hex)	**Peak Value (decimal)**	**dBFS**
0x000	0	$-\infty$
0x001	1	-72
:	:	:
0x19A	410	-20
:	:	:
0x800	2048	-6
:	:	:
0xFFF	4095	0

Table 7.9: Peak Value to dBFS Mapping

7.5.3 Nature of Tests Carried Out

The deskitem control panel shown in Section 7.5.1 was loaded in each instance of Unos Creator for all evaluation boards. Upon loading the deskitem control panels, permanent absolute peer-to-peer relationships were created between deskitem XFN parameters and corresponding XFN

parameters of the evaluation boards. The tests carried out showed that deskitems of all instances of Unos Creator remained synchronised whenever the values of parameters in the evaluation boards were modified. When a device XFN parameter was modified via a deskitem in one Unos Creator, the change was mirrored in the other Unos Creator(s) controlling the same device. Thus, bi-directional remote control was achieved.

Relationships were created between gain XFN parameters:

- In a single evaluation board.
- In two evaluation boards in the same subnet.
- In two evaluation boards in different subnets.

Peer-to-peer relationships, relative and absolute, were created by following the procedure described in Section 7.3.1.1 on page 188. These relationships were successfully broken by following the procedure described in Section 7.3.1.2 on page 190. Master-slave relationships, relative and absolute, were created by following the procedure described in Section 7.3.2.1 on page 194. Master-slave relationships were successfully broken by following the procedure described in Section 7.3.2.2 on page 196. Relationships between parameters residing in different devices were enabled by the ability to create custom control surfaces that combine deskitems from different devices as described in Section 7.3.3 on page 199.

The parameter grouping rules described in Section 5.5 on page 104 were validated by creating specific combinations of master-slave and peer-to-peer relationships. The rules that were validated include:

- Hybridisation of master-slave and peer-to-peer relationships, where
    - The creation of a master-slave relationship with a slave that is part of a peer group, results in a master-slave relationship between the master parameter and each of the slave's peers.
    - Breaking a master-slave relationship with a slave that is part of a peer group results in breaking the master-slave relationship between the master parameter and each of the slave's peers.
    - Creating a peer-to-peer relationship between peers that have separate masters merges the masters such that all peers share the same master parameters.

- Absolute/relative relationship inference, where the relationship combinations listed in Table 5.9 on page 114 were tested.
- Retention of critical relationships, where Unos Creator maintained remote control over device parameters after creating and breaking relationships between remote parameters.

It was possible to create relationships between gain XFN parameters and on/off XFN parameters, within the confines of the dual-state behaviour of on/off parameters. In particular, when a gain XFN parameter was joined to an on/off XFN parameter, only value changes of the gain XFN parameter that cause a transition in value of the on/off parameter across the 50% global unit value mark result in a change in "state" of the on/off XFN parameter. Nevertheless, this is an example of relationships between parameter with disparate underlying value types.

Peak values were monitored successfully via meter deskitems. Enabling or disabling[2] audio routes to the various mixer inputs resulted in meters illuminating or fading out, respectively.

7.6 Comparison of Network Parameter Grouping Mechanisms with Mixing Console Capabilities

This chapter has shown how deskitems implemented in Unos Creator expose the parameter grouping and monitoring mechanisms described in Chapter 5 and Chapter 6, respectively, to end users. This section describes the extent to which these mechanisms allow a distributed system to function as an integrated system (mixing console).

Figure 7.39 depicts a mixing console as an integrated system, where relationships (groupings) between channel controls are only possible within the same device. In contrast, Figure 7.40 depicts a distributed system comprising networked media devices and a control application. Unlike mixing consoles, where all channel controls reside in the same physical device, channel controls within a distributed system can reside in separate networked media devices. A control application, such as Unos Creator, is used to provide an integrated view of the networked media devices. In the context of this study, deskitems were used as a means to manipulate parameters that are distributed across a network. Permanent absolute peer-to-peer joins between deskitems and networked media devices parameters enable the control application to mirror the values of

[2]This was achieved by a "patching" mechanism that is outside the scope of this investigation.

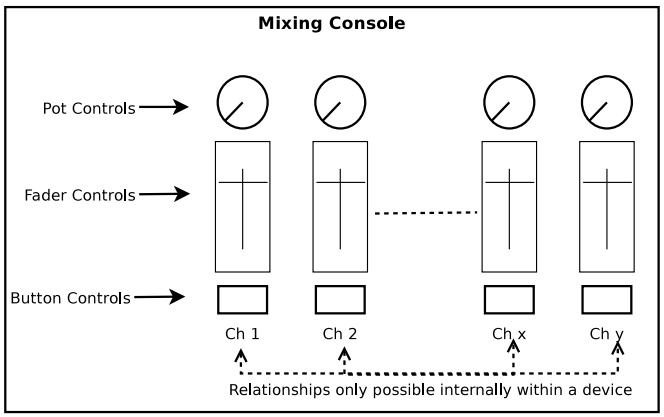


Figure 7.39: Integrated System - Mixing Console

device parameters. Relationships created between the various media device parameters, across the network, are mirrored by the corresponding deskitem parameters.

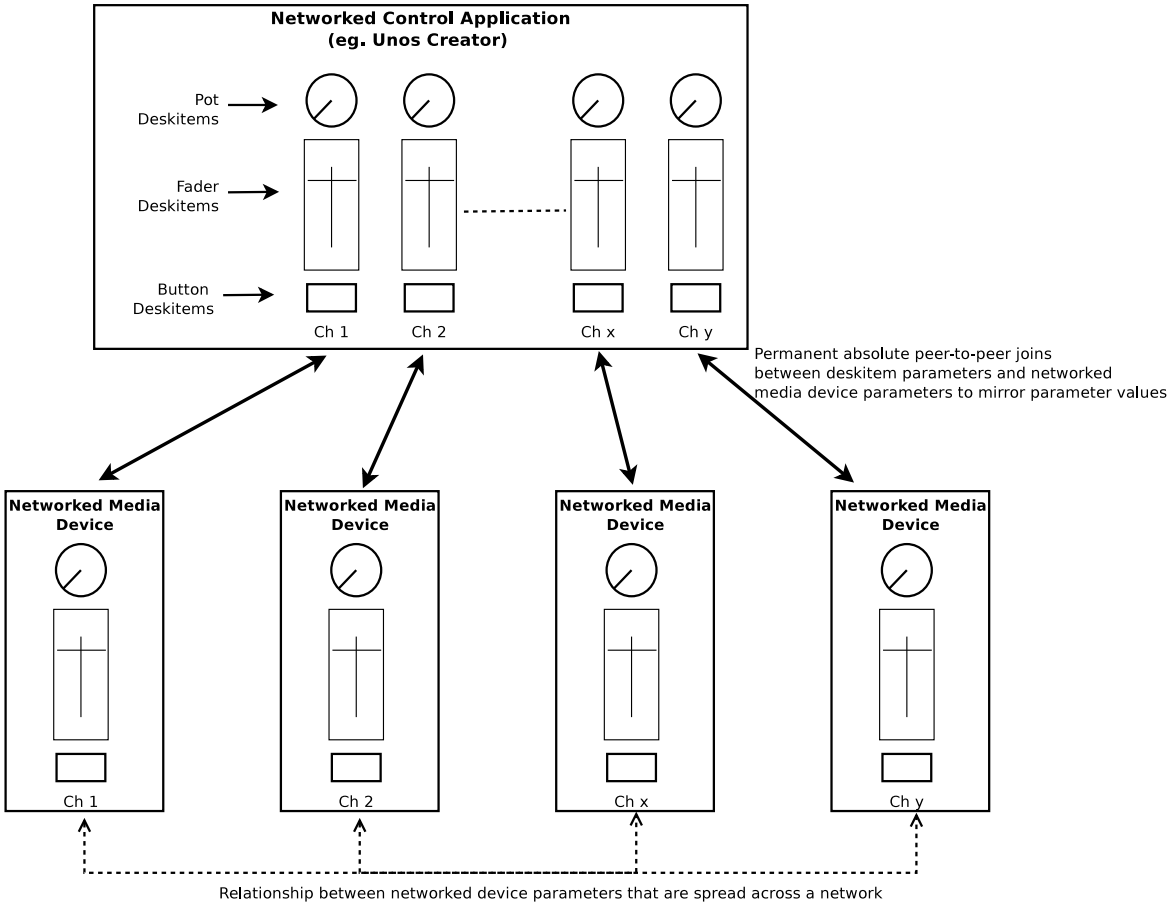


Figure 7.40: Distributed System - Networked Media Devices and Control Application

The remainder of the comparison in this section focuses on the main grouping relationships and points of consideration derived from the review of mixing console capabilities in Chapter 2.

7.6.1 Linking of Controls

Channel controls in mixing consoles can be linked such that values applied to one channel are copied to all other channels. An example of this form of linking is stereo channel pairing. Similar pairing/linking of channels can be achieved in a distributed context by creating absolute peer-to-peer relationships between any two networked parameters.

Once channels are paired in a mixing console, any subsequent groupings of one of the pair partners automatically includes the other pair partner. Similar behaviour has been mimicked in the network grouping mechanisms implemented in this study via the master-slave and peer-to-peer grouping rules described in Section 5.5.1. Here, the creation or breaking of master-slave relationships between a master parameter and slave parameter, where the slave parameter has peers, results in the same action being applied between the master and the slave's peers.

7.6.2 Fader Groups

Standard fader groups in mixing consoles have been mimicked by relative peer-to-peer relationships between networked parameters. The relative level differences between grouped parameters are maintained in response to changes in values of other group members. In addition, master fader groups have been mimicked by relative master-slave relationships between networked parameters. Here, one or more master parameters control one or more slave parameters, while the relative level differences between slave parameters is maintained.

VCA master groups in mixing consoles use a multiplicative approach to combining audio signals. Thus, the resultant audio signal is switched off ($-\infty$) when at least one of the master faders is set to $-\infty$. In contrast, the network grouping mechanisms implemented in this study only support combining parameter values by an additive approach. The behaviour of master groups within mixing consoles, in this regard, is achieved by additional value manipulation using, for example, value modifiers that are outside the scope of this study.

7.6.3 Mute Groups

The behaviour of mute groups in mixing consoles is achievable via absolute master-slave relationships between networked parameters. Whenever the value of the master mute parameter changes, the slave mute parameter takes on the value of the master mute parameter. The mechanisms implemented in this study do not cater for inverse mute groups, where a change in state of one button to the "muted" state results in the toggling of another mute button to the "unmuted" state. However, similar to non-additive combination of values with fader groups, value modifiers can be used to provide similar value manipulation.

7.6.4 Grouping Restrictions

Control grouping in mixing consoles is typically restricted to parameters that have identical types. For example, grouping of input faders only, grouping of output faders only, or grouping of mute buttons. It is not possible to group parameters that have disparate value types. In contrast, the network grouping mechanisms implemented in this study enable grouping of parameters that have disparate value type by using a common value unit, the global unit, that is mapped domain-specific units.

7.7 Future Work

Section 7.2 on page 159 describes the functionality and make up of deskitems, where the behaviour and appearance of the various deskitem types is described using XML. A graphical editing tool that was implemented to ease the deskitem creation process in Unos Creator has been described in Section 7.2.2 on page 177. The use of XML to describe deskitem properties makes it possible for third-party deskitem editing tools to be created. However, there is currently no formalisation of the structure of deskitem XML documents. Future work is required in order to formalise the structure of deskitem XML documents. Such formalisation is possible by the definition of an XML schema definition (XSD) for deskitem XML documents. van der Vlist [2002] describes an XML schema language as a formalisation of constraints that are expressed as rules or a model of structure. These constraints apply to a class of XML documents, deskitem XML documents in this instance.

7.8 Chapter Summary

This chapter has given an overview of the Unos Creator application functionality, particularly focusing on remote control, parameter grouping, and monitoring capabilities. Deskitems are the core drivers of these capabilities. There are descriptions of the properties of the various deskitem types that were created during the course of implementation, as well as a graphical editing tool that was created to simplify the construction of deskitem control surfaces. There are examples of the creation and breaking of master-slave and peer-to-peer relationships even when these parameters are spread across a network. Parameter monitoring in Unos Creator is based on the push mechanism described in Chapter 6. The key elements involved in parameter monitoring in Unos Creator have been highlighted in this chapter. A description of the environment in which parameter grouping and monitoring capabilities were tested has been given. The need for formalisation of the structure of deskitem XML documents, by defining an XML schema definition (XSD), has been identified as an area requiring further work.

The next chapter concludes this thesis by highlighting important aspects of the work carried out in this study.

Chapter 8

Conclusion

The advent of networked multimedia environments has given rise to a number of ancillary requirements, namely device discovery, connection management, control, and monitoring. This work focused on control and monitoring, where a number of proprietary and standards-based protocols have been developed to fulfil various requirements, each with its merits and shortcomings. In order to allow for interoperability across various vendors' devices, standards-based protocols are preferred to proprietary protocols. This work has proposed a standards-based generic approach to parameter grouping and monitoring within high-speed digital multimedia environments. The primary goals were to:

- Prove that by creating relationships between parameters, it is possible to provide decentralised, rule-based control over parameters that are spread across a network.
- Develop a generic mechanism for bulk monitoring of parameters on a network.

The *XFN protocol* was chosen for the implementation of strategies to achieve these two primary goals. Source code for an XFN protocol stack implementation was readily available for modification. At the time of this research, the XFN protocol was undergoing standardisation and most of the implementation considerations from this work have been incorporated into the standard. The standard is entitled "AES64-2012: AES Standard for Audio Applications of Networks - Command, Control, and Connection Management for Integrated Media". The original contributions from this investigation were:

1. The implementation of a mechanism to manage relationships between networked XFN parameters. This mechanism is based on a parameter relationship modelling scheme that was derived from an analysis of parameter grouping and monitoring capabilities of mixing consoles and existing multimedia network technologies.
2. The implementation of a common value unit that is used by all grouped XFN parameters. This common value unit enables the grouping of parameters with disparate value types in a standard manner.
3. The implementation of rules that enable determinism in the behaviour of parameter relationships.
4. Enhancements to an existing one-shot bulk parameter retrieval mechanism of the XFN protocol to support continuous, event-driven, bulk parameter monitoring, where control applications subscribe to receive value change notifications for a given list of parameters.
5. The integration of parameter grouping and monitoring functionality into an existing graphical control application, Unos Creator, by binding graphical components to XFN parameters, where the bound XFN parameters can be involved in relationships with other networked parameters.
6. The implementation of graphical interfaces to demonstrate how parameter groups involving networked parameters can be managed by a control application (Unos Creator).

This chapter will provide an overview of the steps followed to achieve these goals and a critical analysis of the final results.

8.1 Core Requirements for Parameter Grouping and Monitoring in Multimedia Networks

At the onset of this investigation, existing grouping and monitoring capabilities within mixing consoles and multimedia networks were analysed. A core set of four requirements for parameter grouping and monitoring in multimedia networks was derived from this analysis. These requirements were used as the criteria against which the success of a grouping and monitoring solution for networked multimedia environments was measured.

8.1.1 Requirement 1: Utilisation of a Consistent Classification Scheme for Parameter Grouping Relationships

Two main types of relationships between parameters were defined, namely:

- *Master-slave* relationships, where there is a designated master parameter and one or more slave parameters. Changes in value of the master parameter cause changes in values of the slave parameters. However, changes in value of any of the slave parameters do not cause any change in value of the master parameter.
- *Peer-to-peer* relationships, where there is a group comprising two or more parameters. A change in value of any of the parameters causes subsequent changes in values of all other parameters in the group.

These two relationship types were further qualified as either *relative* or *absolute*, where:

- Relative relationships are ones such that an offset (delta) is maintained between the values of the parameters that are part of the relationship. Whenever the value of a parameter, x, changes as a result of another parameter, y, the change in value of x is such that the original offset relative to y remains constant.
- Absolute relationships are ones such that whenever the value of a parameter, x, changes as a result of another parameter, y, the new value of x is always equal to the value of y.

Based on this classification scheme, four types of relationships were defined, namely:

- Relative master-slave relationships.
- Absolute master-slave relationships.
- Relative peer-to-peer relationships.
- Absolute peer-to-peer relationships.

This classification scheme should be comprehensive and covers all relationship/grouping requirements.

8.1.2 Requirement 2: A Standardised Mechanism for Relationships between Parameters on Physically Distant Devices

There have been attempts at grouping networked parameters at the application level. However, implementing grouping at the application level implies that grouping relationships are reinvented by applications. None of these attempts comprehensively incorporates parameter relationships in a protocol. Thus, none have emerged as the *de facto* standard for governing the creation of relationships between parameters that reside in different devices on a network. If grouping of networked parameters is incorporated into a standard protocol, it is possible for all applications to depend on the grouping relationships, and not reinvent grouping within applications. Consequently, the need for a standardised protocol that manages relationships between parameters residing in different devices on a multimedia network was identified.

8.1.3 Requirement 3: A Capability for Relationships between Parameters with Different Underlying Units

The ability of any parameter on the network to influence another parameter regardless of the underlying units was identified as a requirement for parameter grouping in multimedia networks. This enables the creation of customisable control applications that can interpret values sent by parameters without prior knowledge of the associated parameters. This capability was found to be applicable in, for example, *show control systems*, where disparate parameters such as speaker volume and lighting might need to maintain relationships.

8.1.4 Requirement 4: Event-Driven Parameter Monitoring

Event-driven parameter monitoring was identified as a requirement for successful parameter monitoring in multimedia networks. This is due to the need for an efficient mechanism that can handle real-time updates from multiple parameters that constantly change values, such as audio level meters.

8.2 Implementation Strategies to Fulfil Parameter Grouping Requirements

Of the four requirements for grouping and monitoring that were identified, three pertain to grouping relationships that can be created between parameters, namely:

- Requirement 1: The utilisation of a consistent classification scheme for parameter grouping relationships.
- Requirement 2: A standardised mechanism for relationships between parameters on physically distant devices.
- Requirement 3: A capability for relationships between parameters with different underlying units.

Details of the key contributions that address these three requirements are given below.

8.2.1 Management of Relationships between XFN Parameters

In order to model the classification scheme for parameter relationships (requirement 1), each XFN parameter implements three lists containing entries that identify parameters involved in a particular relationship with the parameter hosting the list. Each of the entries in the list contains further information to describe the relative or absolute nature of the relationships, and, if necessary, stores the offset between the parameter values. These lists are:

1. A peer parameter list that shows parameters that have a peer-to-peer relationship with the host parameter.
2. A master parameter list that shows parameters that are designated masters in master-slave relationships involving the host parameter, where the host parameter is a designated slave parameter.
3. A slave parameter list that shows parameters that are designated slaves in master-slave relationships involving the host parameter, where the host parameter is a designated master parameter.

When the value of a parameter changes, the protocol stack firmware scans these lists in order to update any other parameters accordingly. A number of XFN commands were implemented to manage these parameter lists, enabling the creation and breaking of relationships between parameters. Among other data, each list entry contains a device's network identifier, such as an IP address in IP networks, and a device-unique parameter index. This combination of a network identifier and a device-unique parameter index uniquely identifies a single parameter that is hosted by a particular device. This enables relationships to be seamlessly created between parameters that reside in the same device or parameters residing in different devices, or both. Thus, requirement 2 is fulfilled.

8.2.2 Common Value Unit

In order to support the creation of relationships between parameters with different underlying value units (requirement 3), a common value unit known as the *global unit* was defined. All parameters involved in relationships interact in terms of global unit values. The global unit value range is a linear integer value range that has been appropriately apportioned. In some cases, relative parameter relationships result in values that fall outside the defined 0% - 100% global unit value range. The apportionment of the global unit value range takes this into account such that values less than 0% represent *value underflow*, while values greater than 100% represent *value overflow*. The global unit value range is defined in a manner that enables manipulation of values by variable-bit-sized controllers, ranging from 1-bit to 31-bit controllers.

Although the global unit value range is linear, the application-specific parameter values that are mapped to global units are not always linear. For example, the human perception of sound, measured in decibels, is logarithmic. In order to accommodate non-linear value ranges, each application is required to translate global unit values into application-specific units via mapping tables. The use of global unit values for relationships between XFN parameters makes it possible to create relationships between unrelated value types, such as between audio parameters and light intensity parameters.

8.2.3 Rules Defined for Parameter Relationships

During the course of implementation, a number of special cases were encountered. Consequently, rules were defined in order to deal with the special cases. These rules pertain to:

- The hybridisation of master-slave and peer-to-peer relationships.
- The inference of implicit relationships when new relationships are created between parameters that are already involved in other relationships.
- The persistence of certain critical relationships when relationships between parameters are broken, such as the relationship between a controller parameter and a parameter in a remote device.

Details of these rules are given below.

8.2.3.1 Hybridisation of Master-Slave and Peer-to-Peer Relationships

Scenarios involving the coexistence of master-slave and peer-to-peer relationships were analysed. Cases involving a master-slave relationship, where the slave parameter is also involved in one or more peer-to-peer relationships warranted further investigation. Such cases were identified as potential sources of unnecessary value update duplication, particularly when master-slave relationships exist between a master parameter and multiple slave parameters that belong to the same peer-to-peer group. Here, changes in value of the master parameter cause changes in values of each of the slave parameters, while each slave updates its peers. If multiple slaves under the control of a single master, and belonging to the same peer group, are updated in this manner, duplicate updates between the peer parameters exist.

In order to prevent such duplication of value updates for peer parameters that share a master, rules were defined to ensure that all peer parameters share the same set of master parameters. Value changes of a master parameter are then sent directly to each of the slaves, and any of its associated peers. This mechanism also prevents peer parameters from becoming "proxy" parameters that are responsible for relaying value changes of master parameters to any associated peers.

8.2.3.2 Inference of Implicit Relationships

The creation of relationships between parameters that are involved in other relationships was analysed. Two notable scenarios were considered, namely the addition of a peer parameter to an existing peer group and the addition of a master parameter to a slave that belongs to a peer group. These scenarios result in the inference of additional relationships between the new group

member and existing peer parameters. A pattern was observed in the relative/absolute nature of inferred relationships. This pattern resulted in the definition of a rule akin to a logical conjunction, where an unknown relationship is absolute if and only if the known relationships from which the inference is made are both absolute.

Inference rules, coupled with the hybridisation of master-slave and peer-to-peer relationships, enable complex parameter relationships to be created by cascading simple parameter relationships.

8.2.3.3 Persistence of Critical Relationships

Synchronisation of control applications and networked devices is enabled by creating relationships between graphical-component-bound XFN parameters of control applications and those of the remote devices. The use of standard parameter relationships between XFN parameters of control applications and XFN parameters of remote devices enables seamless existence of multiple controllers. Relationships between XFN parameters of control applications and XFN parameters of remote devices were identified as critical, since a controller must always be able to control a remote device. Thus, a rule was defined whereby relationships between XFN parameters of controllers and XFN parameters of remote devices are fixed and can only be relinquished by the respective controllers. These relationships have been classified as "permanent" absolute peer-to-peer relationships.

8.3 Implementation Strategies to Fulfil Parameter Monitoring Requirements

The fourth and last requirement for parameter grouping and monitoring is the use of an event-driven mechanism for parameter monitoring. The capabilities of the XFN protocol were again reviewed in order to develop a mechanism for event-driven parameter monitoring. A mechanism for the bulk retrieval of parameter values from networked devices exists in the XFN protocol. This mechanism is known as the *USG (universal snap groups*). The mechanism makes provision for a one-shot operation and does not support continuous, event-driven, bulk parameter value retrieval. Therefore, the mechanism cannot be used for parameter monitoring in its standard form.

As part of this investigation, the USG mechanism was enhanced to support continuous, event-driven, bulk parameter value retrieval. The enhanced form of the USG mechanism was coined the *push mechanism*. In particular, remote devices publish one or more lists that contain parameters that are available for monitoring, while controllers subscribe to periodically receive notifications upon changes in values of one or more monitored parameters. The layout of each monitored parameter list is dynamically learned; thus, controllers do not need additional meta data in order to monitor parameters of any remote device. The push mechanism prevents unnecessary flooding of the network in the event that subscribed controllers go offline without unsubscribing for change notifications.

In order to make the push mechanism generic, such that any controller can monitor any parameter on the network, global unit values are used. These global unit values are translated into application-specific units via mapping tables.

8.4 An Example Control and Monitoring Application

The implementation strategies that were created in the course of this research were applied to a graphical control application known as *Unos Creator*. Unos Creator reflects the capabilities of the XFN protocol. Unos Creator source code was readily available for modification. Prior to this study, Unos Creator did not implement any parameter grouping or monitoring functionality. Remote parameter control capabilities were unidirectional, where the values of remote parameters could be changed from Unos Creator, while value changes that were applied directly in remote devices were not mirrored in Unos Creator. In addition, a rudimentary graphical fader component that did not support parameter grouping existed. Unos Creator capabilities were enhanced in order to demonstrate the applicability of the parameter grouping and monitoring strategies that were implemented in this study.

Deskitems were created in order to facilitate bi-directional remote control of device parameters using Unos Creator. A deskitem is a skinnable graphical control component that is bound to an XFN parameter. XML elements were used to describe the properties of deskitems. The existing rudimentary graphical fader component was enhanced to support parameter grouping relationships. In addition, a number of new deskitem types were implemented, including background, pot, meter, multi-I/O, and display deskitem types. These simple deskitem types can be combined

to build complex customisable control surfaces that, for example, resemble mixing consoles and media players.

A graphical editing tool that simplifies the construction of control surfaces was created. This tool makes it possible to create, modify, and delete deskitems from control surfaces. In addition, the tool is used to create compressed packages of the deskitem control surfaces. These packages contain pictures that are used for skinning deskitems, mapping tables for conversion from global unit values to application-specific units, and XML files that describe the deskitems in the package. Unos Creator loads these compressed packages from devices in order to display device-specific control surfaces. The use of device-specific deskitem packages, coupled with parameter grouping relationships that are defined at the XFN protocol level, enables the deployment of the parameter grouping and monitoring capabilities across multiple vendor implementations that conform to the XFN standard.

Grouping relationships and the push mechanism are essential for the operation of deskitems. *Permanent absolute peer-to-peer* relationship pairs are created between deskitem parameters and the device parameters that they control. These relationships are permanent, indicating that they are persistent and can only be broken by the associated controller. An absolute peer-to-peer relationship indicates that the deskitem parameter mirrors the value of the parameter on the device that it controls, where an update in one of the parameters results in an update of the other parameter. The use of permanent absolute peer-to-peer relationships ensures a persistent bi-directional remote control channel between Unos Creator and networked devices. However, permanent absolute peer-to-peer relationships are not used for deskitem parameters that are updated using the push mechanism, since parameter monitoring is unidirectional with devices sending updates Unos Creator (and not *vice versa*).

The enhancements made to Unos Creator also demonstrate how relationships between parameters on the same or different devices can be created, managed, and broken using deskitems. This capability is provided at the user level. When a user creates relationships between deskitems in Unos Creator, the relationships cascade to the corresponding parameters within the media devices. Similarly, when a user breaks relationships between deskitems in Unos Creator, the relationships are also broken between parameters within the corresponding media devices.

8.5 Future Work

As a result of this work, there are a number of areas where further work has been identified, namely:

- Cyclic Relationships: At present, master-slave relationships can be chained to create a hierarchy that is composed of recursive master-slave relationships. For example a parameter, *x*, can be a master of a slave parameter, *y*, while *y* is a master of another slave parameter, z, and *z* can have its own slave parameters that also have slave parameters. It is currently possible to create cyclic master-slave relationship chains. An example of a cyclic relationship chain would be a scenario where, *x* is a master of *y*, *y* is master of *z*, while *z* is a master of *x*. Such cyclic relationship chains cause undesirable looping effects upon value changes. Loop detection and prevention approaches should be implemented to alleviate this problem. Graph theory has been used to solve the problem of cyclic dependencies in other contexts. Otten [2013] has explored the use of graph theory techniques to detect and prevent cyclic dependencies in the context of a simulation; it will be important to apply these techniques to a live network.
- Usage of the Global Unit: The global unit was defined and used as a common value unit for all interaction between grouped parameters. Linear and non-linear global-unit-to-other-unit mapping tables were used to translate global units to domain-specific units. Value modifiers (created outside this study) can be used to facilitate the combining of values by non-additive methods, such as multiplication. Although there are some mechanisms to address these issues the usage of global units has not been fully explored. Further work is required to standardise these aspects of global units.
- Handling Updates to Monitored-Parameter Lists: The parameter monitoring mechanism developed relies on devices maintaining static monitored-parameter lists (in the form of USG buffers) containing parameters that are available for monitoring on a device. These lists are learned by controllers when monitoring is initialised. It will be useful to make it possible for controllers to update these monitored-parameter list by adding or removing parameters. Such updates of monitored-parameter lists will require controllers to re-learn the new list layout. In addition to re-learning the new list layout, conflict resolution procedures will be required in cases where, for example, one controller removes parameters that are being actively monitored by other controllers.

- XML Schema: There is no formalisation of the structure of deskitem XML documents. The structure of deskitem XML documents should be formalised by the creation of an XML schema definition (XSD) for deskitem XML documents. An XSD would provide a set of rules or a model of structure for deskitem XML documents, thus easing the development of third-party deskitem creation software.

8.6 Significance of this Research

This work proposed relationships that are essential for the proper functioning of a multimedia network and should, therefore, be incorporated in standard form into a protocol, such that all devices can depend on them. Current implementations of grouping and monitoring within mixing consoles and existing control protocols were used as ideas contributing towards a comprehensive relationship theory. Implementation strategies for this relationship theory within the XFN protocol have been proposed in order to assert the viability of the theory. At the time of this writing, a standard for the XFN protocol had been published by the Audio Engineering Society. The standard is entitled "AES64-2012: AES Standard for Audio Applications of Networks - Command, Control, and Connection Management for Integrated Media". Most of the implementation strategies resulting from this work have been incorporated into this standard.

The grouping mechanisms that have been proposed and implemented enable synchronised control over vast numbers of parameters that are spread across large networks. Relationship information is kept at the parameter level, where each parameter has knowledge of all parameters that it has relationships with. Thus, it is possible to control any parameter in the network via a single decentralised control point.

A graphical user application has been created whereby this type of widespread control can be configured by a user. The graphical user interfaces (GUIs) available to users are customisable. GUI designers can create user interfaces in a manner that is familiar to targeted user groups. For example, a GUI for a sound engineer can be designed in such a way that it resembles a mixing console with a number of controls, such as faders, graph displays, pots, meters, and buttons. A sound engineer can make relationships between parameters that are bound to these controls in a manner that they are familiar with, although across a network.

The grouping and monitoring mechanisms are incorporated into the XFN protocol. All devices that implement the XFN protocol can be expected to respond to grouping and monitoring com-

mands in a consistent manner. This enables seamless, deterministic, synchronisation of parameters within devices that are manufactured by different vendors.

The global unit mechanism enables the creation of relationships between disparate parameters. This opens up the possibility of, for example, lighting parameters being synchronised with audio parameters in, for example, theatre productions. In addition, the ability to map global units to any other units makes it possible for any XFN-protocol-compliant control application to control or monitor any parameter regardless of its underlying data type.

References

1394 Trade Association. *TA Document 1999026: AV/C Digital Interface Command Set General Specification Version 4.0*, July 2001.

AES Technical Committee on Network Audio Systems. Technology trends in audio engineering. *Journal of the Audio Engineering Society*, 60(1/2):101–103, 2012. URL `http://www.aes.org/journal/online/JAES_V60/1_2/`.

ALC NetworX GmbH. *Draft 1: RAVENNA - Operating Principles*, 2011.

Allen & Heath Limited. *Mixing Live with VCAs*. URL `http://www.allen-heath.com/DL/vca_book.pdf`.

Allen & Heath Limited. *ML3000 User Guide AP4512 Issue 2*, 2003a. URL `http://www.allen-heath.com/DL/ml3000ug_ap4512_2.pdf`.

Allen & Heath Limited. *ML4000 User Guide AP4314 Issue 4*, 2003b. URL `http://www.allen-heath.com/DL/ml4000ug_ap4314_4.pdf`.

Allen & Heath Limited. *ML5000 User Guide AP3736 Issue 5*, 2003c. URL `http://www.allen-heath.com/DL/ml5000ug_ap3736_5.pdf`.

Don Anderson. *FireWire System Architecture*. Mindshare Inc., New Jersey, 2nd edition, 1999.

Audinate. *White Paper: Dante*, 2009.

Audio Engineering Society. *AES64-2012: AES Standard for Audio Applications of Networks - Command, Control, and Connection Management for Integrated Media*. Audio Engineering Society, Inc., 2012.

Jeff Berryman. The open control architecture. *J. Audio Eng. Soc*, 61(4):185–200, 2013. URL `http://www.aes.org/e-lib/browse.cfm?elib=16704`.

Bradley Klinkradt. *Enhanced USG Mechanism*. Universal Media Access Networks GmbH, 2010.

J.D. Case, M. Fedor, M.L. Schoffstall, and J. Davin. Simple Network Management Protocol (SNMP). RFC 1157, May 1990. URL `http://www.ietf.org/rfc/rfc1157.txt`.

Nyasha Chigwamba, Richard Foss, Robby Gurdan, and Bradley Klinkradt. Parameter Relationships in High-Speed Audio Networks. *Journal of the Audio Engineering Society*, 60(3): 132–146, 2012. URL `http://www.aes.org/e-lib/browse.cfm?elib=16211`.

D&R Electronica Weesp b.v. *PowerVCA Manual version 2.18 (beta version)*, 2010a. URL `http://www.d-r.nl/dnrsite/pdfmanuals/Power-2.18-Manual.PDF`.

D&R Electronica Weesp b.v. *Sirius Digital On Air Console Manual V1.0.0.8*, 2010b. URL `http://www.d-r.nl/dnrsite/pdfmanuals/Sirius-Manual-V1.0.0.8.PDF`.

D&R Electronica Weesp b.v. *SIRIUS - Modular digital ON-AIR broadcast mixer*, 2010c. URL `http://www.d-r.nl/SIRIUS/SIRIUS.htm`.

Andrew Eales. *The Derivation of a Standard Device Model and Associative Memory Control Protocol Based on a Study of Current Audio Control Protocols*. PhD thesis, Rhodes University, 2012. Pending examination.

Entertainment Services and Technology Association. *ACN Architecture*, October 2005.

Entertainment Services and Technology Association. *ACN Base Behavior Set. As extended by the addition of behaviorsets that are identified as "Core Modules" (urn:uuid:71576eac-e94a-11dc-b664-0017316c497d)*, 2009a.

Entertainment Services and Technology Association. *DRAFT, BSR E1.17-20xx, Architecture for Control Networks Device Description Language (DDL)*, 2009b.

Entertainment Services and Technology Association. *DRAFT, BSR E1.17-200x, Architecture for Control Networks - Device Management Protocol (DMP)*, 2009c.

European Broadcasting Union. *Tech 3326: Audio Contribution over IP - Requirements for Interoperability*, 2008.

Philip James Foulkes. *An Investigation into the Control of Audio Streaming Across Networks having Diverse Quality of Service Mechanisms*. PhD thesis, Rhodes University, 2011.

Kevin Gross. *White Paper: Q-LAN*, 2009.

John Huntington. *Control Systems for Live Entertainment*. Focal Press, 2007.

IEEE Std. 1394a. *Standard for a High Performance Serial Bus - Amendment 1*, 2000.

IEEE Std. 1394b. *Standard for a High Performance Serial Bus - Amendment 2*, 2002.

International Electrotechnical Commission. *IEC 61883-1: Consumer Audio/Video Equipment - Digital Interface - Part 1: General*. 2003.

International Electrotechnical Commission. *IEC 62379 Common control interface, Part 1: General*, June 2007.

International Electrotechnical Commission. *IEC 62379 Common control interface, Part 2: Audio*, June 2008.

Roey Izhaki. *Mixing Audio (concepts, practices and tools)*. Focal Press, 2nd edition edition, 2013.

Frederick John Otten. *Network Simulation for Professional Audio Networks*. PhD thesis, Rhodes University, 2013. Pending examination.

Francis Rumsey and Tim McCormick. *Sound and Recording*. Focal Press, 6th edition, 2009.

TC Applied Technologies, Ltd. *Digital Interface Communications Engine*, 2007.

The Audio/Video Bridging Task Group. *IEEE 801.2*, 2012. URL `http://www.ieee802.org/1/pages/avbridges.html`.

K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms*. John Wiley & Sons, 1992.

Universal Media Access Networks GmbH. *Unos Creator User Manual*, 2010.

Eric van der Vlist. *XML Schema*. O'Reilly Media, Inc., 2002.

Matthew Wright. *Open Sound Control 1.0 Specification*, 2002. URL `http://opensoundcontrol.org/spec-1_0`.

Matthew Wright and Adrian Freed. Open sound control: A new protocol for communicating with sound synthesizers. In *International Computer Music Conference*, pages 101–104, Thessaloniki, Hellas, 1997. International Computer Music Association. URL `http://cnmat.berkeley.edu/publications/open_sound_control_new_protocol_communicating_sound_synthesizers`.

Yamaha Corporation. *01V96 Version 2 Owner's Manual*, 2004.

Glossary

ACN	Architecture for Control Networks.
AES	Audio Engineering Society.
API	Application Programming Interface.
Audio clipping	A type of distortion that limits an audio signal once it exceeds a threshold.
AVB	Audio Video Bridging.
AV/C	Audio Video Control.
Channel	An audio signal/track.
Compression	The reduction in volume of loud sounds, or amplification of quiet sounds, by narrowing ("compressing") an audio signal's dynamic range. Dynamic range is the ratio between the largest and smallest possible values of an audio signal.
Control surface	The human interaction device or computer screen that enables a user to control a device.
Controller	A device, typically a computer workstation, that can be used to configure other devices on a network.
Cut	Switching the signal of specific tracks on or off on a mixing console.
dBFS	Decibels relative to full scale. Measures decibel amplitude levels in digital systems which have a defined maximum available peak level, where 0 dBFS is assigned to the maximum possible digital level.

dB(SPL)	Decibel (sound pressure level). This is the measurement of sound in air, relative to 20 micro pascals.
Deskitem	A graphical control component that is bound to an XFN parameter.
Device	A node that has multimedia capabilities, such as an amplifier or video transmitter.
DHCP	Dynamic Host Configuration Protocol.
DSP	Digital Signal Processor.
EQ	Equaliser.
Fader	A vertical slider control that is typically used to control the volume of an audio channel.
FCP	Function Control Protocol.
FOH	Front of House. The area of a performance venue that is open to the public.
Gain	A measure of the ability of a circuit, typically an amplifier, to increase the power of a signal from the input to the output.
Group	A collection of parameters sharing a relationship.
GUI	Graphical User Interface.
IEC	International Electrotechnical Commission.
IP	Internet Protocol.
Join	In the context of the XFN protocol, a join refers to the creation of a relationship between two parameters.
LAN	Local Area Network.
LCD	Liquid Crystal Display.
LED	Light Emitting Diode.
LSB	Least significant bit.

MIDI	Musical Instrument Digital Interface.
Mixing	The process by which a number of input audio signals are combined to produce a combined output signal.
MSB	Most significant bit.
Nominal level	The typical signal level at which a piece of equipment operates.
OCA	Open Control Architecture.
OID	Object Identifier.
OSC	Open Sound Control.
Pan	Audio panning is the spreading of audio signals, typically stereo, between the right and left channels such that audio may appear to be more concentrated on either side.
Pot	A rotary value input control.
PPM	Peak Programme Meter.
QoS	Quality of Service.
Show control system	A system that connects two or more entertainment control systems, resulting in a global system that is composed of two or more sub-systems.
Skinning	In the context of deskitems, skinning is the changing of the appearance of a deskitem using one or more pictures.
SMPTE timecode	A set of cooperating standards that label individual frames of video with a time code defined by the Society of Motion Picture and Television Engineers in the SMPTE 12M specification.
SNMP	Simple Network Management Protocol.
Solo	A function of a mixing console that enables a sound engineer to listen to specific tracks in isolation.
Stereo	A pair of audio channels that give the impression that sound is coming from two directions.

String orchestra	An orchestra comprising mainly instruments from the string family, including the violin, the viola, the cello, the double bass, the harp, the piano.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
UML	Unified Modelling Language.
Unjoin	In the context of the XFN protocol, unjoin refers to the breaking of a relationship between two parameters.
USG	Universal Snap Groups.
Value overflow	In the context of the XFN protocol, this is the range of values that are greater than the 100% global unit value mark.
Value underflow	In the context of the XFN protocol, this is the range of values that are less than the 0% global unit value mark.
VCA	Voltage Controlled Amplifier.
WAN	Wide Area Network.
XFN	Short for "Cross-Fire Network" protocol. The project code name that was used to refer to the AES-64 protocol prior to standardisation. This is the name used in this thesis.
XLR	An electrical connector commonly used in audio equipment.
XML	Extensible Markup Language.
XSD	XML Schema Definition.

Appendix A

Mixing Consoles

This appendix contains additional information regarding mixing console implementations, namely:

- A brief description of how audio groups work. It should be noted that audio groups are presented for informational purposes only, since they fall outside the scope of this study.
- Detailed descriptions of the human user interfaces for manufacturer-specific mixing console implementations. These descriptions focusing mainly on the procedures for setting up and clearing various control groups described in Chapter 2.

A.1 Audio Grouping

A.1.1 Master Groups

Mixing consoles that make use of master groups typically have a dedicated strip of controls for each group bus on the master section [Izhaki, 2013]. Each of these strips contains a number of controls, such as a fader, a solo button, or a mix-assignment switch. Each group also has an insertion point that allows for processing of the group signal. An unprocessed group signal is typically sent out to an external processing device via an *insert send socket*, while the processed signal is received via the mixing console's *insert receive socket*.

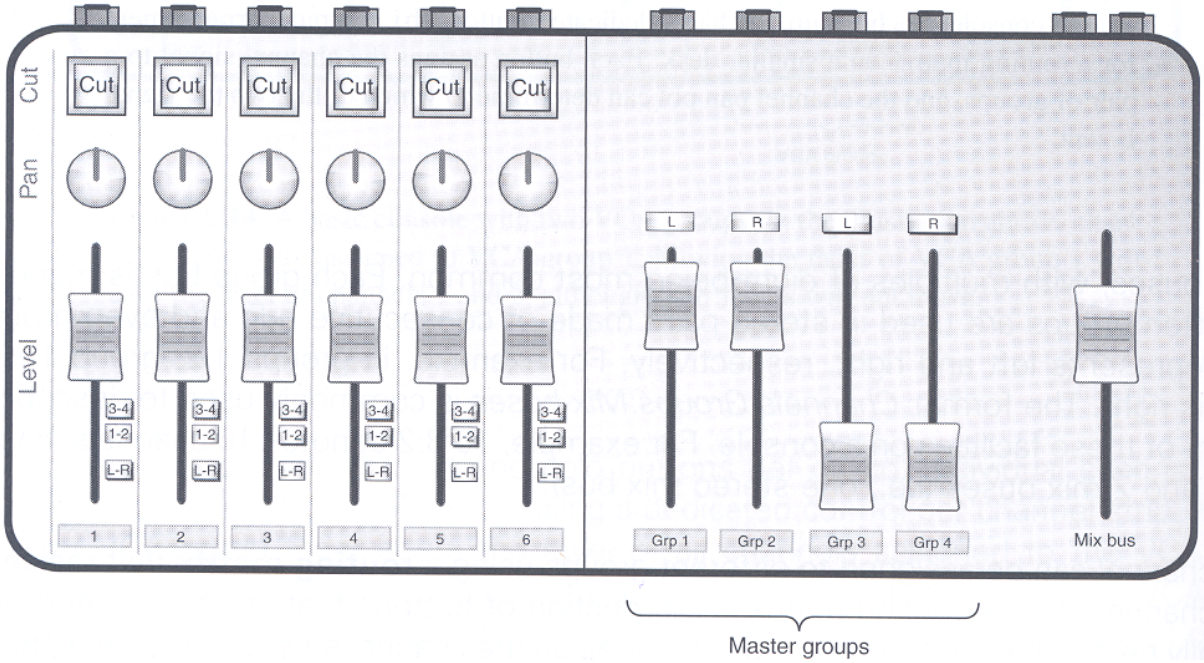


Figure A.1: A Basic Mixing Console with Audio Groups.

Figure A.1 shows a basic layout of a mixing console with a channel section (left hand side) containing six channels and master section (right hand side) with four groups. With reference to the channel section shown in the figure, there is a routing matrix next to each channel. The routing matrix comprises three buttons with the following functions:

1. The “1-2” button - used to route a particular channel to groups 1 and 2.
2. The “3-4” button - used to route a particular channel to groups 3 and 4.
3. The “L-R” button - used to route a particular channel to the stereo mix bus.

With reference to the master section shown in Figure A.1, each group contains a fader and a button that is labelled either “L” or “R”. The fader controls the overall level of the group signal. The button labelled “L” assigns a group signal to the left of the stereo mix bus, while the button labelled “R” assigns a group signal to the right of the stereo mix bus.

Figure A.2 shows a signal flow diagram which indicates the path that is taken by the signal within a mixing console that makes use of master groups as indicated in the layout in Figure A.1.

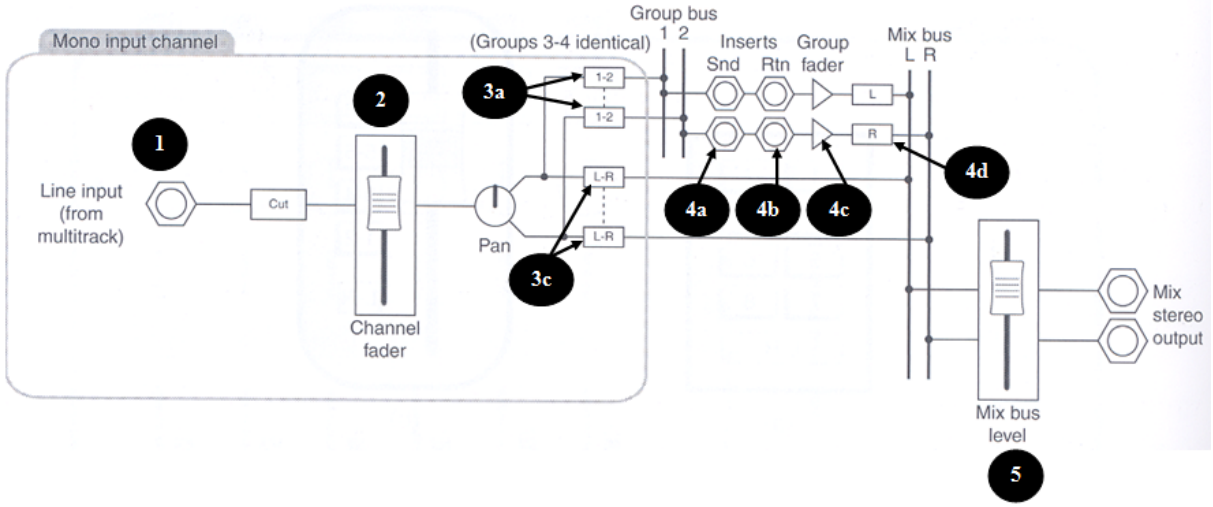


Figure A.2: Signal Flow Diagram for Audio Groups.

Without getting into much technical detail, the following main events occur (also annotated in Figure A.2):

1. Input signal is received from a multitrack recorder.
2. The level of the input signal can be altered by the channel's fader.
3. The channel can be routed to any of the following:
   (a) Groups 1 and 2.
   (b) Groups 3 and 4 (not shown in the signal flow diagram but identical to groups 1 and 2).
   (c) Directly to the mix bus.
4. Channels that are routed to a particular group bus comprise the group signal. The group bus signal can be handled as follows:
   (a) The unprocessed group bus signal is sent out to an external processing device via an insert send socket for the group bus.
   (b) The processed group bus signal is received from the external processing device via an insert receive socket for the group bus.
   (c) The level of the processed group signal can be altered by the group fader control.

(d) The processed group signal can be routed to the mix bus.

5. The level of the output signal is altered by the mix bus level fader before it leaves the mixing console via the mix stereo output.

A.1.2 In-Line Groups

Similar to mixing consoles with master groups, as described in Section A.1.1, mixing consoles with in-line groups have a routing matrix per channel [Izhaki, 2013]. However, mixing consoles with in-line groups do not have any master group strips. Instead, each of the channel section strips has a mutually exclusive dual function of either hosting a group or handling the corresponding track from the multitrack recorder. For example channel 1 is capable of hosting group 1, while channel 2 could host group 2, *et cetera*. Only one of these two functions can be in use at any given point. When a channel is hosting a group, the group signal is affected by all the components that belong to the given channel such as pan pot, fader, and insertion points.

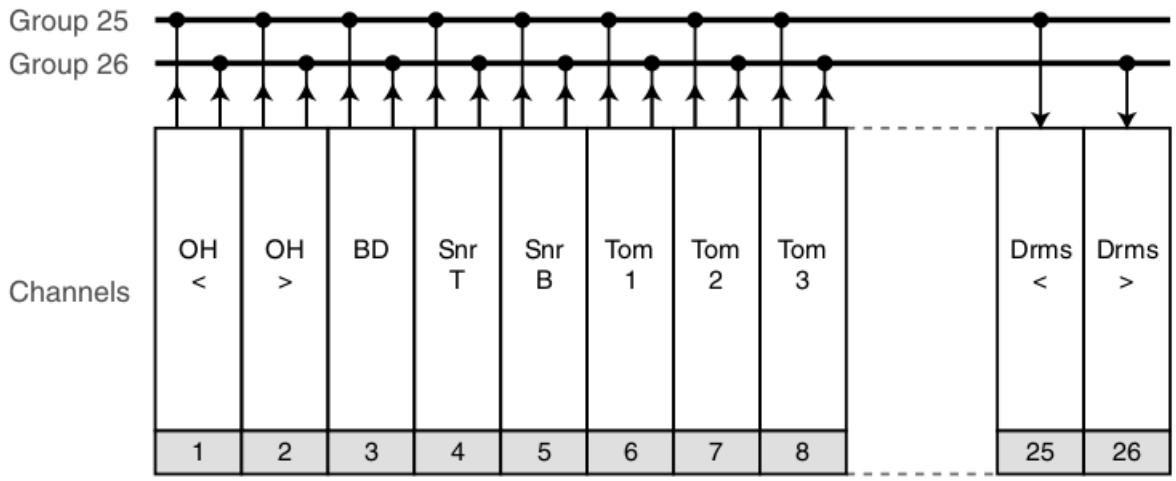


Figure A.3: Basic Schematic of Audio Groups

A group button typically exists on the input section of each channel. This button acts as a selector switch that determines if the signal is to be received from the multitrack recorder or the associated group signal. Figure A.3 shows a basic schematic of in-line groups as adapted from Izhaki [2013]. The figure shows channels 1-8 routed to groups 25-26. It would then be possible

to collectively process channels 1-8 by processing channels 25-26. The processed signals can then be routed to the mix bus by simply routing channels 25-26 to the mix bus.

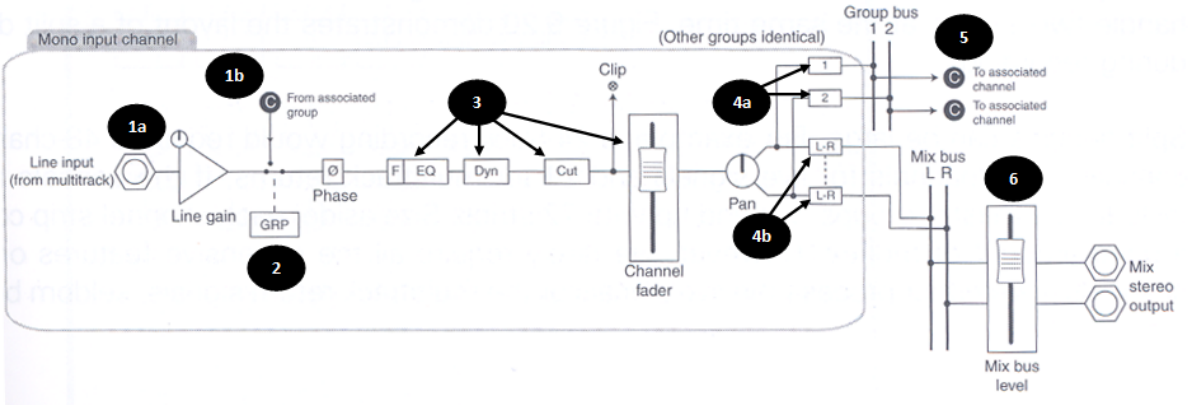


Figure A.4: Signal Flow Diagram for In-line Groups.

The general sequence of events for in-line groups is shown in the signal flow diagram in Figure A.4. In particular, the following events occur (also annotated in the figure):

1. Each channel receives its input signal from one of two sources, namely:
   (a) From a multitrack recorder via the corresponding channel's input.
   (b) From the channel's associated group. For example, group 1 is associated with channel 1, group 2 with channel 2, and so on.
2. As mentioned above, a group button is used to determine which of the inputs (1(a) or 1(b)) is used by the channel.
3. A number of controls are available along the signal path. These controls may include insertion points and faders. Regardless of the source of the input signal, these controls process or control the input signal which could either be the group signal or input signal from a multitrack recorder.
4. Similar to mixing consoles with master groups, a routing matrix exists for each channel on a mixing console with in-line groups. The routing matrix allows for input signal to be routed as follows:

(a) To any of the group buses. Recall that each channel on the mixing consoles is capable of hosting a group so there is a group bus per channel.

(b) Directly to the mix stereo bus from where the output signal is derived.

5. Signals that are routed to the group buses are then forwarded to the corresponding channel as shown in the step 1b of Figure A.4.
6. Signals that are routed to the mix stereo bus can have their levels altered by a mix bus level fader before the signal goes out to the mix stereo outputs.

A.2 Procedures for Configuring Control Grouping Relationships in Manufacturer Specific Implementations of Mixing Consoles

A.2.1 Yamaha Corporation

A.2.1.1 01V96 Version 2 Digital Mixing Console

Figure A.5 shows a diagram of the 01V96 mixing console. The figure shows four annotated sections that are relevant to control grouping, namely:

1. Display access section
2. Display section
3. Data entry section
4. A subset of the channel strip section showing the first two channels

Figure A.6 shows the display access section of the 01V96 mixing console. The display access section shows a number of buttons that allow users to view various capabilities on the display section. The "PAIR/GROUP" button of the display access section is of relevance to this investigation, since it displays a "pair/group" page [Yamaha Corporation, 2004]. The "pair/group" page allows users to create or cancel channel pairs and to group multiple channel faders or "ON" buttons.

Figure A.5: 01V96 Version 2 Digital Mixing Console

Figure A.7 shows the display section of the 01V96 mixing console. The display section has an LCD for displaying various settings. On the right of the LCD there is a pair of stereo meters that display the final output level of the stereo bus. Below the LCD, there are a number of buttons that are used to navigate multi-page screens.

Figure A.8 shows the data entry section of the 01V96 mixing console. The data entry section comprises the following:

- "INC" and "DEC" buttons to increase or decrease a parameter value by one, respectively.
- A parameter wheel that adjusts a parameter value, where turning it clockwise increases the

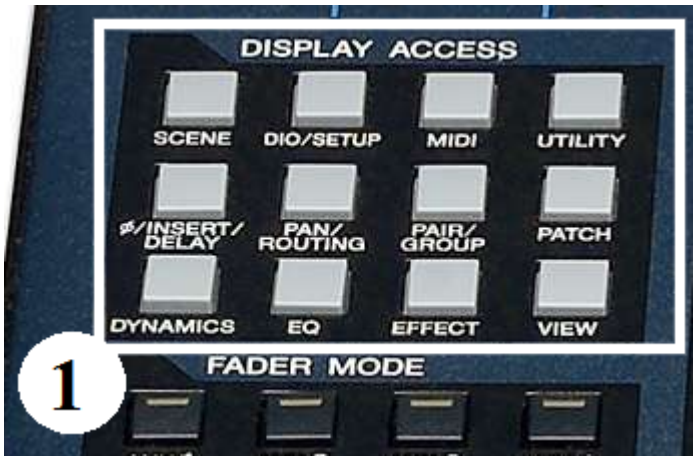


Figure A.6: 01V96 Mixing Console - Display Access Section

Figure A.7: 01V96 Mixing Console - Display Section

value and turning it counterclockwise decreases the value.

- Up, Down, Left, Right cursor buttons to move the cursor around display pages.
- An "ENTER" button to activate selected fields on the display as well as confirm edited parameter values.

Figure A.9 shows the first two channels of the channel strip section of the 01V96 mixing console. Each channel of the channel strip comprises the following:

- A "SEL" button that allows the channel to be selected. This button is used when creat-

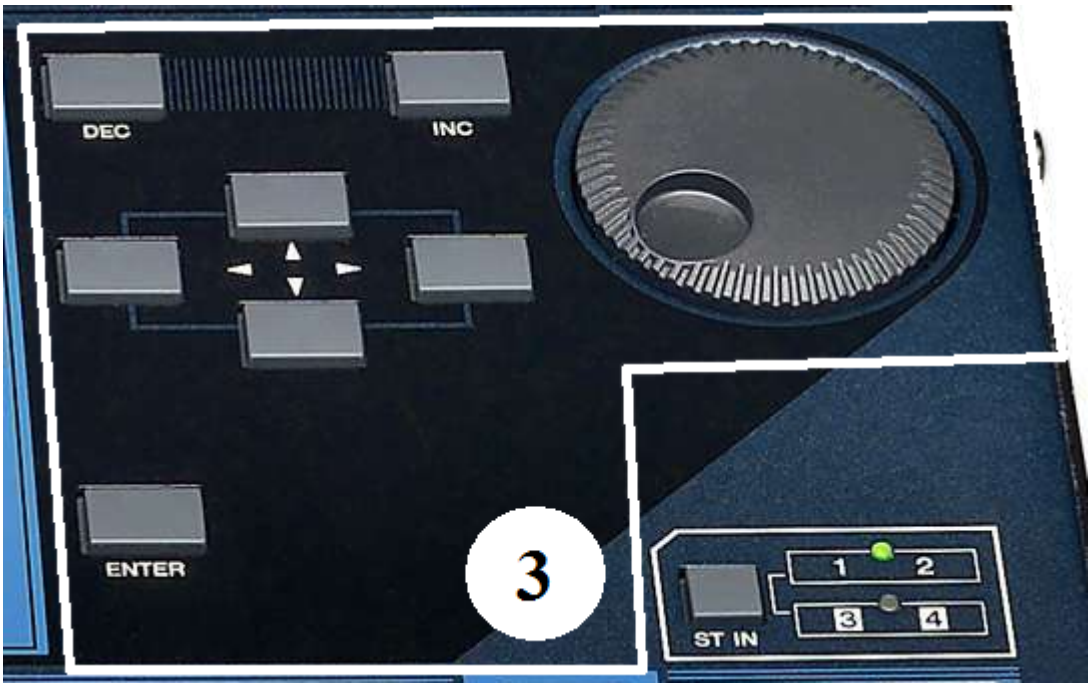


Figure A.8: 01V96 Mixing Console - Data Entry Section

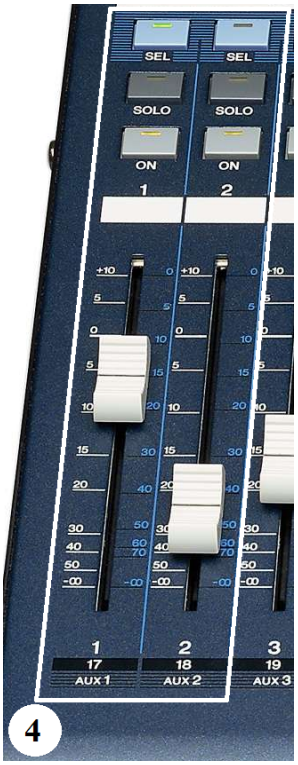


Figure A.9: 01V96 Mixing Console - Partial Channel Strip Section

ing or cancelling channel pairs, and when adding channels to (or removing them from) fader, mute, EQ, and compressor groups. Descriptions of the procedures for setting up and breaking these groups are given below. The button has an indicator that illuminates when the channel is selected.

- A "SOLO" button to solo the channel. The button has an indicator that illuminates when the channel is soloed.
- An "ON" button to turn the channel on or off. The button has an indicator that illuminates when the channel is on.
- A channel fader to adjust the channel level.

It is worth mentioning that there are 16 channels in the channel strip section. The channel strip is shared across three layers with regards to grouping relationships, namely layer 1, layer 2, and layer 3. Layer 1 is associated with input channels 1 - 16, layer 2 with input channels 17 - 32, and layer 3 with auxiliary (aux) sends 1 - 8 and buses 1 - 8. For example, the first channel in the partial channel strip shown in Figure A.9, can be configured to control channel 1, channel 17, or aux send 1, depending on the mode of operation that is selected. Note that aux sends and buses collectively comprise the output channels of the mixing console. We will not look further at the technicalities of this layering although we acknowledge its existence.

Descriptions of the procedures followed when creating various relationships, using the control shown above, are given below.

A.2.1.1.1 Procedures for Input/Output Channel Pairing

There are two procedures that can be followed when pairing channels, either by using "SEL" buttons of the channel strip or via the pair/group page. These procedures are described below.

Channel Pairing Using "SEL" Buttons

To create a pair relationship between two channels, the following steps are required:

1. The "SEL" button of one of the channels to be paired is pressed and held down.

2. While the “SEL” button in (1) above is held down, the “SEL” button of the adjacent channel is pressed.
3. A “Channel Pairing” confirmation window is then shown in the display.
   Figure A.10 shows an illustration of this window. The window shows four options with the following outcomes:

   (a) CANCEL
   The pairing process is aborted.

   (b) CH x –> y
   The channels are paired, with the odd channel’s parameter values being copied to the even channel.

   (c) CH y –> x
   The channels are paired, with the even channel’s parameter values being copied to the odd channel.

   (d) RESET BOTH
   The channels are paired with parameter values for both channels being reset to default values.

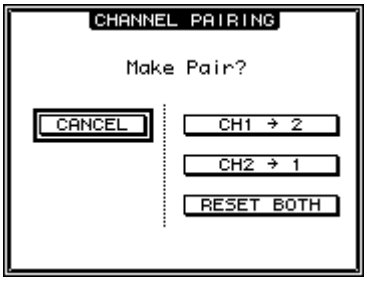


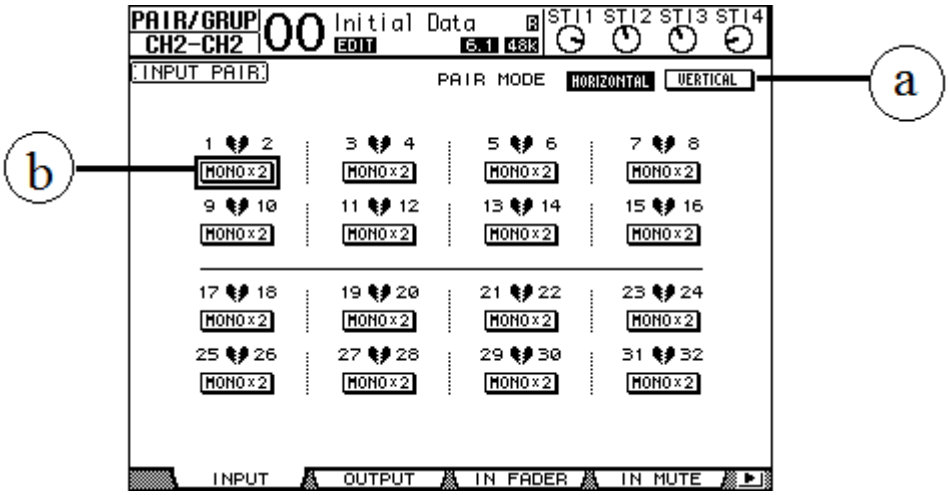
Figure A.10: 01V96 Mixing Console - Channel Pairing Window

4. The cursor is moved to the desired option and the “ENTER” button is pressed to confirm the pair.

To cancel a pair relationship between two channels, the following steps are required:

1. The first “SEL” button of the paired channels is pressed and held down.
2. The second “SEL” button is pressed while the button in (1) is held down in order to cancel the pair.

Pairing Input Channels Using the Display



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner's Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.11: 01V96 Mixing Console - "Pair/Group | Input" Page

To create or cancel pair relationships between input channels by using the display, the following steps are required:

1. The "PAIR/GROUP" button of the display access section is pressed repeatedly until the "Pair/Group | Input" page is displayed as shown in Figure A.11. As shown in the figure, the page has two main fields, namely "pair mode" (labelled (a) in the figure) and "stereo/monox2" buttons (labelled (b) in the figure), where:

   (a) "Pair mode" determines how the channels are paired.

   (b) "Stereo/Monox2" buttons turn pairs on or off.

2. The cursor is moved to the pair mode field, where either the horizontal or vertical modes are selected. The functions of each of the modes are as follows:

   (a) Horizontal mode
   In this mode, the adjacent odd-even channels are paired, such as input channels 1 and 2 or input channels 3 and 4.

   (b) Vertical mode
   Recall that the channel strip is shared across three layers, where layer 1 controls

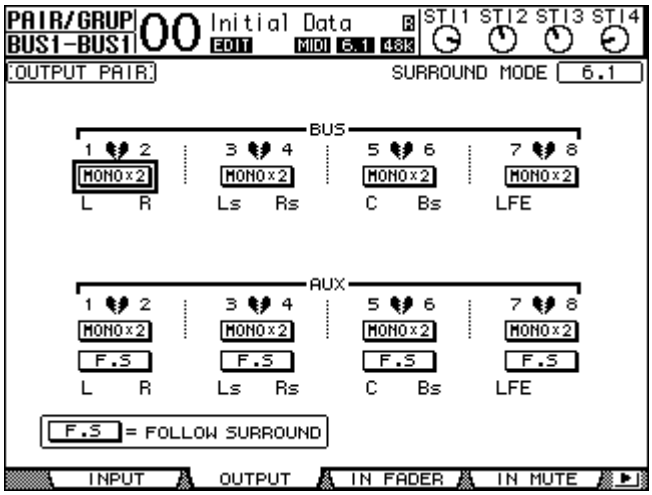
input channels 1 - 16, while layer 2 controls input channels 17 - 32. In the vertical mode, counterpart channels on these two layers that share the same physical fader are paired. For example, input channels 1 and 17 or input channels 2 and 18. This mode is mainly useful when it is desirable to use one physical fader to control both stereo channels.

3. To create a pair relationship, the cursor is moved to a channel's "monox2" button and the "ENTER" button is pressed. Although not shown in Figure A.11, upon enabling the pair relationship, the "monox2" button becomes a "stereo" button.
4. To cancel pair relationships on the "Pair/Group | Input" page, the cursor is moved to the channel's "stereo" button and the "ENTER" button is pressed. Upon cancellation of the pair relationship, the "stereo" button becomes a "monox2" button.

Pairing Aux Sends or Buses Using the Display

Recall that the channel strip is shared across three layers, where layer 3 is associated with aux sends 1 - 8 and buses 1 - 8. Although similar to the procedure indicated above for input channels, a separate page exists to pair either aux sends or buses, where the following steps are required:

1. The "PAIR/GROUP" button of the display access section is pressed repeatedly until the "Pair/Group | Output" page is displayed as shown in Figure A.12. On this page, we focus only on the "stereo/monox2" buttons, since they turn bus or aux send pairs on or off.
2. To create a pair relationship between two buses or two aux sends, the cursor is moved to the "monox2" button for the desired bus or aux send and the "ENTER" button is pressed. Although not shown in Figure A.12, upon enabling the pair relationship, the "monox2" button becomes a "stereo" button.
3. To cancel pair relationships on the "Pair/Group | Output" page, the cursor is moved to the bus' or aux send's "stereo" button and the "ENTER" button is pressed. Upon cancellation of the pair relationship, the "stereo" button becomes a "monox2" button.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner's Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.12: 01V96 Mixing Console - "Pair/Group | Output" Page

A.2.1.1.2 Procedures for Setting Up Fader Groups

A.2.1.1.2.1 Standard Fader Groups

There are two pages available for setting up fader groups, one for groups associated with input channels and the other for groups associated with output channels, where:

- The "Pair/Group | In Fader" page is used to set fader groups (A - H) that control input channels 1 - 32 and stereo in channels 1 - 4.
- The "Pair/Group | Out Fader" page is used to set fader groups (Q - T) that control bus outs 1 - 8, aux outs 1 -8, and stereo out.

The procedures for setting up groups are similar for both input and output channel fader groups. We give an indication of the steps that are required to set up input channel groups, bearing in mind a similar process is required for the output channel groups, although on a separate page.

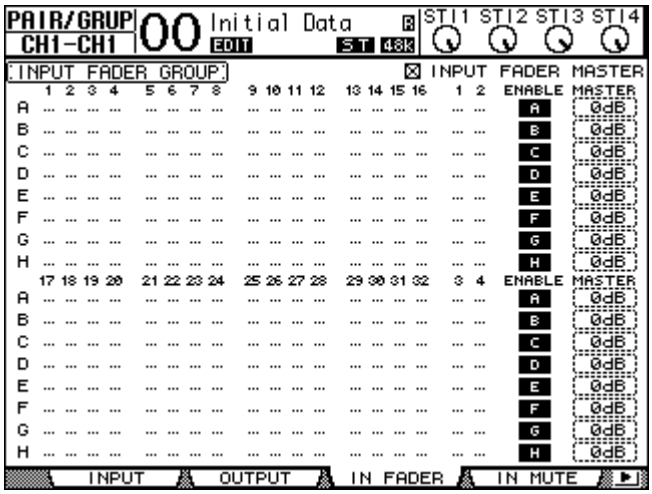
1. The "PAIR/GROUP" button of the display access section is pressed repeatedly until the "Pair/Group | In Fader" page is displayed. Figure A.13 shows an illustration of this page, where there are:

(a) Rows labelled A - H, indicating the groups available.

(b) Columns labelled 1 - 16 and 17 - 32, corresponding to input channels 1 - 32.

(c) Columns labelled 1 - 2 and 3 - 4, corresponding to stereo in channels 1 - 4.

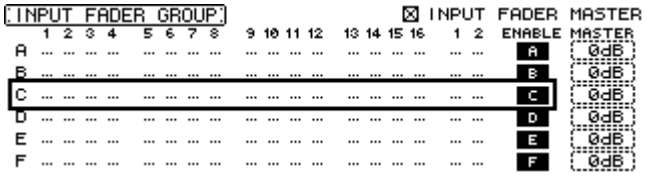
(d) Rows with “enable” buttons labelled A - H that enable and disable the corresponding groups.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner’s Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.13: 01V96 Mixing Console - “Pair/Group | In Fader” Page

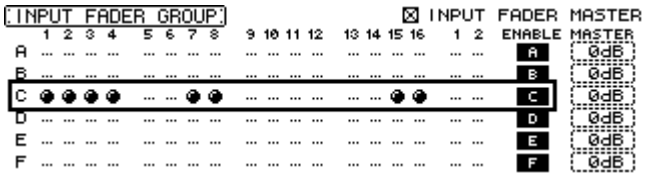
2. The up or down buttons are pressed to select a group. Figure A.14 shows group C selected for input channels 1 - 16 and stereo in 1 - 2.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner’s Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.14: 01V96 Mixing Console - “Pair/Group | In Fader” Page with Group Selected

3. The “SEL” buttons for the channels to be added to the group are pressed and the selected channels are marked with solid circles. The relative levels of the grouped channels are determined by the positions of the faders when the channels are added to the group. Figure A.15 shows input channels 1 - 4, 7, 8, 15, and 16 added to group C.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner's Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.15: 01V96 Mixing Console - "Pair/Group | In Fader" Page with Grouped Channels

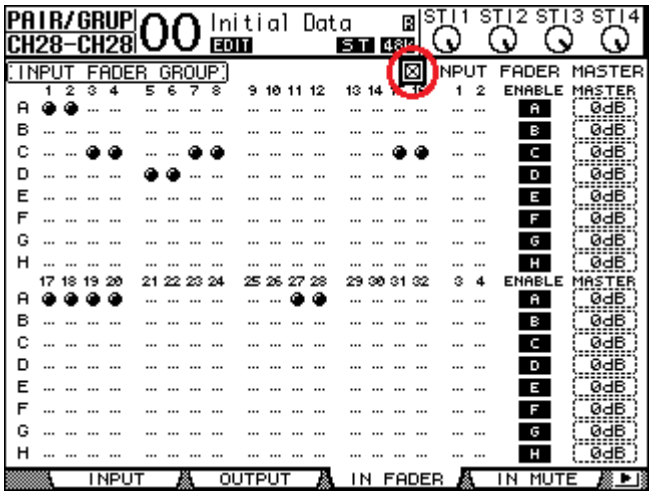
4. When a channel from a pair is added to a group, the pair partner is also added to the group.
5. Groups are turned on or off when the "ENTER" button is pressed over the corresponding "enable" button of the group.
6. The relative level balance of the grouped channels can be adjusted while the "enable" button of the group is off.

A.2.1.1.2.2 Master Fader Groups

The Fader Group Master function assumes that channel faders have already been grouped as described in the procedures for standard fader groups in the previous section. The steps followed when setting up the Fader Group Master function are described below, in the context of input channels, after the channels have been assigned to groups. Again, we note that a similar procedure is applicable for output channels, although on a different page.

1. The cursor buttons are used to select the "INPUT FADER MASTER" check box on the "Pair/Group | In Fader" page as shown in Figure A.16. For output channels, the equivalent is the "OUTPUT FADER MASTER" check box on the "Pair/Group | Out Fader" page. The "ENTER" button is pressed to turn on the Fader Group Master function.
2. While the Fader Master check box is checked, channel levels of the fader groups can be set by rotating the parameter wheel with the cursor over the master column entry of a given group. Repeatedly pressing the "ENTER" button in this state turns the fader group on or off.

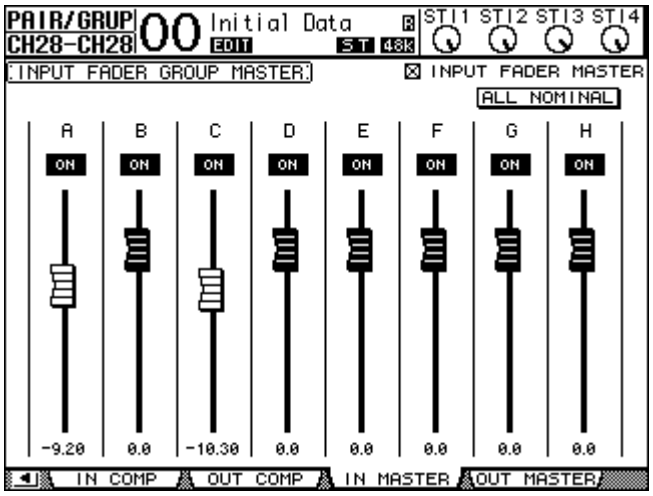
Alternatively, the Fader Group Master settings are modified in either the "Pair/Group | In Master" or "Pair/Group | Out Master" pages. We now describe this procedure in the context of the input channels, although the same is possible for output channels on a different page.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner's Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.16: 01V96 Mixing Console - "Pair/Group | In Fader" Page: Input Fader Master Selected

1. The "PAIR/GROUP" button of the display access section is pressed repeatedly until the "Pair/Group | In Master" page is displayed as shown in Figure A.17.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner's Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.17: 01V96 Mixing Console - "Pair/Group | In Master" Page

2. The cursor buttons are used to select parameters, then the parameter wheel, "INC/DEC" buttons, or "ENTER" button are used to set the parameters, where the following parameters can be set:

   (a) INPUT FADER MASTER

Master levels for the fader groups can only be set while this check box is checked. The resultant channel level is the sum of the channel fader level and the group master level.

(b) ALL NOMINAL
This button resets the master levels for all fader groups to nominal.

(c) ON/OFF
This button mutes and unmutes each input fader group.

(d) Faders
The faders adjust the master levels of the fader groups.

A.2.1.1.3 Procedures for Setting Up Mute Groups

A.2.1.1.3.1 Standard Mute Groups

The procedure for grouping channel "ON" buttons into mute groups is similar to that of standard fader groups as described in Section A.2.1.1.2.1. In particular, there are two pages, namely:

- The "Pair/Group | In Mute" page that is used to set up mute groups (I - P) for input channels 1 - 32 and stereo in channels 1 - 4.
- The "Pair/Group | Out Mute" page that is used to set up mute groups (U - X) for bus outs 1 - 8, aux outs 1 - 8, and stereo out.

An indication of the steps followed when setting up input channel mute groups is given below. A similar procedure is followed for output channel mute groups, although on a different page.

1. The "PAIR/GROUP" button of the display access section is pressed repeatedly until the "Pair/Group | In Mute" page is displayed. Figure A.18 gives an illustration of this page, where there are:

   (a) Rows labelled I - P, indicating the groups available.

   (b) Columns labelled 1 - 16 and 17 - 32, corresponding to input channels 1 - 32.

   (c) Columns labelled 1 - 2 and 3 - 4, corresponding to stereo in channels 1 - 4.

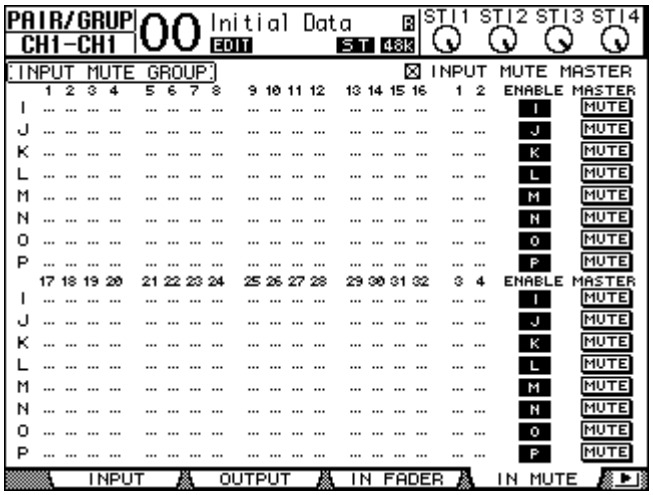


Figure A.18: 01V96 Mixing Console - "Pair/Group | In Mute" Page

(d) Rows with "enable" buttons, labelled I - P, that enable and disable the corresponding mute groups.

2. Similar to fader groups, the up or down buttons are pressed to select a group. The "SEL" buttons for the channels whose "ON" buttons are to be added to the group are pressed and the selected channels are marked with solid circles. The on/off statuses of the grouped channels are determined by the "ON" button status when the channels are added to the group.
3. When a channel from a pair is added to a group, the pair partner is also added to the group.
4. Groups are turned on or off when the "ENTER" button is pressed over the corresponding "enable" button of the group.
5. The grouped channels can be turned on or off while the "enable" button of the group is off.

A.2.1.1.3.2 Master Mute Groups

Similar to the Fader Group Master function, the Mute Group Master function assumes that channel "ON" buttons have already been assigned to groups as outlined in Section A.2.1.1.3.1. We now describe the steps required for the Mute Group Master function in the context of input channels, after the channels have been assigned to groups. Again, we note that a similar procedure is applicable to output channels, although on a different page.

1. The cursor buttons are used to select the “INPUT MUTE MASTER” check box on the “Pair/Group | In Mute” page as shown in Figure A.19. For output channels, the equivalent is the “OUTPUT MUTE MASTER” check box on the “Pair/Group | Out Master” page. The “ENTER” button is pressed to turn on the Mute Group Master function.

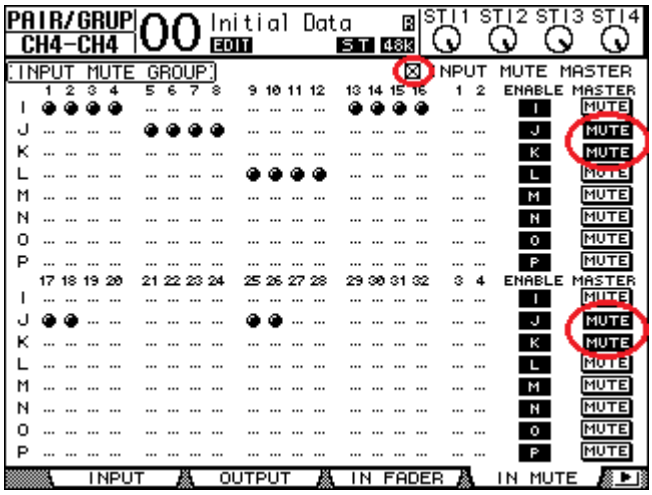


Figure A.19: 01V96 Mixing Console - “Pair/Group | In Mute” Page: Input Mute Master Selected

2. While the “INPUT MUTE MASTER” check box is checked, the group “MASTER MUTE” buttons are used to mute and unmute the groups. In Figure A.19, groups J and K are muted.

A.2.1.1.4 Procedure for Linking EQ and Compressor Parameters

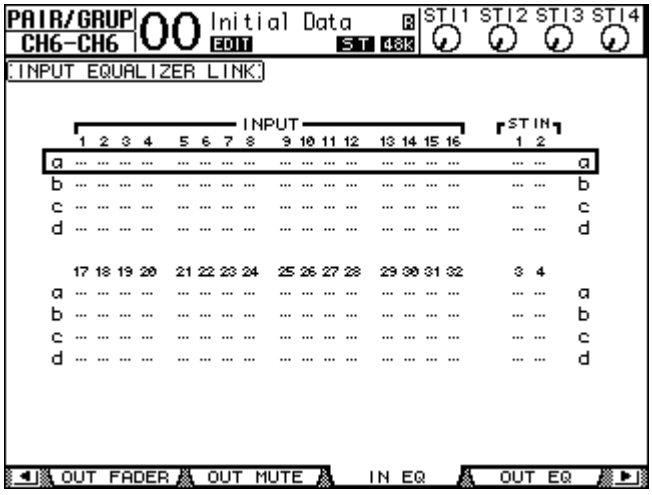
Linking EQ or compressor parameters is done via four pages, namely the “Pair/Group | In EQ”, “Pair/Group | Out EQ”, “Pair/Group | In Comp”, and “Pair/Group | Out Comp” pages, where:

- The “Pair/Group | In EQ” page is used to set EQ links (a - d) for input channels 1 - 32 and stereo in channels 1 - 4.
- The “Pair/Group | Out EQ” page is used to set EQ links (e - h) for bus outs 1 - 8, aux outs 1 - 8, and stereo out.
- The “Pair/Group | In Comp” page is used to set compressor links (i - l) for input channels 1 - 32.

- The “Pair/Group | Out Comp” page is used to set compressor links (m - p) for bus outs 1 - 8, aux outs 1 - 8, and stereo out.

The procedure to set up links is the same across all four pages. We describe the necessary steps with reference to the “Pair/Group | In EQ” page.

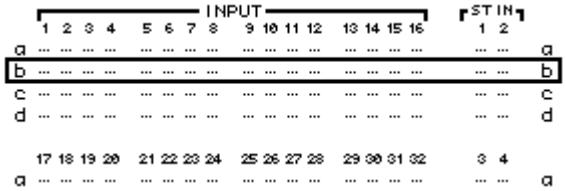
1. The “PAIR/GROUP” button of the display access section is pressed repeatedly until the “Pair/Group | In EQ” page is displayed as shown in Figure A.20. As shown in the figure, the page has four EQ links (a - d) for input channels 1 - 32 and stereo in channels 1 - 4.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner’s Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.20: 01V96 Mixing Console - “Pair/Group | In EQ” Page

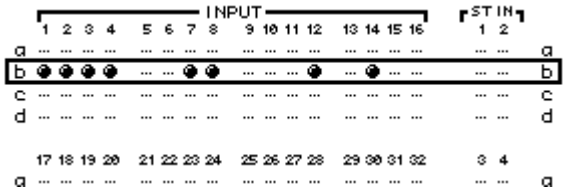
2. The up or down buttons are pressed to select the channels to be added as shown in Figure A.21, where link “b” is selected.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner’s Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.21: 01V96 Mixing Console - “Pair/Group | In EQ” Page: Link Selection

3. The “SEL” buttons for the channels to be added to the link are pressed. The selected channels are marked with solid circles as shown in Figure A.22, where input channels 1 - 4, 7, 8, 12, and 14 have been added to link “b”.



Republished with permission of Yamaha Corporation, from 01V96 Version 2 Owner’s Manual, Yamaha Corporation, 2004; permission conveyed through Shinya Araki, Yamaha Music Gulf FZE, South Africa Office.

Figure A.22: 01V96 Mixing Console - “Pair/Group | In EQ” Page: Link “b” Channels

4. When a channel from a pair is added to a link, the pair partner is also added to the link.
5. The EQ settings for the first channel added to a link are applied to all subsequently added channels. Thereafter, changes to any of the EQ parameters are applied to the rest of the linked channels.

A.2.2 Allen & Heath Limited

A.2.2.1 ML5000 Live Sound Console

Figure A.23 shows a diagram of the ML5000 mixing console. The diagram highlights two main sections of the mixing console that are relevant for the management of control grouping relationships. The highlighted sections are annotated as “D” and “K, L, M”[1]. This annotation is consistent with the annotations used to describe the functions of the ML5000 mixing console in “ML5000 User Guide AP3736 Issue 5” [Allen & Heath Limited, 2003c]. Section “D” comprises an input fader strip containing 24 input faders, “K” comprises controls for VCA groups, “L” comprises controls for mute groups, and “M” comprises control for snapshot memories. Overviews of sections “D” and “M” are given below. Overviews for sections “K” and “L” will be given as the relevant grouping capabilities are presented, in Section A.2.2.1.1 for VCA groups and in Section A.2.2.1.2 for mute groups.

Figure A.24 shows the layout of a single input fader in section “D”.

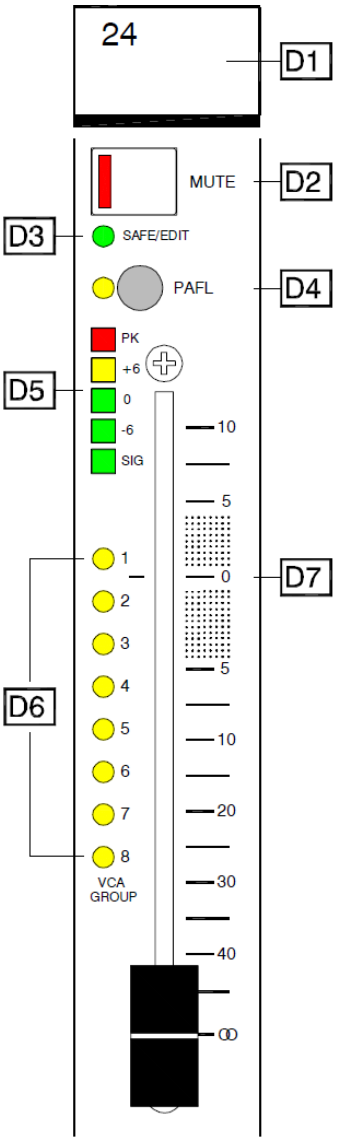
[1]Due to space restrictions, “K, L, M” have been combined into one section although they are separate sections.

Republished with permission of Allen & Heath Limited, from ML5000 User Guide AP3736 Issue 5, Allen & Heath Limited, 2003; permission conveyed through David Kirk, Marketing Manager, Allen & Heath Limited.

Figure A.23: ML5000 Mixing Console

The fader contains seven components labelled “D1” through to “D7”. Components “D2”, “D3”, “D6”, and “D7” are relevant for grouping and have the following functions:

- “D2” (Mute Switch)
  The mute switch turns the channel signal on or off. It is also possible to mute the channel via other means, such as mute groups and VCA groups. The switch is lit when the channel is muted. The switch also functions as an edit key when the mixing console is in “Edit Group” mode.
- “D3” (Safe/Edit LED)
  When the mixing console is in “Edit Group” mode, the LED is lit to indicate that the channel is assigned to the group being edited.
- “D6” (VCA Group Indicators)
  Each channel can be assigned to more than one of eight VCA groups. These LEDs are lit when a channel belongs to the group represented by the corresponding LED. The “Edit Group” function in the VCA master section K is used to assign and remove channels to and from VCA groups. The VCA Group indicators are also used when previewing snapshots of the mixing console. However, snapshots are beyond the scope of this investigation.



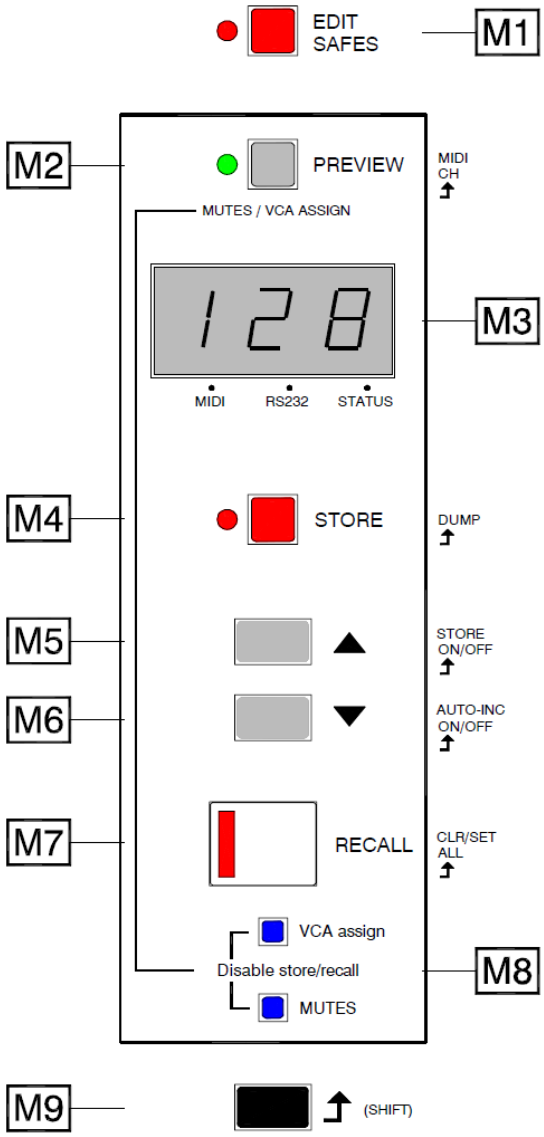
Republished with permission of Allen & Heath Limited, from ML5000 User Guide AP3736 Issue 5, Allen & Heath Limited, 2003; permission conveyed through David Kirk, Marketing Manager, Allen & Heath Limited.

Figure A.24: ML5000 Mixing Console: Input Fader

- "D7" (VCA Fader)

  Recall from Section 2.1.1.1.1 that audio does not pass through the fader for VCA faders. Instead, the audio passes through a VCA chip that is fed with a DC control voltage which results in the audio signal level being adjusted accordingly. In the context of the ML5000 mixing console, the position of the fader is read by the mixing console's computer. The computer then produces a DC voltage to control the levels of the channel VCA.

Figure A.25 shows a layout of the controls used for snapshot memories (section “M”). Although snapshot memories are outside the scope of this investigation, three of the components in section “M” are used to either assign or clear channels from VCA and mute groups, namely components “M3” (Display), “M7” (Recall switch), and “M9” (Shift switch). In the context of VCA and mute groups, the “Recall” switch, when pressed simultaneously with the “Shift” switch, is used to turn all console mutes on and off. The display is used to show information about the function being accessed.

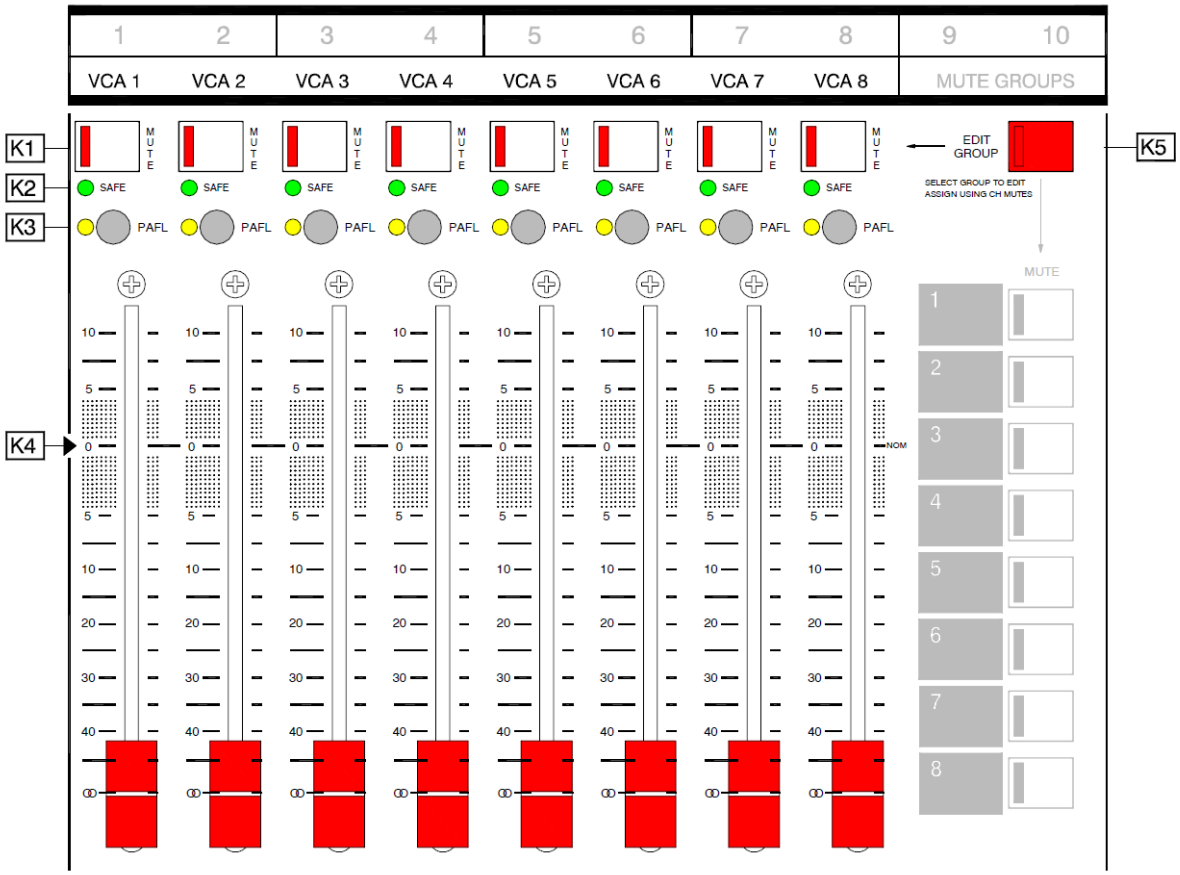


Republished with permission of Allen & Heath Limited, from ML5000 User Guide AP3736 Issue 5, Allen & Heath Limited, 2003; permission conveyed through David Kirk, Marketing Manager, Allen & Heath Limited.

Figure A.25: ML5000 Mixing Console: Snapshot Memories

Procedures for enabling and disabling VCA and mute group capabilities that implemented by the ML5000 mixing console are given below.

A.2.2.1.1 VCA Groups



Republished with permission of Allen & Heath Limited, from ML5000 User Guide AP3736 Issue 5, Allen & Heath Limited, 2003; permission conveyed through David Kirk, Marketing Manager, Allen & Heath Limited.

Figure A.26: ML5000 Mixing Console: VCA Group Controls

Figure A.26 shows a layout of section "K" of the ML5000 mixing console. This is the section where VCA groups are configured. There are eight columns labelled "VCA 1" through to "VCA 8". These columns each represent one of the eight VCA groups implemented by the mixing console. There are five components within this section, labelled "K1" through to "K5". Three of these components are of relevance to this investigation, namely "K1", K4", and "K5", where:

- "K1" (VCA Mute Switch)

The VCA mute switch turns all input channels that are associated with the corresponding VCA group on or off. When the VCA mute switch is turned on, the mute switches of all the associated input channels are also turned on. In contrast, when the VCA mute switch is turned off, the mute switches of all the associated input channels are also turned off.

- "K4" (VCA Group Fader)
  The VCA group fader adjusts the levels of all channels that are assigned to the VCA fader's group.
- "K5" (Edit Group Switch)
  The edit group switch is used to put the console into "Edit Group" mode.

A.2.2.1.1.1 Procedures for Assigning and Clearing VCA Groups

In order to assign channels to a VCA group, the following procedures are required:

1. The "Edit Group" ("K5") switch is pressed to put the console into edit mode.
2. The "VCA Group Mute" ("K1") switch for the group to be edited is pressed. At this point, both switches "K1" and "K5" will be flashing.
3. The channel "Safe/Edit" ("D3") LEDs display the channels that are currently assigned to the group being edited.
4. The channel "Mute" ("D2") switches are pressed to toggle channels in and out of the group.
5. The "Edit Group" switch is pressed again to exit edit mode. Alternatively, another "VCA Group Mute" switch is pressed to edit another group.

To clear all channels from a VCA group while editing the VCA group, the following procedures are required:

1. The "Shift" ("M9") and "Recall" (M7") switches are pressed simultaneously.
2. The "Recall" switch flashes while the display flashes *CLr*.
3. The "Shift" switch is released and the "Recall" switch is pressed to confirm.
4. All channel "Safe/Edit" LEDs turn off, indicating that the group is cleared.

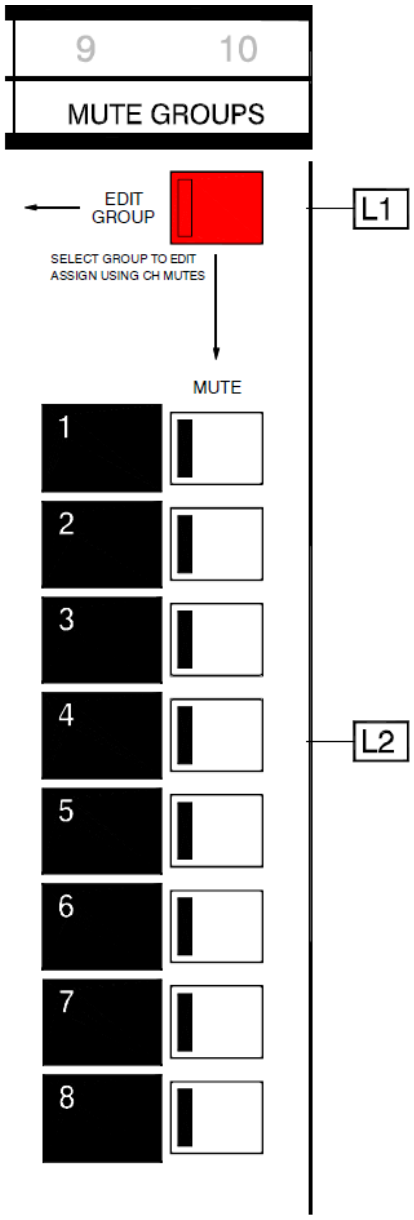
To assign all channels to a VCA group while editing the VCA group, the following procedures are required:

1. The “Shift” (“M9”) and “Recall” (M7”) switches are pressed simultaneously twice until the display flashes *SEt.*
2. The “Shift” switch is released and the “Recall” switch is pressed to confirm.
3. All channel “Safe/Edit” LEDs turn on, indicating that all channels are assigned to the group.

A.2.2.1.2 Mute Groups

Figure A.27 shows an illustration of the mute group controls on an ML5000 mixing console. There are two main components, labelled “L1” and “L2”, where:

- “L1” (Edit Group Switch)
  This switch is used to put the console in “Edit Group” mode. Note that this is the same switch that is referred to as component “K5” in the description of VCA groups in Section A.2.2.1.1.
- “L2” (Mute Group Switches)
  These are the eight switches next to the rows labelled “1” through to “8”, representing mute groups 1 - 8. These momentary action switches are used to mute and unmute channels that are assigned to mute groups. When a group is muted, the switch is illuminated and in contrast, when the group is unmuted, the switch is not illuminated.



Republished with permission of Allen & Heath Limited, from ML5000 User Guide AP3736 Issue 5, Allen & Heath Limited, 2003; permission conveyed through David Kirk, Marketing Manager, Allen & Heath Limited.

Figure A.27: ML5000 Mixing Console: Mute Group Controls

A.2.2.1.2.1 Procedures for Assigning and Clearing Mute Groups

Assigning channels to mute groups follows a similar procedure to that of assigning channels to VCA groups. To assign channels to a mute group, the following steps are required:

1. The “Edit Group” (“L1”) switch is pressed to put the mixing console into “Edit Group”

mode.

2. The “Mute Group” (“L2”) switch of the group to be edited is pressed. At this point, both switched “L1” and “L2” will be flashing.
3. The channel “Safe/Edit” (“D3”) LEDs display the channels that are currently assigned to the group being edited.
4. The channel “Mute” (“D2”) switches are pressed to toggle channels in and out of the group.
5. The “Edit Group” switch is pressed again to exit edit mode. Alternatively, another “Mute Group” switch is pressed to edit another group.

Clearing all channels from a mute group follows exactly the same procedure as described for clearing all channels from a VCA group. To clear all channels from a mute group while editing the mute group, the following procedures are required:

1. The “Shift” (“M9”) and “Recall” (“M7”) switches are pressed simultaneously.
2. The “Recall” switch flashes while the display flashes *CLr*.
3. The “Shift” switch is released and the “Recall” switch is pressed to confirm.
4. All channel “Safe/Edit” LEDs turn off, indicating that the group is cleared.

Again, assigning all channels to a mute group follows exactly the same procedure as described for assigning all channels to a VCA group. To assign all channels to a mute group while editing the mute group, the following procedures are required:

1. The “Shift” (“M9”) and “Recall” (“M7”) switches are pressed simultaneously twice until the display flashes *SEt*.
2. The “Shift” switch is released and the “Recall” switch is pressed to confirm.
3. All channel “Safe/Edit” LEDs turn on, indicating that all channels are assigned to the group.

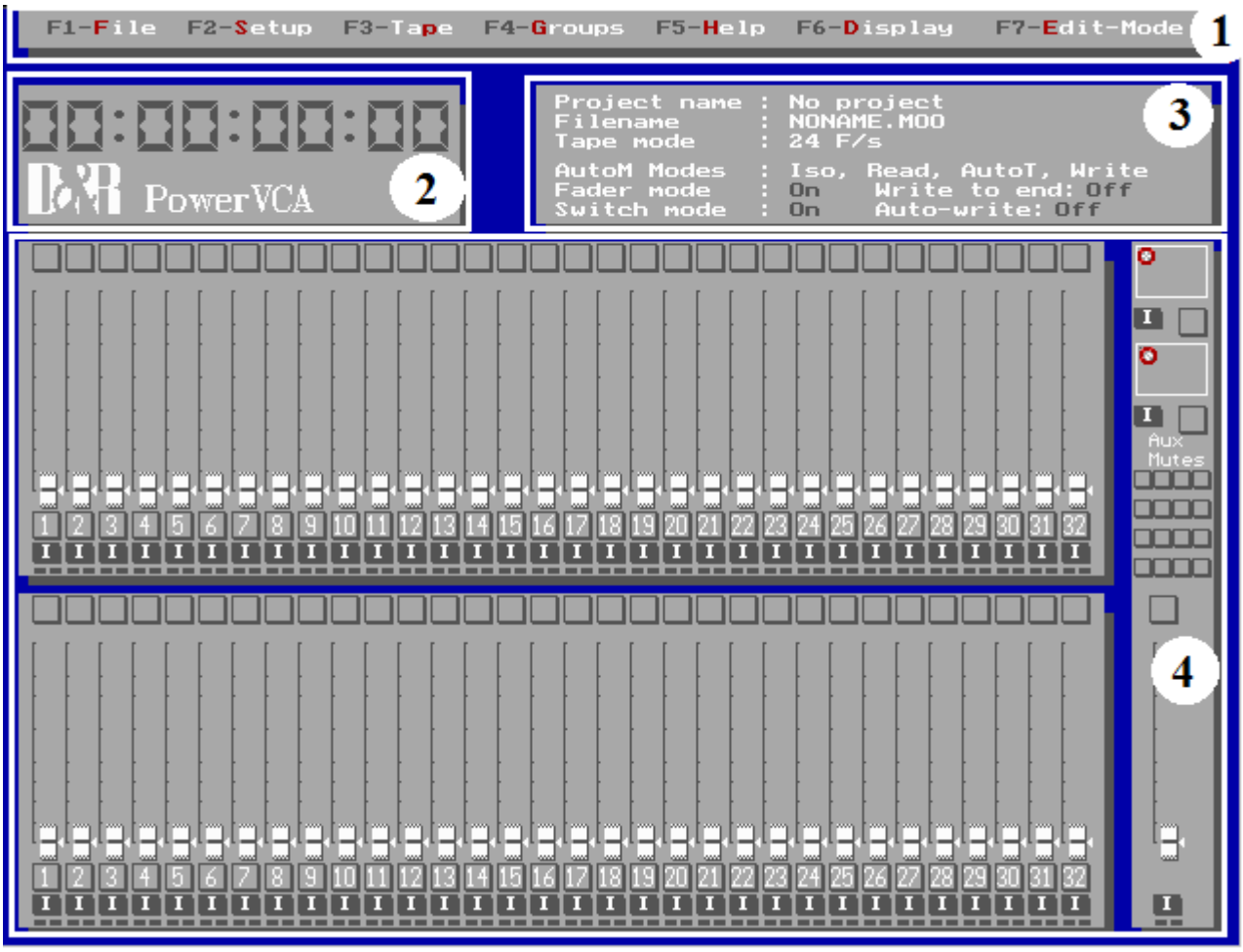


Figure A.28: PowerVCA: Mix Screen [D&R Electronica Weesp b.v., 2010a]

A.2.3 D&R Electronica Weesp b.v.

A.2.3.1 PowerVCA

Figure A.28 show an illustration of the main screen that PowerVCA displays. As shown in the figure, there are four main sections, namely:

1. Menu bar
   This bar provides access to the various capabilities of PowerVCA. For example, the grouping capabilities of PowerVCA are accessed by pressing the “F4” key or “Alt + G” keys on the keyboard.

2. Timecode bar
   This bar shows the timecode that the computer is receiving.
3. Status bar
   The status bar displays information related to the current mode of operation of PowerVCA and the attached D&R mixing console.
4. Mix section
   The mix section displays console information. It shows a number of channel modules, equivalent to the number on the attached mixing console. The master section is also displayed if the attached mixing console has an automated master section.
   Figure A.29 shows the main components of the channel modules in PowerVCA, namely:

   (a) Mute button
   This button shows the state of the associated channel mute button on the attached console. The state of the mute button can either be changed from within PowerVCA or on the mixing console.

   (b) Fader
   The fader shows the exact position of the mixing console's fader. If the fader is configured as a slave or master, the knob is rendered using a selected group colour.

   (c) Channel number
   This is a label for the channel's number.

   (d) Select button
   A select switch for changing the channel's state based on the console's automation mode.

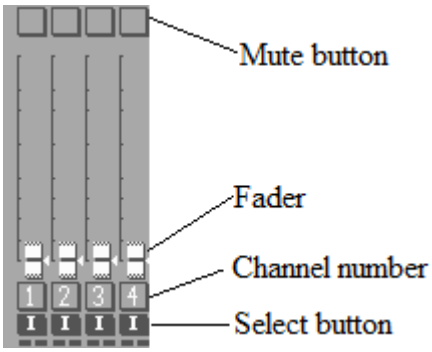


Figure A.29: PowerVCA: Channel Module [D&R Electronica Weesp b.v., 2010a]

A.2.3.1.1 PowerVCA Grouping Procedures

The menu bar shown in section 1 of Figure A.28 contains a menu option that provides access to the grouping capabilities of PowerVCA. In particular, if the "F4" or "Alt + G" keys are pressed on the keyboard there is a display of two further menus, one for masters and another for slaves. No diagrams illustrating these two menus were available. However, the menus are used as described below.

On the masters menu:

- The arrow keys allow for selection of a master fader from the channel faders.
- All eight group masters are displayed on the left top side of the sub menu screen, where each group has a different colour.
- A number between 1 and 8 is chosen to associate the selected master with the corresponding group.
- Masters are removed by pressing the space bar key while the master is selected.

On the slaves menu:

- All master faders associated with the various groups via the masters menu are displayed.
- Arrow keys are used to select a slave from the channels. The selected slave is coupled to the master by choosing of the numbers 1 to 8 as selected earlier in the masters menu.
- Slaves get the same colour as the master.
- Slaves are removed by pressing the space bar key while the slave is selected.

A.2.3.2 Sirius

Figure A.30 shows the PC screen that is used to access and configure various capabilities of the mixing console. The screen shows a number of modules that represent, for example, input or output channels of the mixing console. In order to access fader grouping capabilities for each channel, group buttons exist for each channel fader. These group buttons are highlighted in Figure A.30.

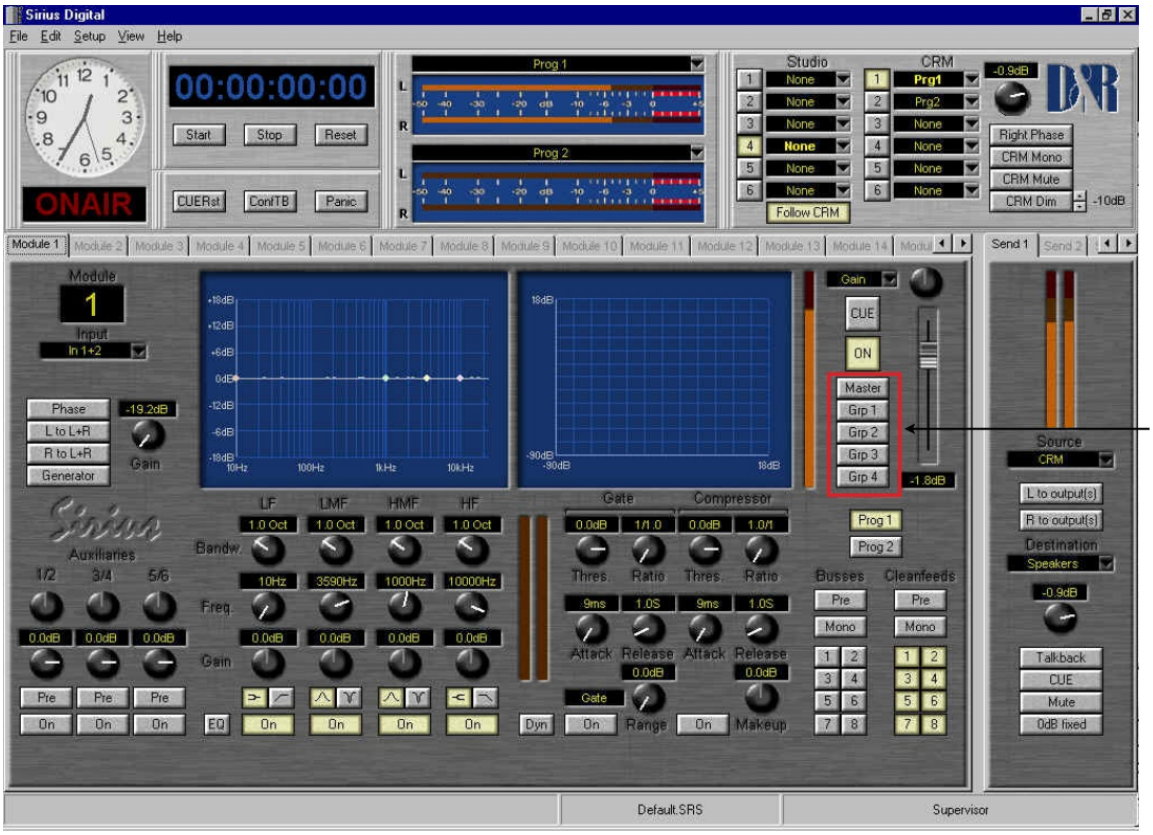


Figure A.30: Sirius Mixing Console Control Page [D&R Electronica Weesp b.v., 2010c]

Figure A.31 focuses closely on the buttons that are used to access fader grouping capabilities of each channel on the Sirius mixing console. In particular, there are five buttons labelled “Master”, “Grp 1”, “Grp 2”, “Grp 3”, and “Grp 4”. We now describe how these buttons are used with reference to procedures for specifying group masters and slaves.

To specify a group master:

1. The “Master” button of the channel that is to become the master of the group is selected.
2. The group button (“Grp 1” through to “Grp 4”) for which the fader is master is also selected. There can only be one master per group.

To specify a slave, only the corresponding group button (“Grp 1” through to “Grp 4”) for the channel is selected.

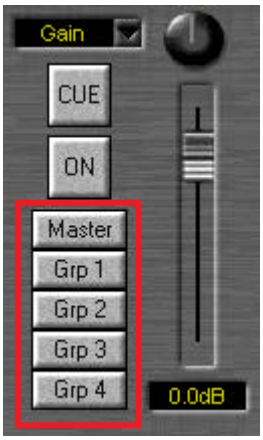


Figure A.31: Sirius Mixing Console Control Page: Group Buttons [D&R Electronica Weesp b.v., 2010b]

Appendix B

Validation for Relationship Inference Rules

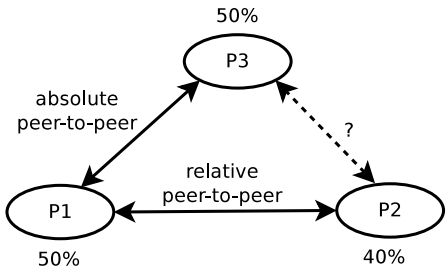
Section 5.5.2.2 identifies seven combinations of relationships that were validated in order to prove the assertion made by the inference rules, where an unknown relationship is absolute if and only if the known relationships from which the inference is made are both absolute. The seven combinations that were validated are shown in Table B.1 below.

Combination	P1-P2 Relationship	P1-P3 Relationship	Inferred P2-P3 Relationship
1	Relative Peer-to-Peer	Absolute Peer-to-Peer	Relative Peer-to-Peer
2	Relative Peer-to-Peer	Relative Master-Slave	Relative Master-Slave
3	Relative Peer-to-Peer	Absolute Master-Slave	Relative Master-Slave
4	Relative Peer-to-Peer	Relative Peer-to-Peer	Relative Peer-to-Peer
5	Absolute Peer-to-Peer	Relative Master-Slave	Relative Master-Slave
6	Absolute Peer-to-Peer	Absolute Master-Slave	Absolute Master-Slave
7	Absolute Peer-to-Peer	Absolute Peer-to-Peer	Absolute Peer-to-Peer

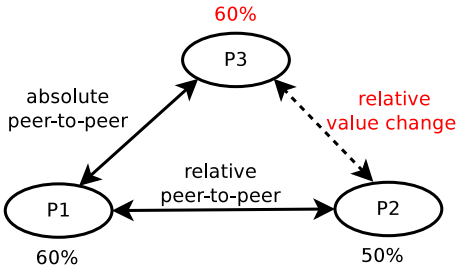
Table B.1: Combinations of Inference Rules Validated

This appendix describes how each of these combinations were validated.

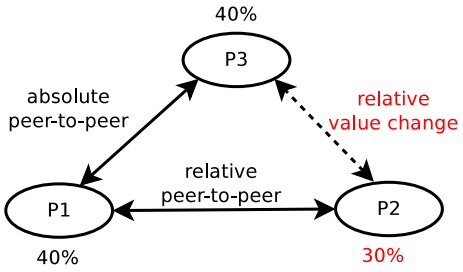
B.1 Combination 1 Validation: Relative Peer-to-Peer and Absolute Peer-to-Peer



(a) Initial state



(b) Value change in P3 results in relative value change in P2



(c) Value change in P2 results in relative value change in P3

Figure B.1: Combination 1: Relative Peer-to-Peer (P1-P2) and Absolute Peer-to-Peer (P1-P3)

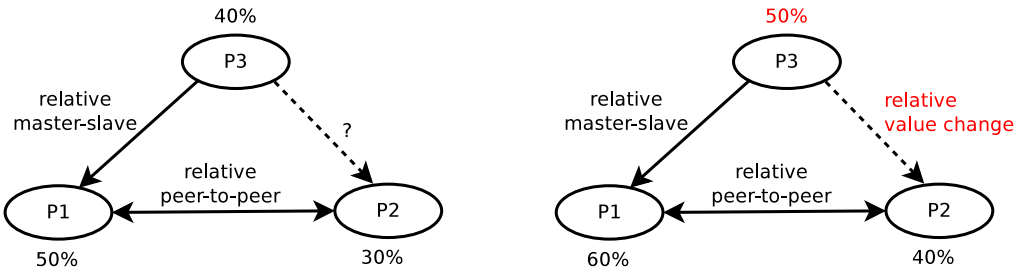
Figure B.1 shows an example of a configuration that validates Combination 1. As shown in Figure B.1a, there are:

- Three parameters P1, P2, and P3 with values 50%, 40%, and 50%, respectively.
- Relative peer-to-peer relationship between P1 and P2.
- Absolute peer-to-peer relationship between P1 and P3, thus P1 and P3 have the same value (50%).
- A peer-to-peer relationship between parameters P2 and P3, where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by observing the manner in which values of either parameters are influenced by value changes of the other. For example, an increase in value of P3 from the initial state by 10% (from 50% to 60%) results in P1 taking the value of P3, 60%, by virtue of the absolute peer-to-peer relationship between P3 and P1. In order for the relative peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 also increases by 10% (from 40% to 50%); the net result being a

relative change in value between P3 and P2 as shown in Figure B.1b. Conversely, a reduction in value of P2 from the initial state by, for example, 10% (from 40% to 30%), results in a similar reduction in value of parameter P1 (from 50% to 40%), which in turn causes an absolute change in value of P3 to 40%; the net result being a relative change in value between P3 and P2 as shown in Figure B.1c. Therefore, a relative peer-to-peer relationship between P2 and P3 is implied.

B.2 Combination 2 Validation: Relative Peer-to-Peer and Relative Master-Slave



(a) Initial State

(b) Implied Relative Master-Slave Relationship between P2 and P3

Figure B.2: Combination 2: Relative Peer-to-Peer (P1-P2) and Relative Master-Slave (P1-P3)

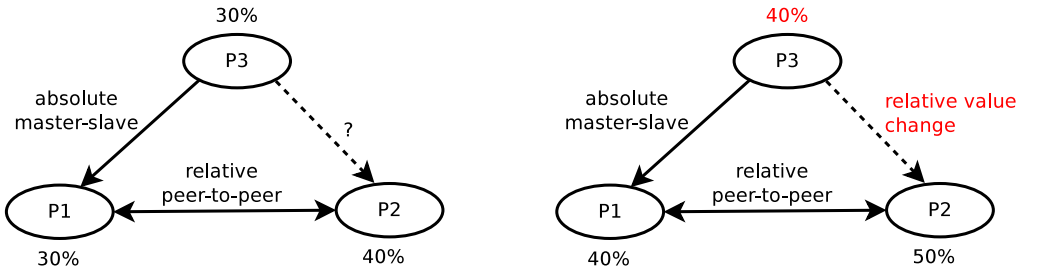
Figure B.2 shows an example of a configuration that validates Combination 2. As shown in Figure B.2a, there are:

- Three parameters P1, P2, and P3 with values 50%, 30%, and 40%, respectively.
- Relative peer-to-peer relationship between parameters P1 and P2.
- Relative master-slave relationship between parameters P1 and P3, where P3 is the master.
- A master-slave relationship between parameters P2 and P3 (master), where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by the observing the manner in which the value of P2 is influenced by changes in value of P3. For example, an increase in value of P3 by 10% (from 40% to 50%) results in a 10% increase in value of P1 from 50% to 60%, by virtue of the relative master-slave relationship between P1 and

P3. In order for the relative peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 should also increase by 10% (from 30% to 40%). Therefore, a relative peer-to-peer relationship between P2 and P3 is implied as shown in Figure B.2b.

B.3 Combination 3 Validation: Relative Peer-to-Peer and Absolute Master-Slave



(a) Initial State

(b) Implied Relative Master-Slave Relationship between P2 and P3

Figure B.3: Combination 3: Relative Peer-to-Peer (P1-P2) and Absolute Master-Slave (P2-P3)

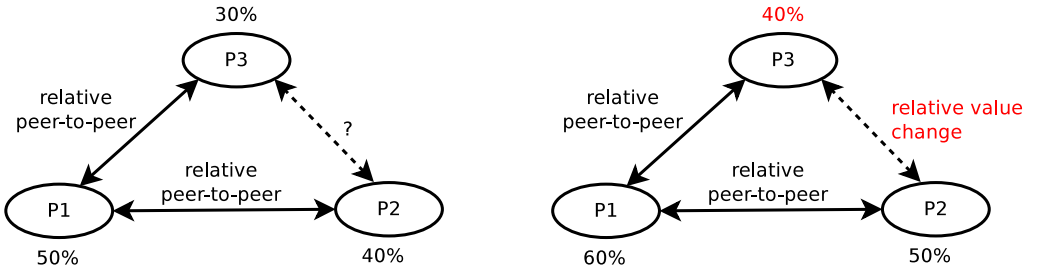
Figure B.3 shows an example of a configuration that validates Combination 3. As shown in Figure B.3a, there are:

- Three parameters P1, P2, and P3 with values 30%, 40%, and 30%, respectively.
- Relative peer-to-peer relationship between parameters P1 and P2.
- Absolute master-slave relationship between parameters P1 and P3, where P3 is the master.
- A master-slave relationship between parameters P2 and P3 (master), where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by the observing the manner in which the value of P2 is influenced by changes in value of P3. For example, an increase in value of P3 by 10% (from 30% to 40%) results in P1 taking on the value of P3 (40%), by virtue of the absolute master-slave relationship between P3 and P1. In order for the relative peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 increases by 10% (from 40% to 50%); the net result is a relative change in value of P2 when compared to

that of P3. Therefore, a relative master-slave relationship between P2 and P3 is implied as shown in Figure B.3b.

B.4 Combination 4 Validation: Relative Peer-to-Peer and Relative Peer-to-Peer



(a) Initial State

(b) Implied Relative Peer-to-Peer Relationship between P2 and P3

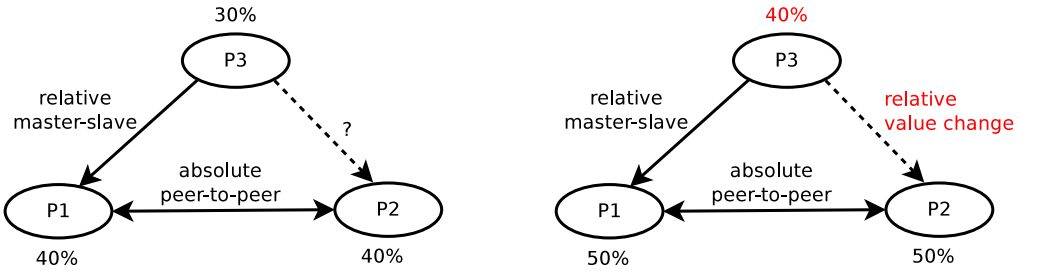
Figure B.4: Combination 4: Relative Peer-to-Peer (P1-P2) and Relative Peer-to-Peer (P2-P3)

Figure B.4 shows an example of a configuration that validates Combination 4. As shown in Figure B.4a, there are:

- Three parameters P1, P2, and P3 with values 50%, 40%, and 30%, respectively.
- Relative peer-to-peer relationships between parameters P1 and P2, and between P1 and P3.
- A peer-to-peer relationship between parameters P2 and P3, where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by the observing the manner in which values of either parameters are influenced by value changes of the other. For example, an increase in value of P3 by 10% (from 30% to 40%) results in an increase in value of P1 by 10% (from 50% to 60%), by virtue of the relative peer-to-peer relationship between P1 and P3. In order for the relative peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 also increases by 10% (from 40% to 50%). Conversely, a change in value of P2 results in a similar change in value of parameter P1, which in turn causes a relative change in value of P3. Therefore, a relative peer-to-peer relationship between P2 and P3 is implied as shown in Figure B.4b.

B.5 Combination 5 Validation: Absolute Peer-to-Peer and Relative Master-Slave



(a) Initial State

(b) Implied Relative Master-Slave Relationship between P2 and P3

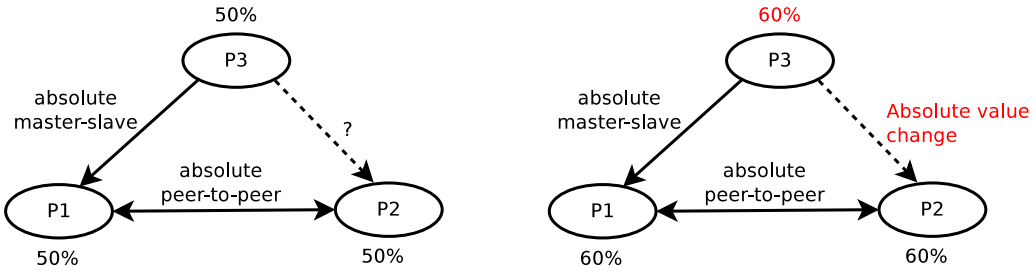
Figure B.5: Combination 5: Absolute Peer-to-Peer (P1-P2) and Relative Master-Slave (P1-P3)

Figure B.5 shows an example of a configuration that validates Combination 5. As shown in Figure B.5a, there are:

- Three parameters P1, P2, and P3 with values 40%, 40%, and 30%, respectively.
- Absolute peer-to-peer relationship between parameters P1 and P2.
- Relative master-slave relationship between parameters P1 and P3, where P3 is the master.
- A master-slave relationship between parameters P2 and P3 (master), where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by the observing the manner in which the value of P2 is influenced by changes in value of P3. For example, an increase in value of P3 by 10% (from 30% to 40%) results in a 10% increase in the value of P1 (from 40% to 50%) by virtue of the relative master-slave relationship between P1 and P3. In order for the absolute peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 must be the same as that of P1, 50%; the net result is a relative change in value of P2 when compared to that of P3. Therefore, a relative master-slave relationship between P2 and P3 is implied as shown in Figure B.5b.

B.6 Combination 6 Validation: Absolute Peer-to-Peer and Absolute Master-Slave



(a) Initial State

(b) Implied Absolute Master-Slave Relationship between P2 and P3

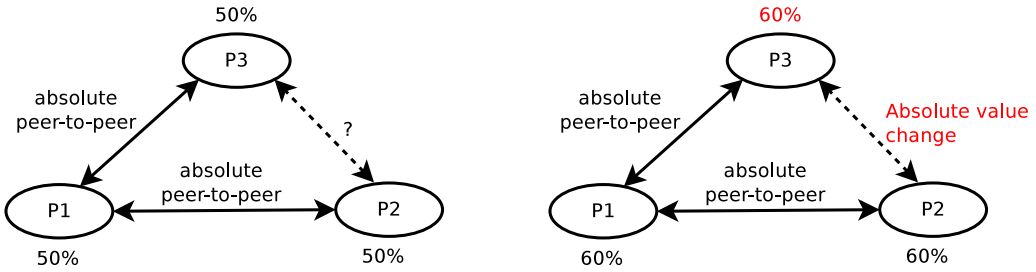
Figure B.6: Combination 6: Absolute Peer-to-Peer (P1-P2) and Absolute Master-Slave (P1-P3)

Figure B.6 shows an example of a configuration that validates Combination 6. As shown in Figure B.6a, there are:

- Three parameters P1, P2, and P3 all with the value of 50%.
- Absolute peer-to-peer relationship between parameters P1 and P2.
- Absolute master-slave relationship between parameters P1 and P3, where P3 is the master.
- A master-slave relationship between parameters P2 and P3 (master), where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by the observing the manner in which the value of P2 is influenced by changes in value of P3. For example, an increase in value of P3 by 10% (from 50% to 60%) results in P1 taking on the value of P2, 60%, by virtue of the absolute master-slave relationship between P1 and P3. In order for the absolute peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 must be the same as the value of P1. Since the value of P1 is the same as the value of P3, it follows that there is an implied absolute master-slave relationship between P2 and P3 as shown in Figure B.6b.

B.7 Combination 7 Validation: Absolute Peer-to-Peer and Absolute Peer-to-Peer



(a) Initial State

(b) Implied Absolute Peer-to-Peer Relationship between P2 and P3

Figure B.7: Combination 7: Absolute Peer-to-Peer (P1-P3) and Absolute Peer-to-Peer (P2-P3)

Figure B.7 shows an example of a configuration that validates Combination 7. As shown in Figure B.7a, there are:

- Three parameters P1, P2, and P3 all with values of 50%.
- Absolute peer-to-peer relationships between parameters P1 and P2, and between P1 and P3.
- A peer-to-peer relationship between parameters P2 and P3, where the relative or absolute nature of the relationship is unknown (?).

The relative or absolute nature of the relationship between P2 and P3 can be determined by the observing the manner in which values of either parameters are influenced by value changes of the other. For example, an increase in value of P3 by 10% (from 50% to 60%) results in P1 taking on the value of P3, 60%, by virtue of the absolute peer-to-peer relationship between P1 and P3. In order for the absolute peer-to-peer relationship between P1 and P2 to remain valid, the value of P2 must be the same as the value of P1, 60%. Since the value of P1 is the same as the value of P3, it follows that there is an implied absolute relationship between P2 and P3 as shown in Figure B.7b.