

PRIME Automation User Guide

Version 4.10.11

December 2025



Chyron PRIME Automation User Guide • 4.10.11 • December 2025 • This document is distributed by Chyron in online (electronic) form only, and is not available for purchase in printed form.

This document is protected under copyright law. An authorized licensee of Chyron PRIME Automation may reproduce this publication for the licensee's own use in learning how to use the software. This document may not be reproduced or distributed, in whole or in part, for commercial purposes, such as selling copies of this document or providing support or educational services to others.

Product specifications are subject to change without notice and this document does not represent a commitment or guarantee on the part of Chyron and associated parties. This product is subject to the terms and conditions of Chyron's software license agreement. The product may only be used in accordance with the license agreement.

Any third party software mentioned, described or referenced in this guide is the property of its respective owner. Instructions and descriptions of third party software is for informational purposes only, as related to Chyron products and does not imply ownership, authority or guarantee of any kind by Chyron and associated parties.

This document is supplied as a guide for Chyron PRIME Automation. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Chyron and associated companies do not accept responsibility of any kind for customers' losses due to the use of this document. Product specifications are subject to change without notice.

Copyright © 2025 Chyron, ChyronHego Corp. and its licensors. All rights reserved.

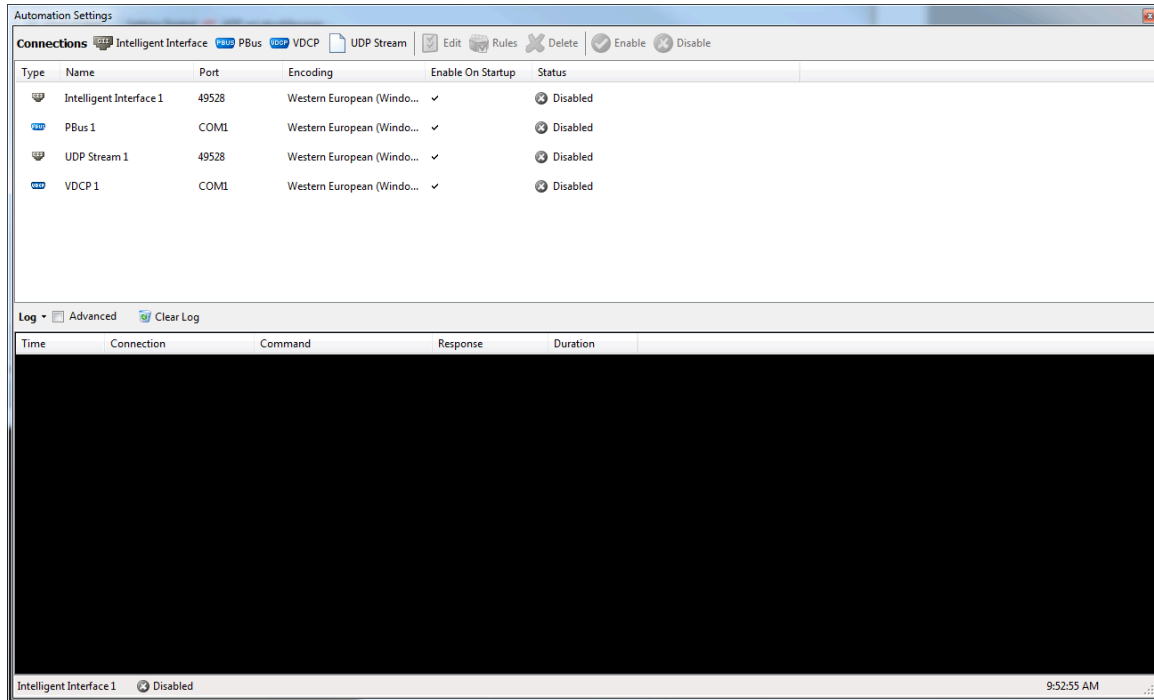
Table of Contents

Introduction.....	5
PBus.....	7
Adding a PBus Connection.....	7
Serial Port and Address Configuration.....	8
Register List.....	12
Commands.....	13
Learn Command.....	13
Query Command.....	13
Recall Command.....	13
Trigger Command.....	13
Write Command.....	13
Intelligent Interface.....	14
Introduction.....	14
Adding a Connection.....	14
Editing a Connection.....	17
Deleting a Connection.....	18
Enabling/Disabling a Connection.....	18
Command Processing.....	18
Default Heartbeats.....	18
Rule Engine.....	19
Rule Terminology.....	19
Rule Editor.....	19
Behaviors.....	21
Supported Parameters.....	25
Setting Default Parameter Values.....	28
PRIME Runtime Operation.....	29
P Commands.....	30
B Command Set (Query).....	34
Message Command Set.....	36
Deprecated Command Set.....	36
Checksum Calculation.....	37
Error Codes.....	37
VDCP.....	38
Generic.....	40
Oxtel.....	44
Command Support.....	45

UDP Stream.....	46
Message Object.....	48
EAS.....	51

INTRODUCTION

The **Automation Settings** panel provides the ability to add and configure an automation connection. It also displays an automation log once the connection is live.



The following parameters are displayed in the **Automation Settings** panel.

- **Type** - The type of Automation connection being used. The automation may be displayed as an **Intelligent Interface**, **PBus**, **VDCP** or **UDP Stream** icon.
- **Name** - The name of the Automation connection.
- **Port** - The port number correlating to the Automation connection.
- **Encoding** - The type of encoding being used by the Automation connection.
- **Enable on Startup** - If enabled, the Automation connection will be enabled every time PRIME is started.
- **Status** - Displays **Enabled**, **Disabled** or **Waiting for Connection** as the current status of the Automation connection.

To add an Automation connection:

- Select **Intelligent Interface** to add an Intelligent Interface connection.
- Select **PBus** to add a PBus connection.
- Select **VDCP** to add a VDCP connection.
- Select **UDP Stream** to add a UDP Stream connection.

To configure an Automation connection:

- Select **Edit** to edit the properties of the highlighted Automation connection.
- Select **Rules** to set Rules for **ONLY** Intelligent Interface connections.

To delete an Automation connection:

- Select **Delete** to delete the highlighted Automation connection.

To enable/disable an Automation connection:

- Select **Enable/Disable** to enable or disable an Automation connection.

To configure the **Automation Log**:

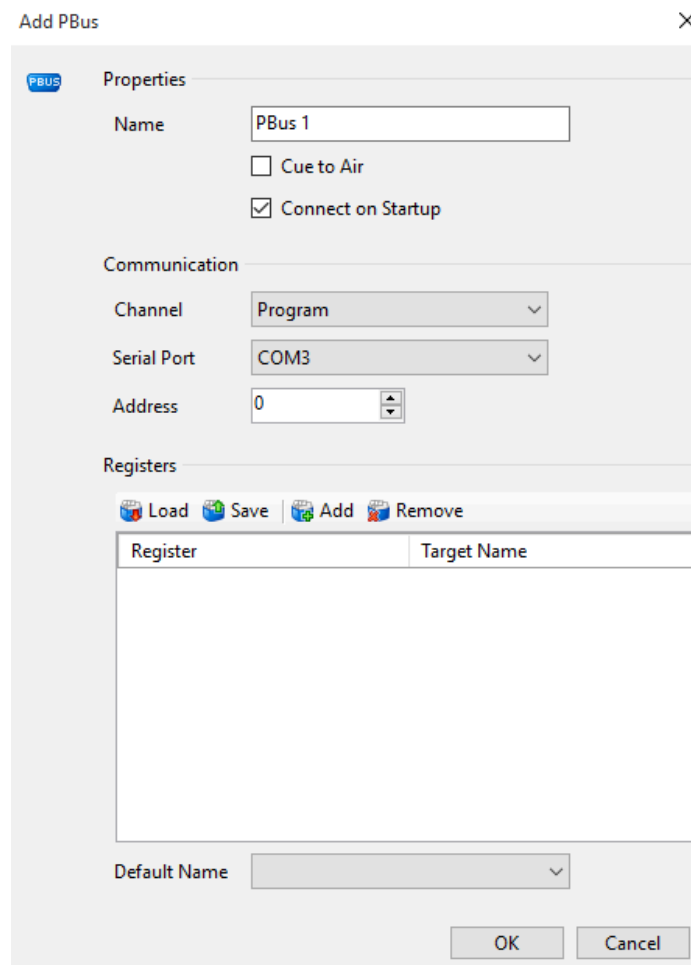
- Select the **Automation Log** dropdown menu to either **Copy** or **Save** the log.
- Enable **Show Data** to show the data from the Automation Log.
- Select **Clear Log** to clear the displayed data from the Automation Log.

PBus

PBus is one of the automation protocols used to control playout of a PRIME system. Communication is over a serial connection. However, the same connection may be used to service multiple PBus Connections.

Adding a PBus Connection

A new PBus connection may be added by clicking the PBus button in the Connections toolbar. Once clicked, the following dialog will appear with a variety of configurable options.



The "Add PBus" dialog box is a standard Windows-style window with a title bar and a close button. It contains several sections for configuring a PBus connection. The "Properties" section includes a "Name" text box (containing "PBus 1"), a "Cue to Air" checkbox (unchecked), and a "Connect on Startup" checkbox (checked). The "Communication" section includes a "Channel" dropdown (set to "Program"), a "Serial Port" dropdown (set to "COM3"), and an "Address" spinner box (set to "0"). The "Registers" section features a toolbar with "Load", "Save", "Add", and "Remove" icons, and a table with two columns: "Register" and "Target Name". The table is currently empty. At the bottom, there is a "Default Name" dropdown and "OK" and "Cancel" buttons.

Register	Target Name
----------	-------------

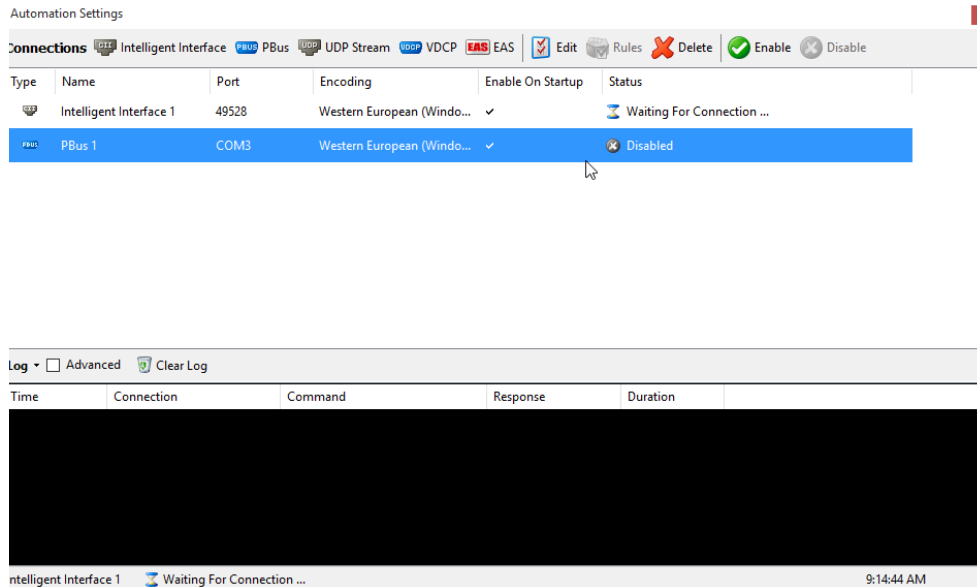
- **Name** – Uniquely identifies the connection.
- **Cue to Air** – Determines the behavior of the recall command. If enabled, a Recall command will play the scene to Program. Otherwise, the scene will be loaded to Preview. By default, this option is disabled.
- **Connect on Startup** – Indicates whether the connection will be automatically enabled whenever PRIME is started.

- **Channel** – Indicates the target channel (preview/program pair) to which scenes will be loaded and played.
- **Serial Port** – Indicates which serial port will be used to send and receive data. The list of available serial ports will include both physical and installed COM ports, as well as reference to other PBus connections in PRIME. The latter allows a user to setup multiple PBus connections that receive data on a single serial port, with the main connection forwarding to the rest.
- **Address** – Specifies an identifier that can be used by a connection to determine whether received commands should be processed. If multiple PBus connections are sharing access to a single COM port, then the identifier setup in each PBus connection determines which commands are relevant.
- **Registers** – Allow the user to define a mapping of PBus command register values to specific scenes. PBus commands that are received may provide a specific register value that should be affected; consequently, this configurable list determines which scenes should be affected.

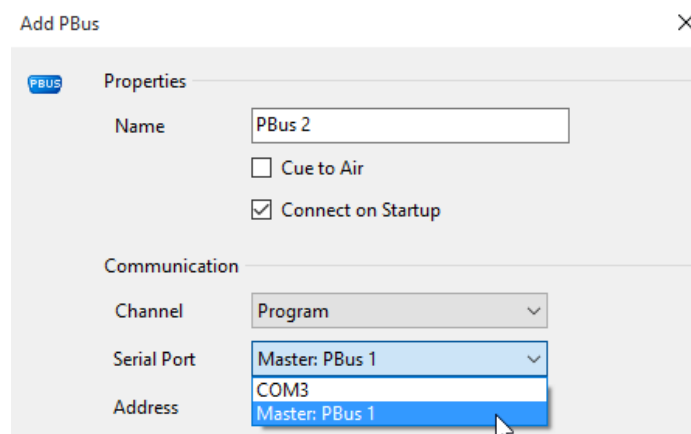
Serial Port and Address Configuration

In some cases, the user may want to configure multiple PBus connections, each with independent register lists. However, there may not be enough available serial ports on the system for a one-to-one mapping. If attempting to setup multiple PBus connections, such that each uses the same physical serial port, then only the first to connect to the port will succeed. All others will fail, stating that the port is already in use.

To work around this limitation, PBus connections may be configured to utilize either a physical serial port, or another PBus connection. For example:



- PBus 1 can use Serial Port: COM3
- PBus 2 can use Serial Port: Master (PBus 1)
- PBus 3 can use Serial Port: Master (PBus 2)



Notice below that the Port for PBus 2 actually lists “Master: PBus 1” rather than COM3. Commands will be received on COM3 by the “PBus 1” connection. However, they will then be forwarded to any other attached connections (e.g. PBus 2 in this case).

Automation Settings

Connections

Intelligent InterfacePBusUDP StreamVDCPEASEditRulesDeleteEnableDisable

Type	Name	Port	Encoding	Enable On Startup	Status
Intelligent Interface	Intelligent Interface 1	49528	Western European (Windo...	✓	Waiting For Connection ...
PBus	PBus 1	COM3	Western European (Windo...	✓	Disabled
PBus	PBus 2	Master: PBus 1	Western European (Windo...	✓	Disabled

Log

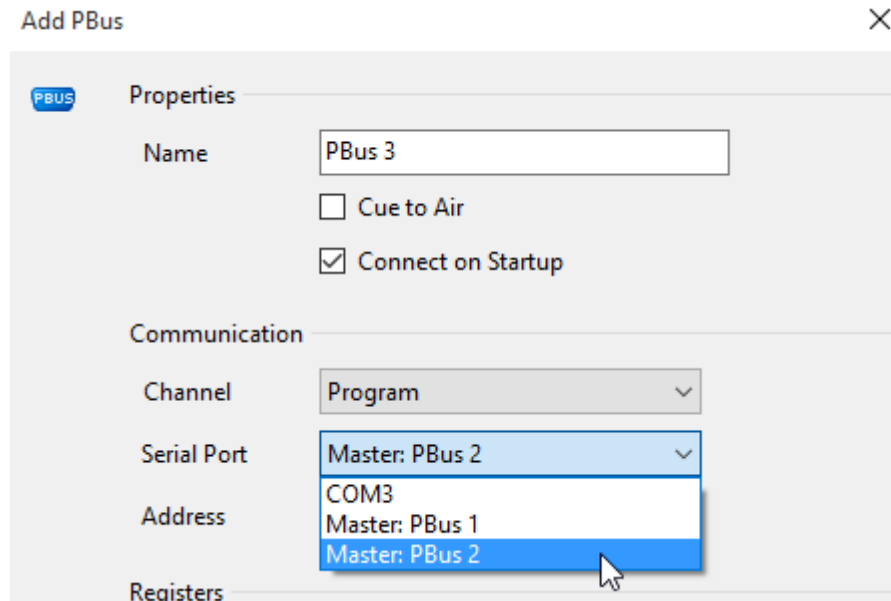
☐ Advanced

Clear Log

Time	Connection	Command	Response	Duration
------	------------	---------	----------	----------

Intelligent Interface 1Waiting For Connection ...9:14:44 AM

Similarly, when adding an additional PBus connection, the user may target the same master connection (PBus 1), or add another layer of forwarding by using PBus 2 connection.



In practice, this means that the user may configure either of the following scenarios:

- **PBus 1** (COM 3)
 - **PBus 2** (PBus 1)
 - **PBus 3** (PBus 2)

In the above scenario, a command is received on COM3 by PBus 1. If the address configured for PBus 1 matches the address specified in the command, then PBus 1 will process the command using its register list. Regardless, the command is then forwarded to PBus 2, which also verifies its address to determine processing. Lastly, PBus 2 forwards the command to PBus 3, which also verifies its address to determine processing.

- **PBus 1** (COM 3)
 - **PBus 2** (PBus 1)
 - **PBus 3** (PBus 1)

Register List

The register list defines a mapping between command register numbers and PRIME scenes. A register number may only affect a single scene. The register list may be configured entirely by a PBus automation system using the **Recall** and **Learn** commands. However, manual configuration is also supported using the Add and Remove buttons as seen below.

The register list may be exported and imported to transfer settings between connections.

The screenshot shows the 'Add PBus' dialog box with the following sections and controls:

- Properties:**
 - Name: PBus 1
 - ☐ Cue to Air
 - ☒ Connect on Startup
- Communication:**
 - Channel: Program
 - Serial Port: COM3
 - Address: 0
- Registers:**
 - Buttons: Load, Save, Add, Remove
 - Table:

Register	Target Name
1	test
- Default Name:** (empty dropdown)
- Buttons: OK, Cancel

The **Default Name** value specifies which scene should be used, if a command is received with a register number that has not been configured. If no default scene is specified, then the command will be ignored.

Commands

The PBus command behaviors are detailed below.

Learn Command

If a scene is currently loaded to preview, the **Learn** command will update the register list, setting the register number provided by the command to the scene loaded to preview. If no scene is loaded to preview, then the provided register number will be cleared.

Query Command

This command is unsupported.

Recall Command

If the Cue to Air option is enabled, then the **Recall** command will play the scene associated with the provided register number to Program. Otherwise, the scene will be loaded to Preview.

Trigger Command

Executes a specific function based on a provided Trigger Value:

- 0: Plays the active scene in Preview.
- 1: Pause, unsupported.
- 2: Stops the active scene in Program.
- 3: Reloads the active scene in Program.
- 4: Clears all scenes currently open.
- 5: Record, unsupported.

Write Command

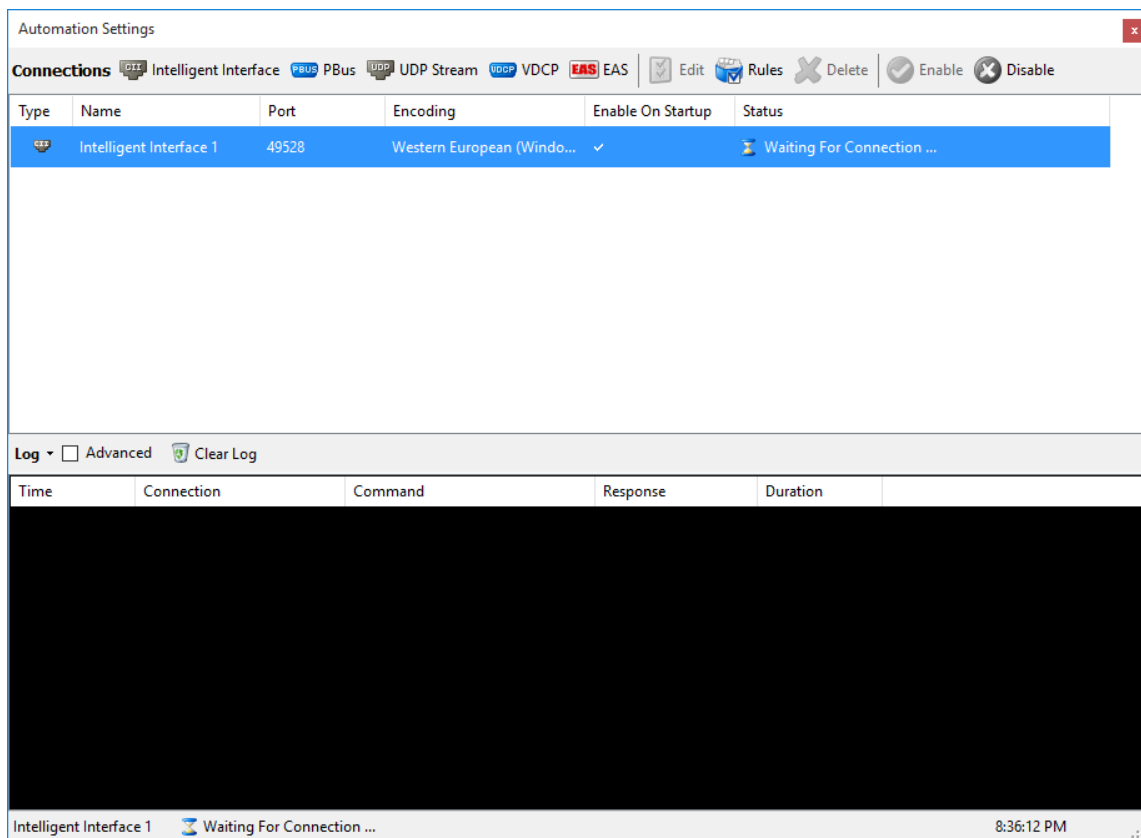
Updates the register list using the provided register number and scene name.

INTELLIGENT INTERFACE

Introduction

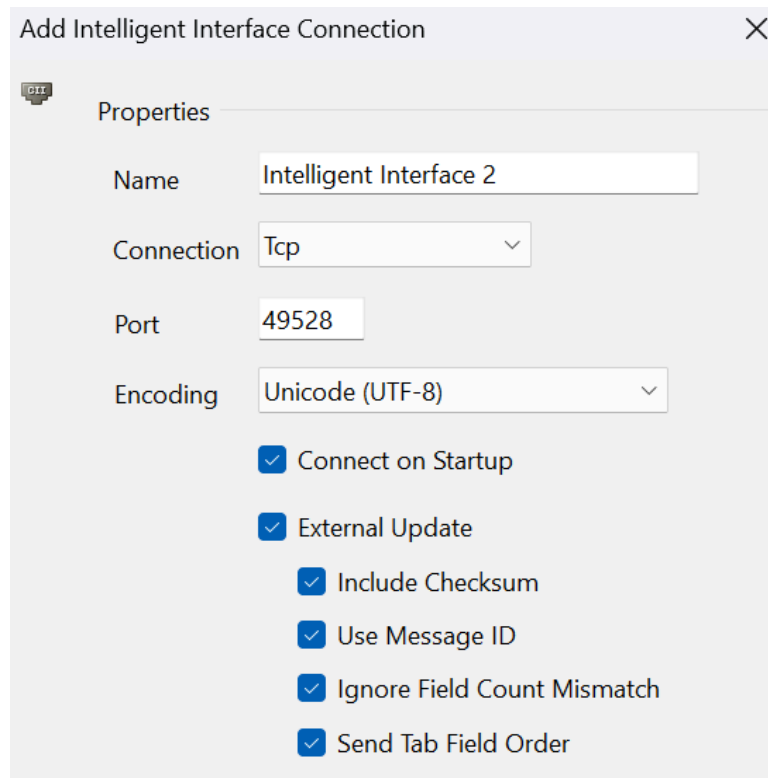
Intelligent Interface is one of the automation protocols used to control playout and query status of a PRIME system. Communication can be over a network or serial connection and may utilize a variety of text encodings.

All automation connections can be viewed in the **Automation Settings** window accessible through the **View** menu of the PRIME application. From this window, a user may add, remove or edit existing automation connections including Intelligent Interface.



Adding a Connection

A new Intelligent Interface connection can be added by clicking the Intelligent Interface button in the Connections toolbar. Once clicked, the following dialog will appear with a variety of configurable options. By default, an Intelligent Interface connection will utilize a network connection (TCP/IP) for data communication and the Western European (ISO-8859-1) command text encoding. To ensure proper command interpretation, the automation system sending commands must be configured with a matching text encoding.



Add Intelligent Interface Connection

Properties

Name: Intelligent Interface 2

Connection: Tcp

Port: 49528

Encoding: Unicode (UTF-8)

☒ Connect on Startup

☒ External Update

☒ Include Checksum

☒ Use Message ID

☒ Ignore Field Count Mismatch

☒ Send Tab Field Order

External Update: Loaded scenes will send and receive X and R Commands respectively. PRIME will construct the X Command when a scene or message is loaded and contains replaceables flagged for External Update. PRIME will send the X Command to the designated II connection. The automation system will respond with an R Command. PRIME will validate the R Command and apply the update to the replaceables marked for External Update.

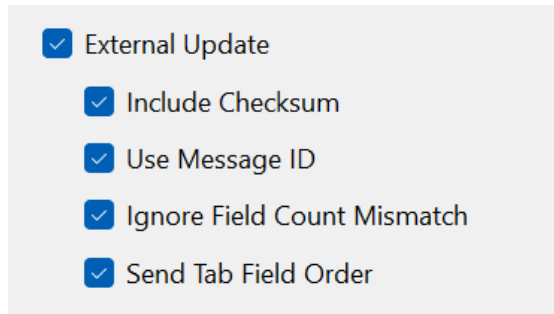
If multiple II connections have External Update checked only the last active connection will send and receive the X and R Commands.

External Update is new to PRIME 3.6.0 and not backwards compatible to earlier versions. If a user runs an earlier version of PRIME, the R Command will be removed from the rules engine. At this point if 3.6.0 is opened again, the R Command will be injected back into the rules engine but rule behavior will be empty. Simply checking External Update will reset the behavior to the

original state of the R Command. Users who have customized rules should always export their rules as an XML.

See External Update in the User Guide for more information.

External Update Options



- ☒ External Update
 - ☒ Include Checksum
 - ☒ Use Message ID
 - ☒ Ignore Field Count Mismatch
 - ☒ Send Tab Field Order

Include Checksum

When checked, sends and expects to receive a 2 character checksum of the command being sent or received

Use Message ID

When checked, sends the Message ID in the third field of the X command instead of using the Scene Name

Ignore Field Count Mismatch

When checked, PRIME will no longer ignore the data if there is a mismatch in data field counts. If fewer fields are received, then the unreceived replaceables will not be updated. If more fields are received, then the extra fields will be unused.

Send Tab Field Order

When this setting is checked, instead of sending out each ID of the replaceables, it will send out the Tab Field Order

Alternatively, by changing the connection type to Serial, communication will occur on one of the system COM ports. Consequently, the available port settings change from that of a numeric port to the full range of COM-specific values.

The screenshot shows a dialog box titled "Add Intelligent Interface Connection". It has a close button (X) in the top right corner. Below the title bar is a "Properties" tab. The settings are as follows:

- Name: Intelligent Interface 2
- Connection: Serial (dropdown)
- Port: (empty dropdown)
- Baud Rate: 9600 (dropdown)
- Data Bits: 8 (dropdown)
- Parity: None (dropdown)
- Stop Bits: One (dropdown)
- Encoding: Unicode (UTF-8) (dropdown)
- Handshake: None (dropdown)
- ☐ DTR Enable
- ☐ RTS Enable
- ☒ Connect on Startup
- ☐ External Update
 - ☐ Include Checksum
 - ☐ Use Message ID
 - ☐ Ignore Field Count Mismatch
 - ☐ Send Tab Field Order

As with the encoding property, there must be parity between the serial port settings configured for the Intelligent Interface connection and the automation system that will be sending commands.

The **Connect on Startup** option indicates whether the Intelligent Interface connection will automatically be enabled when the PRIME application is started. If enabled, PRIME will be ready to receive commands on this connection immediately after launching.

Editing a Connection

Settings for an existing Intelligent Interface connection can be modified by selecting the desired connection and either clicking the **Edit** button in the Connections toolbar or by right-clicking the connection itself and choosing **Edit Connection**.

The Edit option will only be available if the selected connection is not currently enabled.

Deleting a Connection

An existing Intelligent Interface connection may be deleted by selecting the desired connection and either clicking the **Delete** button in the Connections toolbar or by right-clicking the connection itself and choosing **Delete Connection**.

The Delete option will only be available if the selected connection is not currently enabled.

Enabling/Disabling a Connection

The selected Intelligent Interface connection may be enabled or disabled by clicking the **Enable** or **Disable** button in the Connections toolbar or by right-clicking the connection itself and choosing **Enable/Disable**. Which option is available depends on the current state of the connection; e.g. an already enabled connection can only be disabled and an already disabled connection can only be enabled.

Once enabled, the Intelligent Interface status will appear in the *Waiting For Connection* state, indicating that external automation systems may now connect and begin transmitting commands.

Command Processing

Commands are put into 1 of 2 queues. A HIGH Priority queue and a LOWER Priority queue:

- Commands that are defined within the Rule Engine are considered higher priority and are placed on an independent queue.
- Query commands are hard-coded and may not be configured through the rule engine. These are considered lower priority commands and are placed on a separate queue.

Default Heartbeats

The default heartbeat command (consisting only of two back-slashes followed by a carriage return and line-feed) will be processed immediately. ***This heartbeat is never placed on either of the processing queues.*** The default heartbeat and its response can be modified by editing the file:

C:\ChyronHego\Prime\Automation.xml

Each Intelligent Interface command can be configured to have a unique Heartbeat command and response

Add or modify the following nodes to the <IntelligentInterface> section:

```
<DefaultHeartbeatCommand>P\HEARTBEAT\\</DefaultHeartbeatCommand>
<DefaultHeartbeatResponse>#</DefaultHeartbeatResponse>
```

Rule Engine

Each Intelligent Interface connection maintains an independent rule engine. The rule engine provides an end user with greater control over how an automation system interacts with PRIME, allowing a wide range of customization for both well-defined and documented commands as well as any custom commands that might facilitate meeting their needs.

Rule Terminology

The term **behavior** refers to a single operation such as a load, play or transfer scene. Typical automation system interaction with PRIME takes the form of requesting that these behaviors be executed in some sequence; e.g. Load scene 1000, Play scene 1000 and so on. How these behaviors are requested by the automation system depends on the data transmitted to PRIME.

The term **command** refers to a single statement that may be received by an automation system, requesting that one or more behaviors are executed. For example, the default behavior of the P\LOAD command in Intelligent Interface is to load a particular scene to Preview. The command is comprised of a sequence of specific characters that may also include optional or arbitrary parameters.

P\LOAD\1000\\<CR><LF>

There are two components in the above command string.

P\LOAD\1000\\<CR><LF>

The characters in bold correspond to those that define the specific command. All commands that match these particular characters in sequence should request that the same behavior or list of behaviors be executed. What differentiates one instance of the P\LOAD command from another is the unique **parameter**, which is highlighted below.

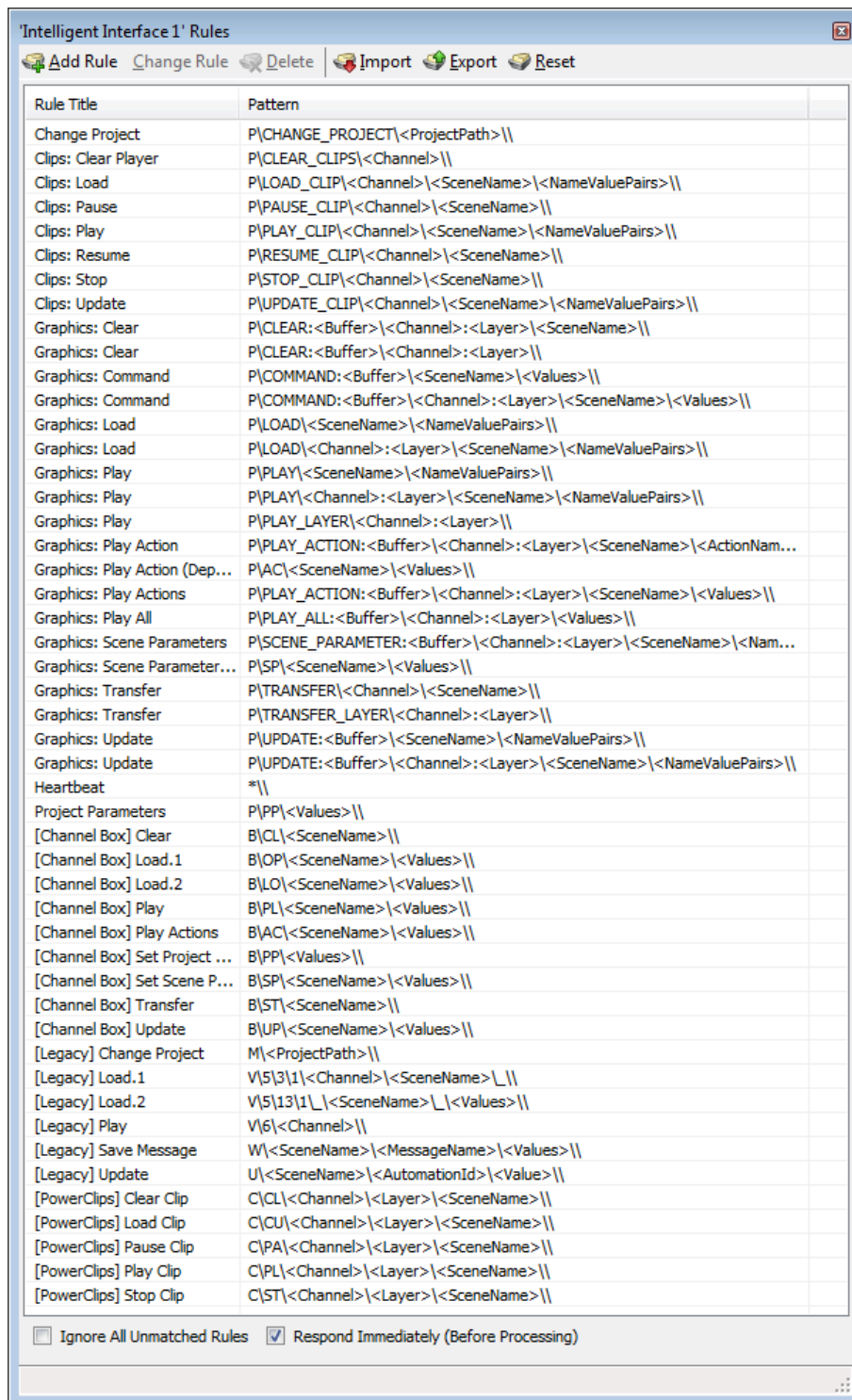
P\LOAD**1000**\\<CR><LF>

Parameter value 1000 is provided as an argument to a load scene request, indicating that scene 1000 should be loaded to preview. Thus a parameter is considered a variable, which may or may not be required as defined by the needs of each of the behaviors that constitute the execution of a particular command. Lastly, a **rule** is simply a means of matching particular command text to a list of one or more behaviors.

Rule Editor

PRIME ships with a default set of rules that define many of the Intelligent Interface commands supported in previous products. The default rule set is utilized whenever a new Intelligent Interface connection is added, but may be customized to fit the needs of a user. Rules may be modified at any time and regardless of whether the connection is currently enabled.

Rules for a selected connection may be edited by either clicking the **Rules** button in the Connections toolbar or by right-clicking the connection and choosing **Customize Rules**. Either action will open a dialog listing the current rule set as seen in the example below.



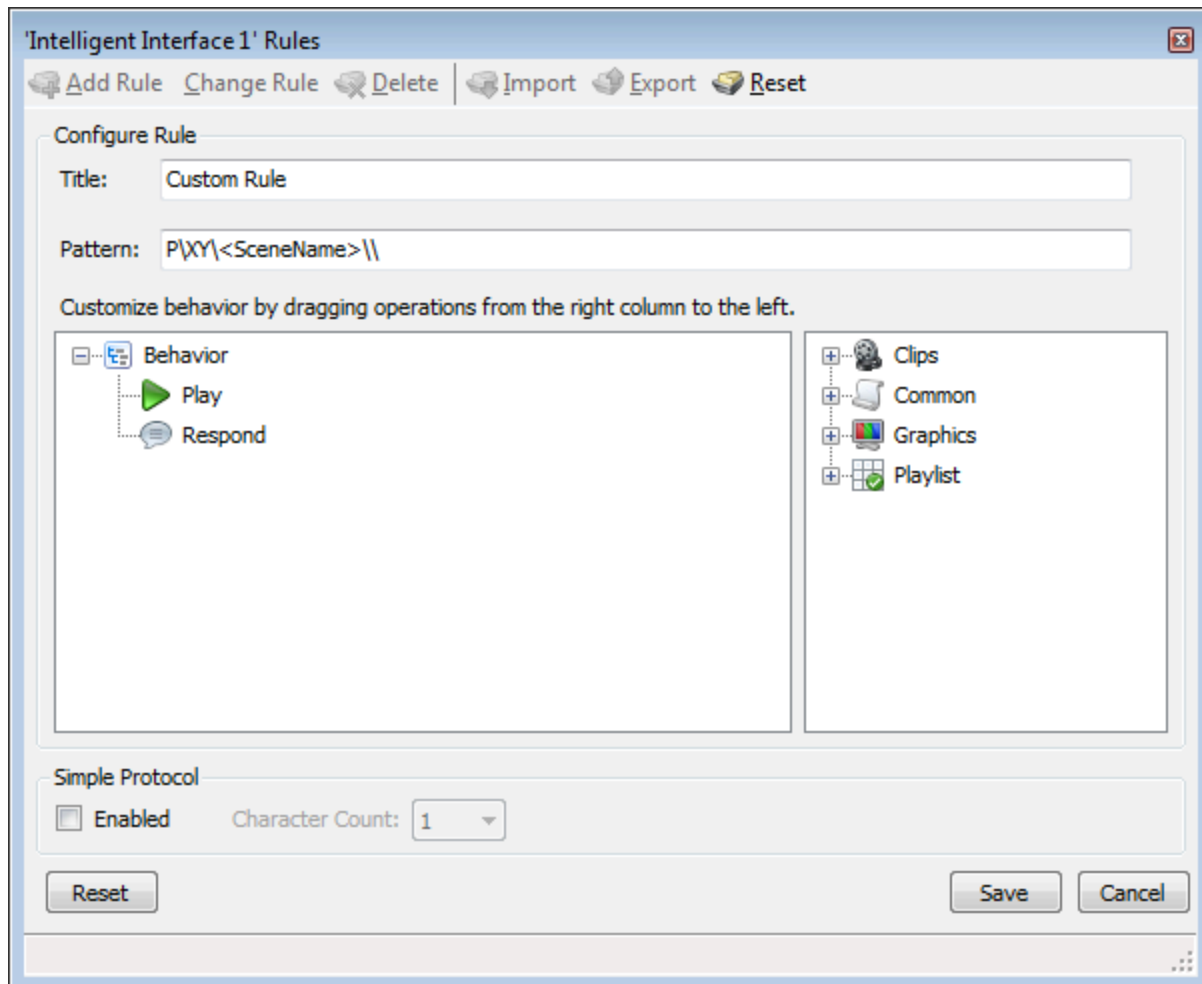
Notice that each rule in the list has a title (description) and a pattern. The pattern is used to match incoming Intelligent Interface command text using a combination of literal characters (such as P\LOAD) and keywords designed as placeholders for parameterized values. It is important to note that patterns are matched from left to right, with the left most character necessarily equivalent to the first character of an incoming command. Patterns are not case-sensitive.

Pattern	Received Command	Successful Match?
P\	P\LOAD\1000\\<CR><LF>	Yes
P\	p\LOAD\1000\\<CR><LF>	Yes (case-insensitive)
P\	p\P\LOAD\1000\\<CR><LF>	No (undefined command P\P\LOAD)

As commands are received from the automation system, PRIME will attempt to match them against the patterns defined in the rule engine. Whenever a rule is matched, the list of behaviors associated with that rule will all be executed in sequence. Two exceptions may result in behaviors not executing, but these will be discussed in the next section.

Behaviors

Double-clicking on one of the rules in the list or clicking the **Change Rule** button will change the view to an editor for the individual rule that had been selected. In this view, the title and pattern are editable as is the list of behaviors that constitute the rule.



Behaviors are divided into different categories based upon their function.

The **Clips** category affects configured clip players.

- **Clear All Clips:** Clears all cued and playing clips from affected clip players.
- **Clear Clip:** Clears a specific clip from a clip player.
- **Clear Clip Player:** Clears a specific clip player.
- **Load Clip:** Loads a clip to a specific clip player.
- **Pause Clip:** Pauses a clip.
- **Play Clip:** Plays a clip to a specific clip player.
- **Play Clip Action:** Plays one or more actions within a PowerClip.
- **Resume Clip:** Resumes a paused clip.

- **Stop Clip**: Stops a playing clip.
- **Update Clip**: Updates a PowerClip with replaceable values.

The **Common** category contains behaviors that are not specific to a channel or clip player.

- **Change Project** - This behavior attempts to change the currently active project. This affects the entire application and is not limited to the scope of the current Intelligent Interface connection.

Required Parameters: <ProjectPath>

- **Ignore** - This behavior causes all subsequent behaviors to be skipped and allows Intelligent Interface to disregard unsupported commands without returning an error.
- **Respond** - Returns a response to the automation system. The user may customize the response returned by double-clicking the Respond behavior when it appears in the behavior list.

Typically, a response will occur after all other behaviors have executed unless the **Respond Immediately** option is enabled in the main rule editor window. If one behavior in the list fails to complete successfully, then subsequent behaviors will not be executed except for the response, which may be configured to send a different value depending upon the success or failure of the previous behaviors.

Alternatively, a single response may always be returned by enabling the **Custom Message** checkbox. Custom responses need not include the trailing backslashes, carriage return or line feed that terminate Intelligent Interface responses as these are added automatically later.

- **Set Project Parameters** – Update one or more project parameter values.

Required Parameters: <Values> or <NameValuePairs>

<Values> will accept data in the form

<Parameter1Name>:<Parameter1Value>\<Parameter2Name>:<Parameter2Value>\

<NameValuePairs> will accept data in the form

<Parameter1Name>\<Parameter1Value>\<Parameter2Name>\<Parameter2Value>\

- **Script:** Executes a local script file or an Application script function.

The **Graphics** category affects configured output channels.

- **Clear** - This behavior clears content from one or more layers. Depending on the parameters provided, this may affect Preview, Program or both buffers on one or more channels.

Optional Parameters: <Channel>, <Buffer>, <Layer>, <SceneName>

- **Load** - This behavior attempts to load a particular scene to Preview. The scene name may be provided as a parameter of the pattern (<SceneName>); otherwise the identifier can be provided by double-clicking the behavior and providing a default.

Optional Parameters: <Channel>, <Buffer>, <Layer>, <SceneName>, <NameValuePairs>

- **Play** – This behavior plays one or more scenes depending upon the parameters provided. Scenes in Preview will be played to Program; other scenes targeted by a received command will played directly to Program from the file system.

Optional Parameters: <Channel>, <Buffer>, <Layer>, <SceneName>, <NameValuePairs>

- **Save Message** - Saves a template data message using provided values and a base scene.

Required Parameters: <SceneName>, <MessageName>, <Values>

- **Set Scene Parameters** – Update one or more scene parameter values.

Required Parameters: <SceneName>, <Values> or <NameValuePairs>

<Values> will accept data in the form

<Parameter1Name>:<Parameter1Value>\<Parameter2Name>:<Parameter2Value>\

<NameValuePairs> will accept data in the form

<Parameter1Name>\<Parameter1Value>\<Parameter2Name>\<Parameter2Value>\

- **Transfer** - This behavior transfers one or more scenes from Program to Preview. Depending on the parameters provided, this may affect multiple channels.

Optional Parameters: <Channel>, <Buffer>, <Layer>, <SceneName>

- **Update** - This behavior attempts to update a particular scene. The scene name may be provided as a parameter of the pattern (<SceneName>); otherwise the identifier of the scene can be provided by double-clicking the behavior and providing a default. Depending on the parameters provided, this may affect multiple instances of the scene in Preview, Program or both buffers on one or more channels.

Optional Parameters: <Channel>, <Buffer>, <Layer>, <SceneName>

The **Playlist** category affects the first configured playlist.

- **Focus Item:** Sets the playlist focus to the item with the specified index.
- **Move Next:** Advances the current selection to the next item within the playlist.
- **Previous Item:** Rewinds the current selection to the previous item within the playlist.
- **Stop Item:** Takes an item in the playlist off-air.
- **Take Item:** Takes an item in the playlist to Program.

The **Sequence** category affects the command sequence within a particular scene.

- **Execute:** Executes the current item; the sequence cursor may change depending on the type of item executed (e.g. action triggers automatically move the cursor, condition triggers must opt to move the cursor explicitly based upon the logic of the condition).
- **Move End:** Advances the current selection to the end of the command sequence.
- **Move First:** Rewinds the current selection to the beginning of the command sequence.
- **Move Last:** Advances the current selection to the last item of the command sequence.
- **Move Next:** Advances the current selection to the next item in the command sequence.
- **Move Next, Execute:** Moves to and executes the next item in the command sequence.

Each unique behavior may only appear in the list once. Parameters mentioned below are documented in the [Supported Parameters](#) section.

Supported Parameters

As a constraint of the Intelligent Interface protocol, all supported parameters must be enclosed with back-slashes in order to be parsed correctly. Parameters may not be mixed without proper delimiters.

- A single under-score enclosed in back-slashes (_\\) may be used to reference numeric fields in Intelligent Interface commands that can be safely ignored. This can be used to match against commands that contain variable numeric fields that may not be useful to PRIME.

```
P\XY\1000\1\\<CR><LF>
```

A rule can then be created to disregard the bolded number above.

```
P\XY\<SceneName>\_\\<CR><LF>
```

- **<ActionName>** may be used to capture an action name that will be used as a parameter to all behaviors that require one.
- **<SceneName>** may be used to capture a scene name that will be used as a parameter to all behaviors that require one.
- **<MessageName>** may be used to capture a message name that will be used as a parameter to all behaviors that require one.
- **<Buffer>** may be used to specify whether Preview, Program, or all buffers should be affected. A command may explicitly provide the text value Preview, Program or AllBuffers; alternatively 0, 1 and * respectively work.
- **<Channel>** may be used to capture the numeric index of the channel that should be affected.
- **<Layer>** may be used to capture a text expression that defines one or more layers to affect.

<LayerExpression> >		Affected Layers
>3		All layers greater than 3
<=3		All layers less than or equal to 3
4		Only layer 4

1-3	Layers 1 through 3 inclusive. Negative values are not supported in this style.
1,3,5	Layers 1, 3 and 5.
*	All layers

- **<ProjectPath>** is a fully qualified project path that has been formatted to be Intelligent Interface compatible.

Actual Project Path: I:\Project\Name

Formatted for Intelligent Interface: I/Project/Name

- **<NameValuePairs>** can be used to reference optional values that will be captured and used as parameters to any behaviors that call for data fields. **Data parsed in this format is assumed to be in the form of <Name>\<Value>**

For example:

P\Update\1000\ABC\DEF\\<CR><LF>

The above command would update an object ABC with value DEF

- **<Values>** is a deprecated parameter used for parsing values in legacy and Channel Box commands. <Values> greedily captures all characters up to the end of the command string or subsequent parameter. Usage of this parameter varies depending on context. Older, Lyric-style commands provide values as a list of backslash separated strings which are mapped onto objects based on automation IDs; the B command, however, expects values as backslash separated strings which are further delimited as name/value pairs. Therefore the Intelligent Interface protocol provides special handling for the B command, but otherwise assumes that values will be mapped by automation ID.

To see the difference, examine the value captured by the <Values> parameter in the following rules:

- o V\5\13_ \<SceneName>_ \<Values>\\
- o B\LO\<SceneName>\<Values>\\

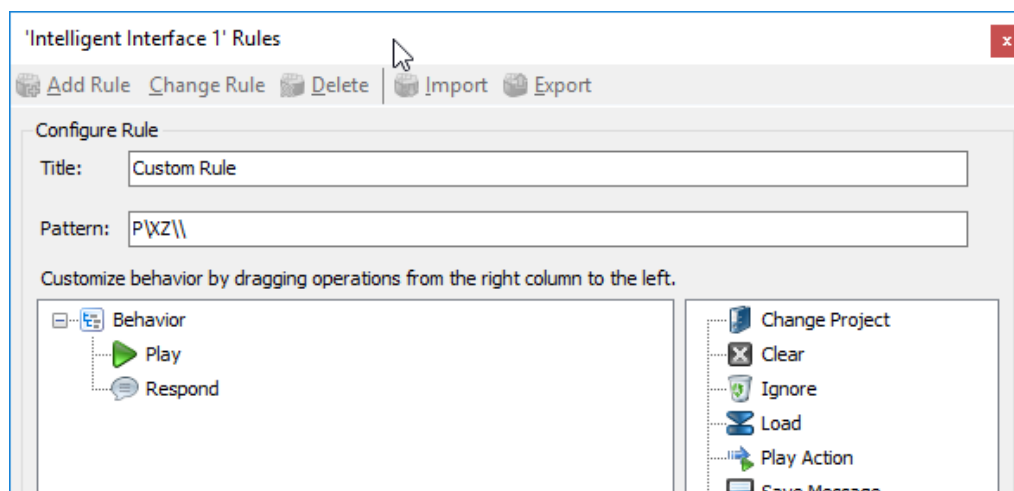
Rule	Received Command	<Values>
------	------------------	----------

V\5\13	V\5\13\1\1000\2\ABC\\<CR><LF>	ABC will be applied to an object with automation ID order 1.
V\5\13	V\5\13\1\1000\2\ABC\DEF\\<CR><LF>	ABC will be applied to an object with automation ID order 1 and DEF will be applied to an object with automation ID order 2
B\LO	B\OP\1000\Text1\$ABC\\<CR><LF>	ABC will be applied to an object named Text1 .
B\LO	B\OP\1000\Text1\$ABC\Text2\$DEF\\<CR><LF>	ABC will be applied to an object named Text1 and DEF will be applied to an object named Text2 .

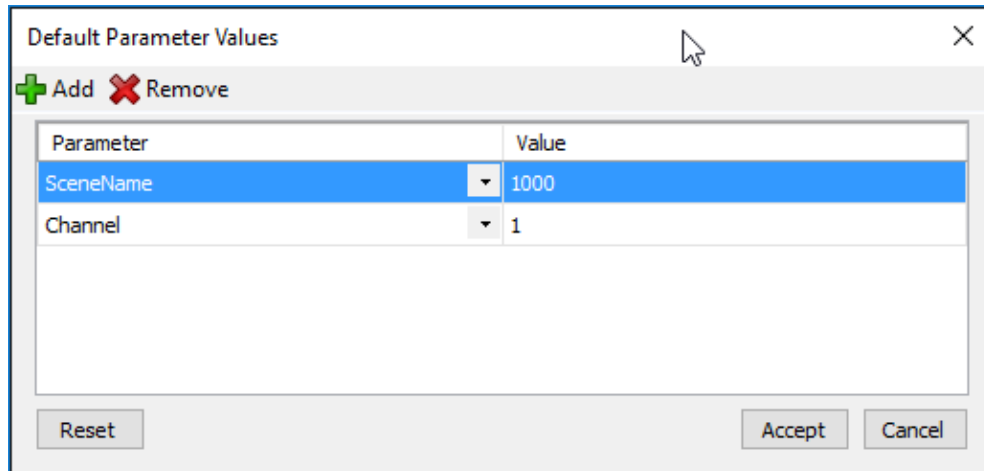
Setting Default Parameter Values

By double-clicking on a behavior in the rule editor, the user may provide a set of default parameter values that will be used by the behavior in the event that the command itself does not supply them.

For example, imagine a new rule as such:



The pattern itself does not allow the command to supply a scene name. The user can double-click the Play behavior and add a default directly.



Now the P\XZ\\ command will affect scene 1000.

Using a single underscore as the default <SceneName> value will allow a command to reference the scene last affected by the Intelligent Interface connection. For example, one command might provide the <SceneName> explicitly while another, assumed to be sent second, does not. By setting the <SceneName> to “_” the second command will implicitly know to reference the scene affected by the first command.

PRIME Runtime Operation

All runtime PRIME operations behave identically regardless of whether they are requested by an automation system or through manual user interaction.

Operation	Result
Load	Loads a scene into Preview.
Play	Plays a scene from Preview into Program or, if the scene isn't in Preview plays directly to Program.
Transfer	If the specified scene was in Program, transfers it back to Preview.
Clear	If the specified scene is in Preview or Program, close it.
Update	Updates one or more objects in a specific scene.

P Commands

The standard P command introduced for PRIME takes the form:

P<COMMAND>:<Buffer>\<Channel>:<Layer>\<SceneName>\<NameValuePairs>

<SceneName> and <ClipName> are synonymous.

Not every command will require all of the above parameters, but in general the above is a good guideline.

Important!

- The <Buffer> and <Layer> parameters are optional and may be excluded completely, including the preceding colon. For example both of the following are equally valid commands, but their meaning changes based upon the provided parameters.
 - o Update scene 1000 on Program, Channel 1, Layer 1
P\UPDATE:Program\1:1\1000\ABC\DEF\\<CR><LF>
 - o Update scene 1000 on Preview or Program, Channel 1, Any Layer.
P\UPDATE\1\1000\ABC\DEF\\<CR><LF>

Command Set

- **Clear Clip Player:** Clears the cued and playing clip from the specified clip player.
 - o P\CLEAR_CLIPS\<Channel>\\<CR><LF>
- **Load Clip:** Loads a clip to the specified clip player.

- o P\LOAD_CLIP\<Channel>\<ClipName>\\<CR><LF>
- **Pause Clip:** Pauses a playing clip on the specified clip player.
 - o P\PAUSE_CLIP\<Channel>\<ClipName>\\<CR><LF>
- **Play Clip:** Play a clip on the specified clip player.
 - o P\PLAY_CLIP\<Channel>\<ClipName>\\<CR><LF>
- **Resume Clip:** Resumes a paused clip on the specified clip player.
 - o P\RESUME_CLIP\<Channel>\<ClipName>\\<CR><LF>
- **Stop Clip:** Stops a playing clip on the specified clip player.
 - o P\STOP_CLIP\<Channel>\<ClipName>\\<CR><LF>
- **Clear Scene(s):** Clears all scenes with the specified name, limited to the scope of the provided buffer, channel and layer expression.
 - o P\CLEAR:<Buffer>\<Channel>:<Layer>\<SceneName>\\<CR><LF>
- **Clear (Override):** Clears all scenes from the specified channel/layer.
 - o P\CLEAR:<Buffer>\<Channel>:<Layer>\\<CR><LF>
- **Change Project:** Switches to the specified project.
 - o P\CHANGE_PROJECT\<ProjectPath>\\<CR><LF>
- **Execute Object Command:** Executes one or more object commands.
 - o P\COMMAND\<Channel>:<Layer>\<SceneName>\<Values>\\<CR><LF>

Each value in the backslash delimited <Values> collection must take the form:

ObjectName.CommandName

For example, Data1.MoveNext or Crawl1.Restart.

- **Execute Sequence Command:** Executes the current item in a scene command sequence.
 - o P\SEQUENCE\<SceneName>\<CR><LF>
 - o P\SEQUENCE\<Channel>\<SceneName>\<CR><LF>
- **Load:** Loads a specific scene to Preview to the default Channel/Layer.
 - o P\LOAD\<SceneName>\<NameValuePairs>\<CR><LF>
- **Load (Override):** Loads a specific scene to Preview, overriding the default Channel/Layer.
 - o P\LOAD\<Channel>:<Layer>\<SceneName>\<NameValuePairs>\<CR><LF>
- **Play:** Plays a specific scene to Program in the default Channel/Layer.
 - o P\PLAY\<SceneName>\<NameValuePairs>\<CR><LF>
- **Play (Override):** Plays a specific scene to Program, overriding the default Channel/Layer.
 - o P\PLAY\<Channel>:<Layer>\<SceneName>\<NameValuePairs>\<CR><LF>
- **Play (Override):** Plays all scenes in the designated layer(s).
 - o P\PLAY_LAYER\<Channel>:<Layer>\<CR><LF>
- **Play All:** Plays one or more scenes.
 - o P\PLAY_ALL:<Buffer>\<Channel>:<Layer>\<Values>\<CR><LF>

Example Usages:

P\PLAY_ALL*\Scene100\Scene200\<CR><LF>
 Play Scene100 and Scene200 to all program channels.

P\PLAY_ALL\1\Scene100\Scene200\<CR><LF>
 Play Scene100 and Scene200 to channel 1.
- **Play Action:** Plays a specific action.
 - o P\PLAY_ACTION:<Buffer>\<Channel>:<Layer>\<SceneName>\<ActionName>\<CR><LF>

Example Usage:

P\PLAY_ACTION\1\Scene100\Action 1\<CR><LF>
 Play Action 1 on Scene 100 on channel 1.

- **Play Actions:** Plays one or more actions.

- o P\PLAY_ACTION:<Buffer>\<Channel>:<Layer>\<SceneName>\<ActionNames >\

Example Usage:

P\PLAY_ACTION\1\Scene100\Action 1\Action 2\\
Play Action 1 and Action 2 to Scene100 on channel 1.

- **Set Scene Parameters:** Updates one or more scene parameter values.

- o P\SCENE_PARAMETER:<Buffer>\<Channel>:<Layer>\<SceneName>\<NameValuePairs>\

Example Usage:

P\SCENE_PARAMETER\1\Scene100\Parameter 1\Value 1\\
Applies Value 1 to Parameter 1 in Scene 100 on channel 1.

- **Transfer:** Transfers a specific scene from Program to Preview.

- o P\TRANSFER\<Channel>\<SceneName>\\<CR><LF>

- **Transfer (Override):** Transfers all scenes in the designated layer(s).

- o P\TRANSFER_LAYER\<Channel>:<Layer>\\<CR><LF>

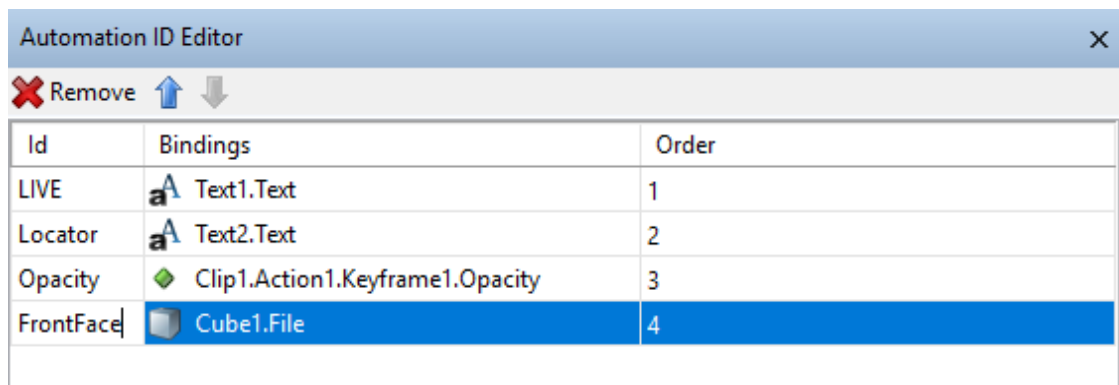
- **Update:** Updates a specific scene with new values.





- o P\UPDATE:<Buffer>\<SceneName>\<NameValuePairs>\

- **Update (Override):** Updates a specific scene with new values, limited to the scope of the provided buffer, channel and layer expression.

Update commands can target 1 of 3 items

1.) The automation ID from the Automation ID List



Id	Bindings	Order
LIVE	 Text1.Text	1
Locator	 Text2.Text	2
Opacity	 Clip1.Action1.Keyframe1.Opacity	3
FrontFace	 Cube1.File	4

2.) The Control Panel Name

3.) The Scene Object Name

Whereas the ID is the Name in the Update command NameValuePair

- o P\UPDATE:<Buffer>\<Channel>:<Layer>\<SceneName>\<NameValuePair>\
- **Play Action (Deprecated):** Play a list of actions for a specific scene on Program of the currently active channel.
 - o P\AC\<SceneName>\<ActionNames>\<CR><LF>

<ActionNames> is a backslash delimited list of action names.

Example: P\AC\1000\Action1\Action2\<CR><LF>
- **Update Scene Parameters (Deprecated):** Sets the value of a scene parameter.
 - o P\SP\<SceneName>\<ParameterName>:<ParameterValue>\<CR><LF>
- **Update Project Parameters:** Sets the value of a project parameter.
 - o P\PP\<ParameterName>:<ParameterValue>\<CR><LF>

B Command Set (Query)

1. **Query addressable control panel:** Returns a list of control names in the specified scene.
 - B\AN\<SceneName>\<CR><LF>
2. **Query the current value of an object**
 - B\AV\<SceneName>\<ObjectName>\<CR><LF>
3. **Query the state of a particular scene.**
 - B\QS\<SceneName>\<CR><LF>

Supported states include: *Closed*, *Loaded*, *Playing*, *Stopped* and *NonExistent*.
4. **Query all scenes that match a particular state**
 - B\QL\<SceneState>\<CR><LF>
5. **Retrieve the scene thumbnail as a base-64 encoded string**
 - B\TH\<SceneName>\<CR><LF>
6. **Retrieve a list of names of scenes in the current project**
 - B\SL\<CR><LF>

7. Retrieve the asset list for a particular scene

- B\SA\<SceneName>\\<CR><LF>

8. Retrieve the XML for a particular scene

- B\SX\<SceneName>\\<CR><LF>

9. Retrieve the XML for a particular message

- B\MX\<MessageName>\\<CR><LF>

10. Capture output of the currently active Program channel and save it to a file in the <FolderPath> folder. Prime automatically assigns a timestamp as the filename.

- B\CO\<FolderPath>\\<CR><LF>

11. Captures the current screen (Windows) and saves it to a file in the <FolderPath> folder. Prime automatically assigns a timestamp as the filename.

- B\CS\<FolderPath>\\<CR><LF>

12. Returns system diagnostic information

- B\SI\\<CR><LF>

13. Query if a scene or message exists

- B\QE\<Name>\\<CR><LF>

Message Command Set

1. **Save Message:** Updates or creates a template data message for a given scene using provided values. A message file will be created or modified as necessary.

- W\<MessageName>\<SceneName>\<Values>\<CR>\<LF>

W commands can also target other messages (instead of scenes) to create new messages. This allows operators to extend single scenes in various ways by enabling and disabling different replaceables, then targeting these messages with W commands to create new extended messages

- W\<NewMessageName>\<ExistingMessageName>\<Values>\

Deprecated Command Set

1. **Change Project:** Switches to the specified project.

- M\<ProjectPath>\<CR>\<LF>

2. **Update:** Updates a specific scene with values from this command.

- U\<SceneName>\<AutomationId>\<Value>\<CR>\<LF>

3. **Load:** Loads a specific scene to Preview of the currently active channel. This command exists for backward compatibility with Channel Box.

- V\5\13\1\<ChannelNumber>\<SceneName>_ \<Values>\<CR>\<LF>
- V\5\3\1\<ChannelNumber>\<SceneName>_ \<CR>\<LF>

4. **Take:** If a scene is in Preview, take it to Program. Otherwise, play to the currently active channel.

- V\6\<ChannelNumber>\<CR>\<LF>

Checksum Calculation

Checksums are used to verify the integrity of commands that are sent to the system, and appears as checksum in the commands shown in this manual. The checksum is calculated by adding the ASCII value of all characters in the command string starting at the first character and ending at the second backslash. The total is computed as modulo 256 and displayed as a 2-byte hex value in ASCII. A response <0D><0A> indicates that the command has been accepted with no checksum error

Error Codes

Most commands will return success in the form of a single asterisk, however failure is instead denoted using one of the error codes listed below.

Error Code	Description
000040B3	The requested asset could not be found.
00004190	The command is invalid.
00004191	The command is malformed.
00004192	An unknown error has occurred.
000041D4	Invalid automation ID. <i>(Deprecated B commands only)</i>
00005501	Error loading the scene. <i>(Deprecated B commands only)</i>
00005502	Error opening the scene. <i>(Deprecated B commands only)</i>
00005503	Error initiating play out. <i>(Deprecated B commands only)</i>
00005504	Error stopping play out. <i>(Deprecated B commands only)</i>
00005505	Error closing the scene. <i>(Deprecated B commands only)</i>
00005506	Unable to update an object. <i>(Deprecated B commands only)</i>
00005507	Error executing a query. <i>(Deprecated B commands only)</i>

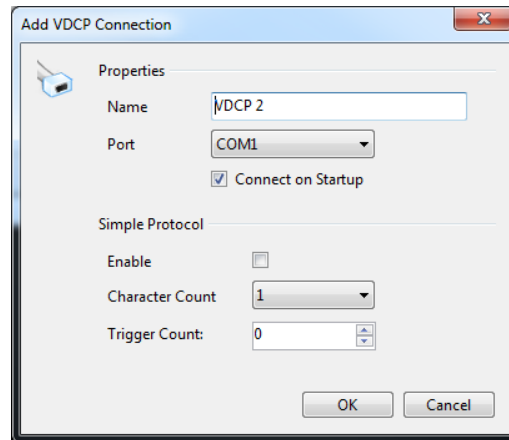
VDCP

PRIME can be controlled via external automation using the VDCP protocol.

Configuration

To enable PRIME to receive VDCP commands, a VDCP connection must be added. To do so:

- Click the **VDCP** icon at the top of the Automation Configuration Panel. The **Add VDCP Connection** panel is displayed.



The following **VDCP Properties** settings may be configured:

- **Name** - Enter a name for the VDCP connection.
- **Port** - From the Port dropdown, select the port that will receive the VDCP commands.
- **Connect on Startup** - Check the **Connect on Startup** checkbox to enable automatic connect to the automation system on PRIME startup.
- **Baud rate** - is specified at 38400 by the VDCP Specification. No other baud rates supported

Electrical and Mechanical Specifications¹

1. Communications Signal
 - a. Asynchronous bit serial, word serial
 - b. Conforms to EIA RS-422A
 - c. Full duplex communications channel
 - d. Transfer rate: 38.4 kb/s
2. Bit Configuration
 - a. 1 start bit (space)
 - b. 8 data bits
 - c. 1 parity bit (odd)
 - d. 1 stop bit (mark)
 - e. Byte time = .286 msec.
3. Connection (9 Pin D-subminiature)

	PIN	CONTROLLING DEVICE	CONTROLLED DEVICE
1.	1	Frame Ground	Frame Ground
2.	2	Receive A	Transmit A
3.	3	Transmit B	Receive B
4.	4	Transmit Common	Receive Common
5.	5	Spare	Spare
6.	6	Receive Common	Transmit Common
7.	7	Receive B	Transmit B
8.	8	Transmit A	Receive A
9.	9	Frame Ground	Frame Ground

The Following **Simple Protocol** settings may be configured:

- **Enable** - Check the Simple Protocol check box to enable the use of Simple VDCP Protocol. Simple Protocol is a mechanism through which commands can trigger clips. Commands that take clip ID's as parameters may have a proprietary format when Simple Protocol is enabled.
- **Character Count** - With Simple Protocol enabled a clip ID is split into two distinct identifiers. The bottom X characters, as defined by selecting a number in the Character Count drop-down, refer to an automation ID, i.e., the clip to be triggered, while the remaining characters comprise the actual clip ID.
- **Trigger Count** - Use the spin box to determine the trigger count.

Supported Commands

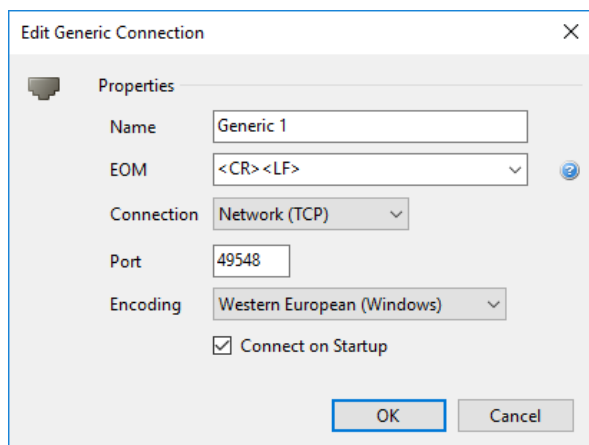
- Active Id
- Clear:
 - *Clips: Clears channel*
 - *Graphics: Close all playing scenes*
- Close Port
- Continue
 - *Clips: Resume a paused clip*
- Cue With Data
- Id List
- Id Request
- Id Size
- Ids Added
- Ids Deleted
- Jog
 - *Clips only*
- Next
- Open Port
- Play
- Play Cue
- Port Status
- Position Request
- Select Port
- Sort Mode
- Still
- Stop
- System Status Request
- Variable Play
 - *Clips only*

GENERIC

The Generic automation connection allows for basic string of any format to be sent to PRIME via Serial, TCP or UDP. The default commands that PRIME supports are the “Ross Talk” commands.

This connection requires all commands to be manually added to the Rules Engine.

Users must define an “End of Message” character(s)



Edit Generic Connection

Properties

Name: Generic 1

EOM: <CR><LF>

Connection: Network (TCP)

Port: 49548

Encoding: Western European (Windows)

☒ Connect on Startup

OK Cancel

i Format

The custom header/footer supports common non-printing codes, hex values, and plain text. Codes and hex values need to be surrounded in a tag (angle brackets < and >) to be interpreted correctly. Plain text can be written anywhere and do not require tags. If a tag is not recognizable, it will be left untouched. If necessary, angle brackets can be escaped with a leading backslash (\<).

Code examples:

<LF> will be replaced with the line feed character
<TAB> will be replaced with the tab character
<EOT> will be replaced with the end of transmission character

Hex examples:

<A> will be replaced with the line feed character
<41> will be replaced with the A character
<7D> will be replaced with the } character

Any string matching will execute the behaviors associated with the rule. The “Default” rules are based off the Ross Talk protocol:

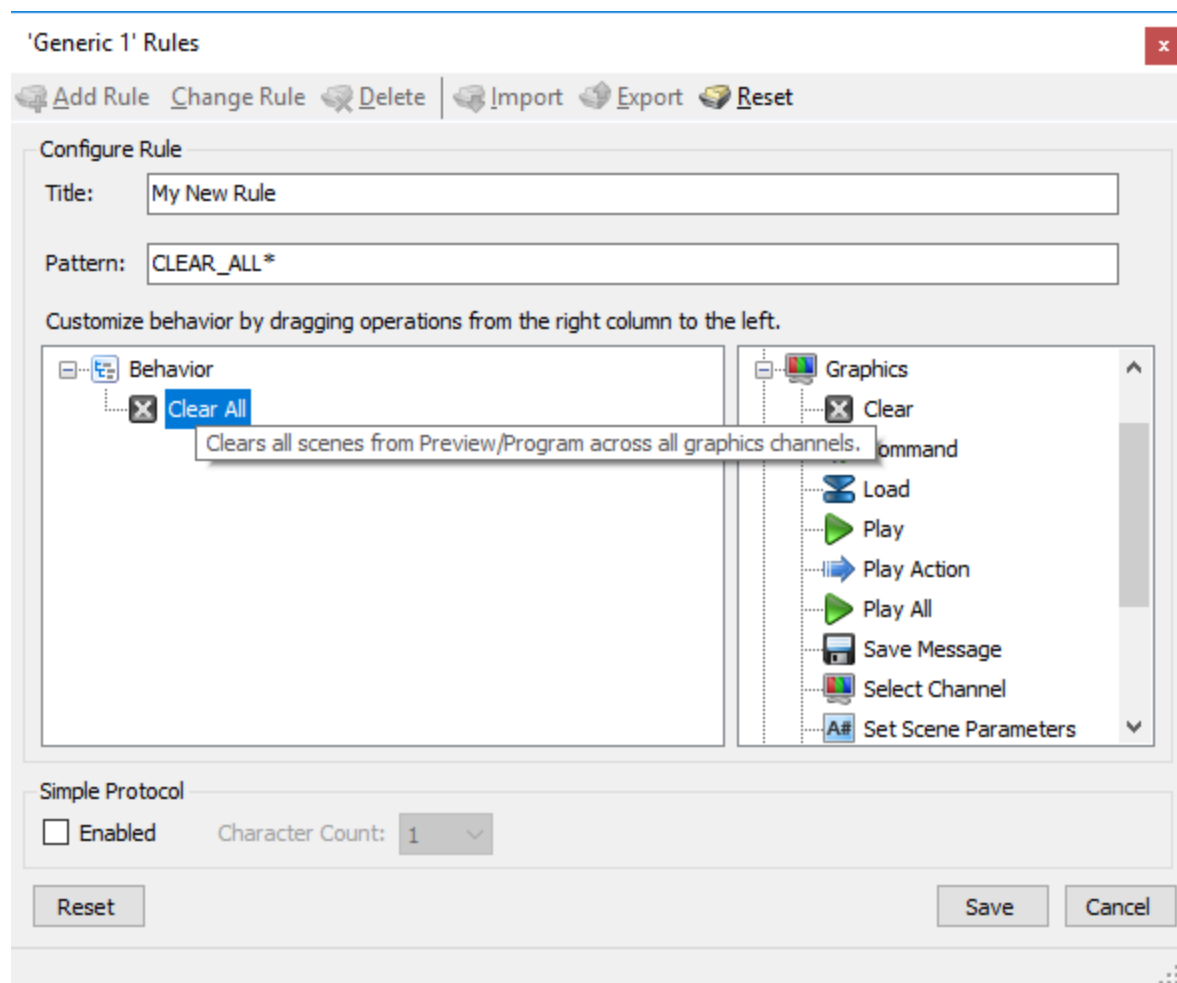
Generic 1' Rules

Add Rule
 Change Rule
 Delete
 Import
 Export
 Reset

Rule Title	Pattern
Clear	CLFB <Channel>:<Layer>
Clear All	CLRA
Focus Item	FOCUS <TakeId>
Move Next	DOWN
Move Previous	UP
Next	NEXT
Read	READ
Sequence Stop	SEQO <TakeId>
Sequence Take	SEKI <TakeId>:<Layer>
Take	TAKE <TakeId>:<Channel>:<Layer>

☐ Ignore All Unmatched Rules
 ☐ Respond Immediately (Before Processing)

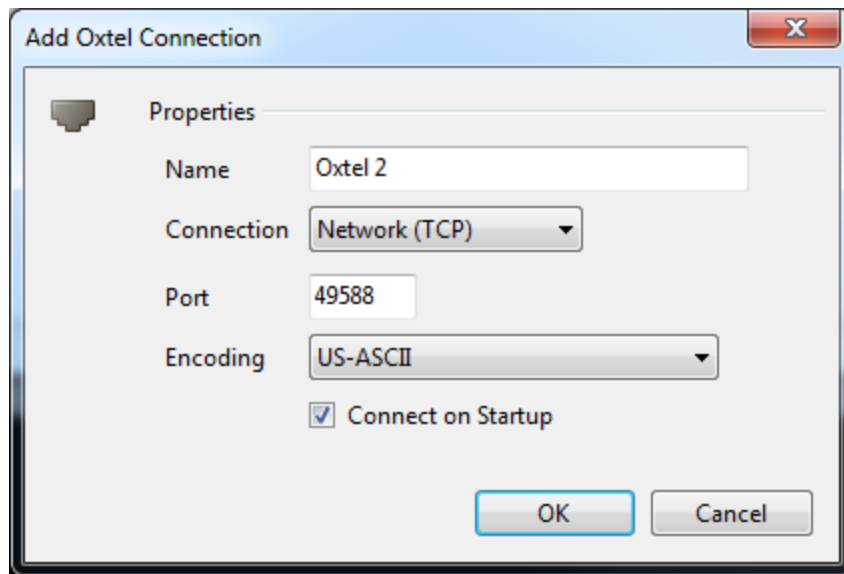
Users can create as many rules as they need in any format. Define the pattern and then apply Behaviors. In this example if the text “CLEAR_ALL*” is sent followed by the command delimiter <CR><LF> PRIME will clear all outputs.



OxTEL

PRIME can be controlled via a subset of the IntuitionXG commands supported under the Oxtel protocol. To receive IntuitionXG commands, an Oxtel connection must first be added. To do so:

- Click the **Oxtel** icon at the top of the Automation Configuration Panel. The **Add Oxtel Connection** panel is displayed.



Oxtel commands differ slightly when communicating over serial or TCP. Commands sent over serial port will start with a header (STX0 <02> or STX1 <03>) and end with a two-byte CRC that follow the normal terminator character (a single colon). Furthermore, Oxtel commands received over serial will result in either an ACK or NAK response.

Sample command over serial:

<STX0><CommandText>:<CRC Low><CRC High>

Network based communication is missing both the header and CRC; no response is returned.

Sample command over TCP:

<CommandText>:

Command Support

The following IntuitionXG commands are currently supported:

- **V0 (Cue)**

V0 <Channel>|<IDType>|<SceneName>:

<Channel> is either the name or index of an output channel.

<IDType> is an unsupported integer parameter; a value must be specified (0, 1, 2) however it currently has no effect.

<SceneName> is either a scene name or file path.

- **V0 1|1|500:**

Cue scene 500 on channel 1.

- **V1 (Take)**

V1 <Channels>|<TakeNumber>:

<Channels> is a comma-separated list of output channel names or indices.

<TakeNumber> is an unsupported integer parameter; it currently has no effect.

- **V1 1|0:**

Play any cued scenes on channel 1.

- **V1 1,2|0:**

Play any cued scenes on channel 1 or channel 2.

- **V2 (Clear)**

V2 <Channels>:

<Channels> is a comma-separated list of output channel names or indices.

- **V2 1:**

Close any playing scenes on channel 1.

- **V2 1,2:**

Close any playing scenes on channel 1 or channel 2.

- **V5 (Set Property)**

V5 <ObjectNames>|<ObjectValues>|<Channels>|<Buffer>:

<ObjectNames> is a comma-separated list of object and property names used to reference targetable items in a scene.

<ObjectValues> is a comma-separated list of values that will be applied to the corresponding items in the <ObjectNames> list.

<Channels> is a comma-separated list of output channel names or indices.

<Buffer> is an optional integer parameter used to limit the updates to either Preview (0) or Program (1, default).

- **V5 Text1.Text|Sample|1|1:**
Update Text1.Text with the value 'Sample' in any scene playing on channel 1 that has such an object.
- **V5 Text1.Text|Sample|1|0:**
Update Text1.Text with the value 'Sample' in any scene cued on channel 1 that has such an object.
- **V5 Text1.Text,Text2.Text|ABC,DEF|1|1:**
Update Text1.Text with the value 'ABC' and Text2.Text with the value 'DEF' in any scene playing on channel 1 that has either object.

UDP STREAM

PRIME's Automation component can have multiple UDP listeners.

The screenshot shows the 'Add UDP Stream Connection' dialog box. It contains the following fields and options:

- Name:** UDP Stream 1
- Port:** 49528
- Encoding:** Western European (Windows)
- Connect on Startup:** ☒
- Custom Data Stream:** ☐
 - Header (SOM):** [Dropdown menu]
 - Footer (EOM):** [Dropdown menu]
- Extract Parameter Values:** ☐
 - Delimiter:** [Text input field]
 - Parameter:** [Text input field]

Buttons: OK, Cancel

The UDP listener can accept command data in the following format:

`P\[parameter name]:[parameter value]\[parameter name]:[parameter value]\\`

This command drives PRIME's parameters, which are further detailed in the [PRIME General User Guide](#).

However, a custom data format can also be specified by using the Header (SOM) and Footer (EOM) options to define the start and end of a data packet. For example, by enabling the Custom Data Stream option and then entering the '[' character as the Header and the ']' character as the Footer, this connection will accept any data message with the following format '[SomeDataHere]' and update two project parameters accordingly.

Parameters created/updated (UDP Stream 1 is the name of the connection in this example):

- **UDP Stream 1 [Raw]** => Byte array with all of the data of the custom message
- **UDP Stream 1 [Text]** => Text representation of the custom message

These project parameters can then be leveraged elsewhere (e.g. through scripting, etc.). The user may also specify that the data between the header and footer contains delimited parameter values by enabling the Extract Parameter Values option.

Once enabled, a delimiter and base parameter name must be defined. The delimiter specifies what character(s) separate each of the incoming values. The parameter property defines a base name for project parameters which will be created for the incoming data.

For example:

Delimiter: Semi-colon

Parameter Name: Sample

Data received:

<SOM>A;B;C;D</EOM>

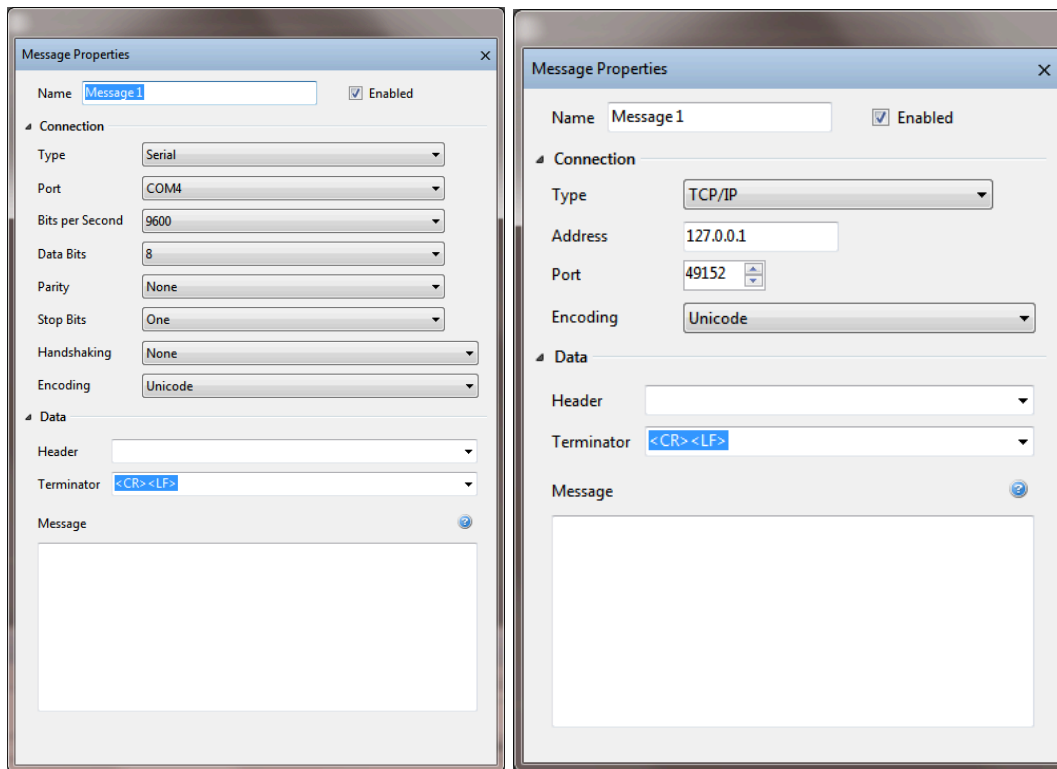
The above command would set 4 project parameters:

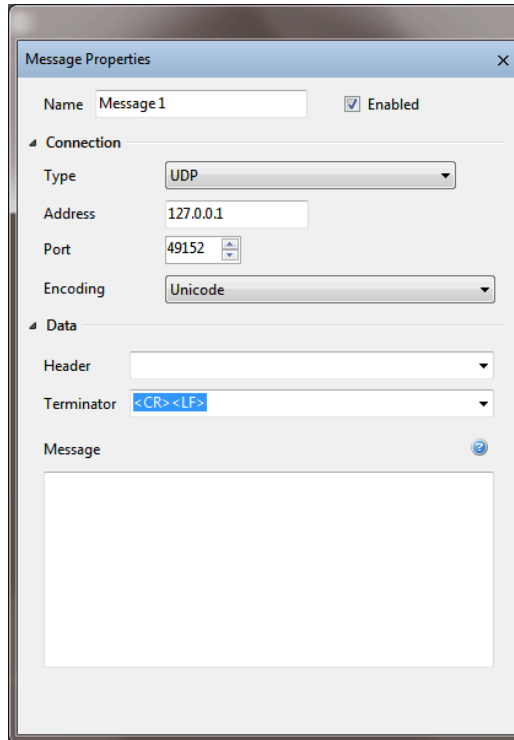
- Sample1 => A
- Sample2 => B
- Sample3 => C
- Sample4 => D

MESSAGE OBJECT

The message object allows you to output messages via Serial port or by Network (Via TCP or UDP).

The “Message” property may be “key framed” in an Action.





The message property supports common non-printing codes, hex values, and plain text. Codes and hex values need to be surrounded in a tag (angle brackets < and >) to be interpreted correctly. Plain text can be written anywhere and do not require tags. If a tag is not recognizable, it will be left untouched. If necessary, angle brackets can be escaped with a leading backslash (\<).

Code examples:

- <LF> will be replaced with the line feed character
- <TAB> will be replaced with the tab character
- <EOT> will be replaced with the end of transmission character

Hex examples:

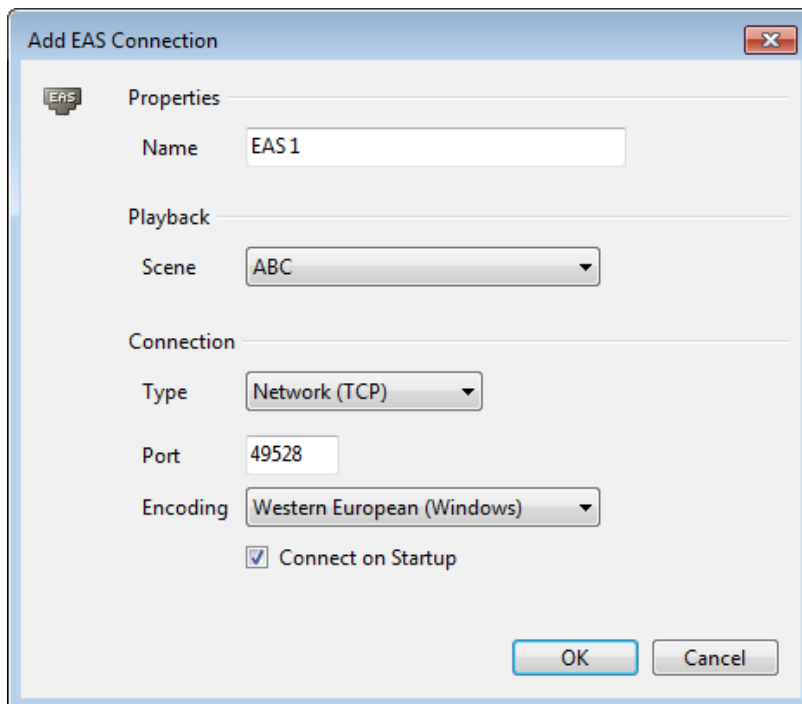
- <A> will be replaced with the line feed character
- <41> will be replaced with the A character
- <7D> will be replaced with the } character

Message formatting also applies to the Header and Terminator properties found in the screenshots above.

EAS

EAS support is currently limited to two CODI commands which affect a single scene as defined in the connection configuration window:

- **Message Effect Crawl Setup** (\ME\CS):
Loads the designated scene.
- **Message Effect Crawl Begin** (\ME\CB)
Plays the designated scene.



Any arguments provided by these commands will be stored within project parameters:

- EAS Argument Data
- EAS Argument Speed
- EAS Argument Iterations
- EAS Argument YAddress
- EAS Argument YHeight
- EAS Argument XStart
- EAS Argument Width

These parameters should be referenced by the selected scene.

ABOUT US

Chyron is ushering in the next generation of storytelling in the digital age. Founded in 1966, the company pioneered broadcast titling and graphics systems. With a strong foundation built on over 50 years of innovation and efficiency, the name Chyron is synonymous with broadcast graphics. Chyron continues that legacy as a global leader focused on customer-centric broadcast solutions. Today, the company offers production professionals the industry's most comprehensive software portfolio for designing, sharing, and playing live graphics to air with ease. Chyron products are increasingly deployed to empower OTA & OTT workflows and deliver richer, more immersive experiences for audiences and sports fans in the arena, at home, or on the go.

CONTACT SALES

EMEA • North America • Latin America • Asia/Pacific
+1.631.845.2000 • sales@chyron.com

