Umetrix® Video

GED Remote Control User Guide

Copyright

Page Part Number: 71-008534, Version A2

Table of Contents

1. Introduction

Overview

This guide describes how to use Umetrix® Video GED Remote Control. Umetrix Video GED Remote Control is an optional command set for controlling Umetrix Video GED tests remotely via any socket client. Remote functionality includes:

- synchronizing time
- configuring capture channels
- starting/stopping a capture session
- performing post-processing on a session

Access the Latest Documentation

To access the latest Umetrix Video documentation, perform the following steps:

1. Log into the Spirent Customer Service Center website (**http://support.spirent.com**) using the email address and password assigned to you by Spirent.
2. In the Search Knowledge Base box, enter ***DOC1151*** and click on **Search KB**.

   The Spirent Umetrix® Video (Chromatic) Documentation page appears.
3. In the Documentation section, click on the link for the document in which you are interested.

   The page for the selected document appears.
4. Click on the link in the Attachment area to view the corresponding PDF.

How to Contact Us

To obtain technical support for any Spirent Communications product, please contact our Support Services department using any of the following methods:

Americas
E-mail: support@spirent.com
Web: http://support.spirent.com
Toll Free: +1 800-SPIRENT (+1 800-774-7368) (North America)
Phone: +1 818-676-2616
Hours: Monday through Friday, 05:30 to 18:00 Pacific Time

Europe, Africa, Middle East
E-mail: support@spirent.com
Web: http://support.spirent.com
Phone: +33 (1) 6137 2270 (France)
Phone: +44 1803 546333 (UK)
Hours: Monday through Thursday, 09:00 to 18:00, 9:00 to 17:00 Friday, Paris Time

Asia Pacific
E-mail: support@spirent.com
Web: http://support.spirent.com
Phone: +86 (800) 810-9529 (toll-free mainland China only)
Phone: +86 (10) 8233 0033 (China)
Operating Hours: Monday through Friday, 09:00 to 18:00 Beijing Time

The Spirent Knowledge Base (**http://support.spirent.com**) is designed to serve your technical information needs. The Knowledge Base gives you access to tens of thousands of documents that help answer your network analysis and measurement questions. New content is added daily by Spirent's communications and networking experts. Sign in with your user ID and password to gain access to additional content that is available only to customers – user manuals, Help files, release notes, Tech Bulletins, and more. When you sign in, you can also use the Knowledge Base to download software and firmware, and to manage your SRs.

Information about Spirent Communications and its products and services can be found on the main company website at **http://www.spirent.com**.

Company Address
Spirent Communications, Inc.
26750 Agoura Road
Calabasas, CA 91302
USA

Document Conventions

This document uses the following conventions:

- Text you type appears ***in this type style***
- Keyboard keys are displayed IN THIS TYPE STYLE

2. Accessing Umetrix Video GED Remote Control

Overview

You can use either the HyperTerminal utility or a Telnet client application such as PuTTY to access Umetrix Video GED Remote Control. PuTTY is a free, downloadable program.

Prerequisites

To access Umetrix Video GED Remote Control, you must have either HyperTerminal utility or a Telnet client application such as PuTTY installed on your PC.

Configure Umetrix Video for GED Remote Control

Perform the following steps to download the Umetrix Data PC client software:

1. In the Umetrix Video window, click on the Settings button.

   The Settings page appears.

2. Click on **Gross Error Detection** at the top of the page.

   The **Enable remote control via TCP socket 7073** check box appears in the Application Startup area as shown in the following figure.

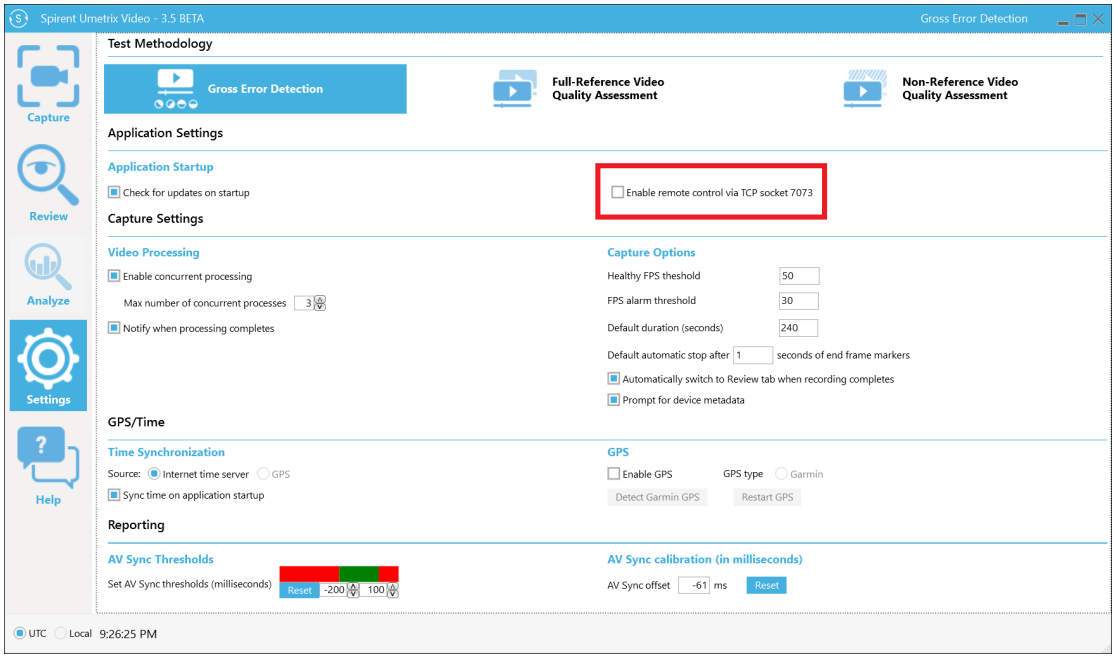

Figure 1. Location of Enable remote control via TCP socket 7073 check box.

3. Check (select) the **Enable remote control via TCP socket 7073** check box.
4. If Microsoft Windows or your security software attempt to block this request, “unblock” or allow access to the program.

Access Umetrix Video GED Remote Control Using PuTTY

Before performing this procedure, make sure PuTTY is installed on your PC.

Perform the following steps:

1. Start PuTTY.

   The PuTTY Configuration dialog box appears.

2. Configure the Host Name and Saved Sessions settings as shown in the following figure.

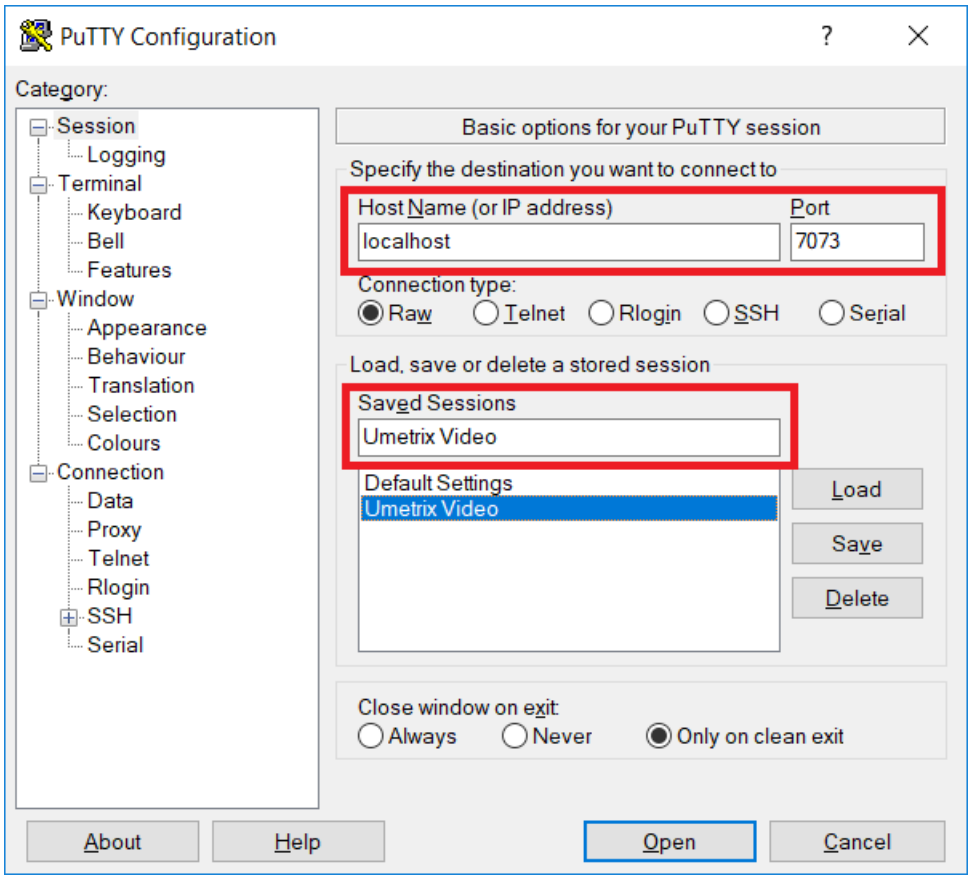

Figure 2. Configured setting in the PuTTY Configuration dialog box .

3. Click the **Save** button.
4. Click the **Open** button.

   When PuTTY successfully establishes a connection, the following PuTTY window appears and displays a welcome message.

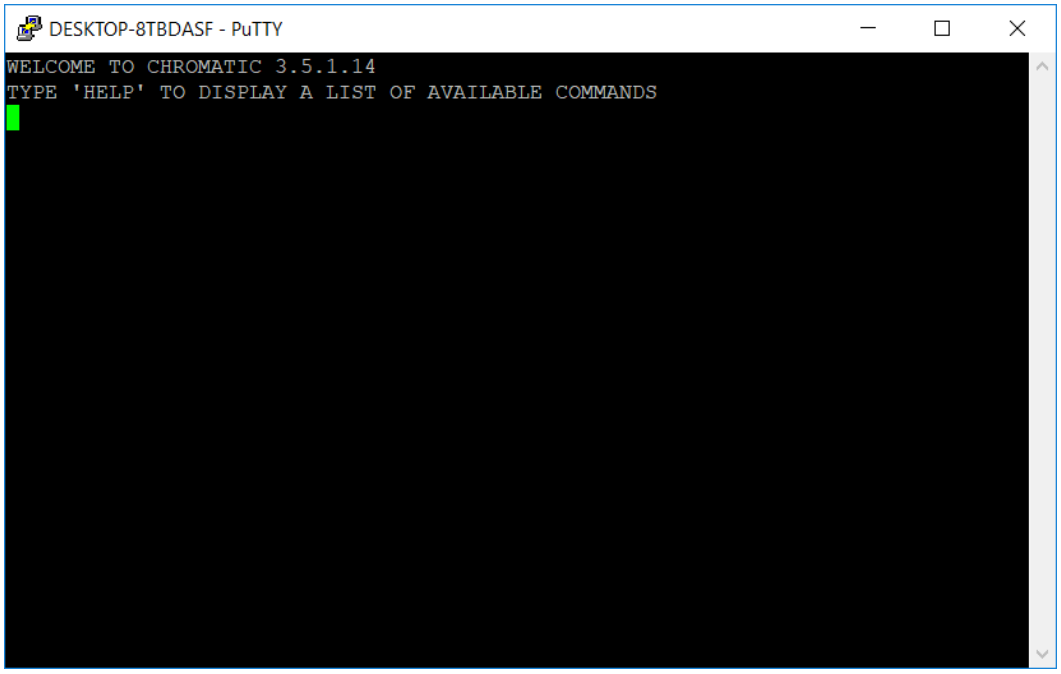

Figure 3. Welcome message in PuTTY window.

5. Type ***HELP*** and press the ENTER key.

   The list of Umetrix Video commands is displayed.

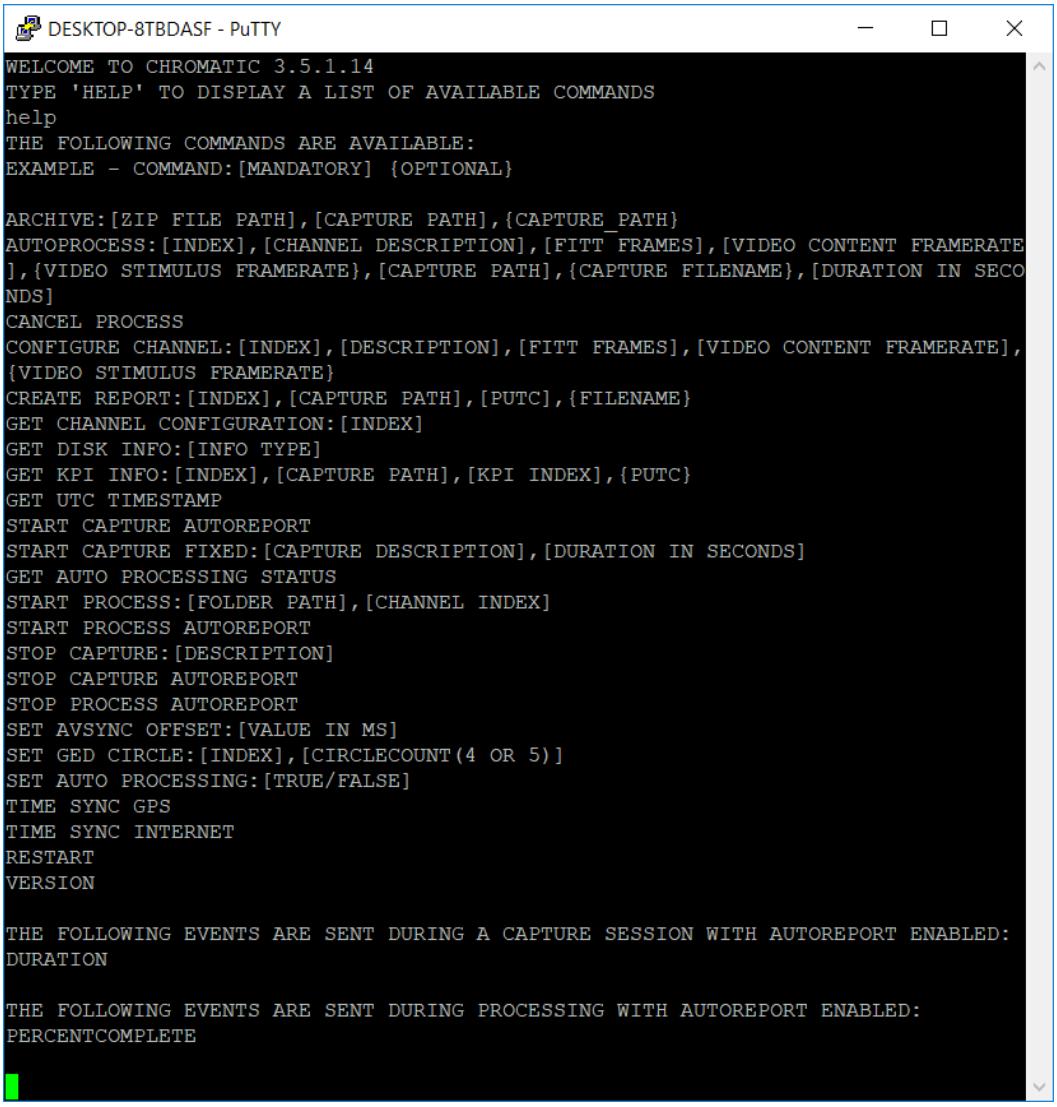

Figure 4. List of Umetrix commands in PuTTY window.

Using this window, you can enter the remote control commands for Umetrix Video. Go to the section “GED Remote Control Command Set” on page 11 for a description of each command.

3. GED Remote Control Command Set

Overview

This section describes the GED remote control commands available for Umetrix Video.

Umetrix Video installs a server host that automatically runs when the program is executed, providing limited remote control ability. Socket level communication is available on port 7073 from a “socket client” such as PuTTY or Windows HyperTerminal.

GED Remote Control Commands

The following remote control commands are available:

- CONFIGURE CHANNEL: ChannelIndex, ChannelDescription, VideoContentName
- GET CHANNEL CONFIGURATION: ChannelIndex
- START CAPTURE AUTOREPORT
- STOP CAPTURE AUTOREPORT
- START CAPTURE FIXED: SessionDescription. DurationInSeconds
- STOP CAPTURE: SessionDescription
- START PROCESS AUTOREPORT
- STOP PROCESS AUTOREPORT
- START PROCESS: CapturePath, ChannelIndex
- CANCEL PROCESS
- AUTOPROCESS
- CREATE REPORT
- GET UTC TIMESTAMP
- TIME SYNC INTERNET
- TIME SYNC GPS
- VERSION
- GET DISK INFO
- ARCHIVE
- RESTART
- SET GED CIRCLE

- SET AVSYNC OFFSET
- SET AUTO PROCESSING
- GET AUTO PROCESSING STATUS

When using the remote control commands, keep in mind the following information:

- Commands may be entered in either upper or lower case.
- Commands should be terminated by with a CRLF (Carriage-Return/Line-Feed). In Hyper Terminal, configure the ENTER button to perform this task by selecting the Send line ends with ends with line feeds option in the ASCII Setup dialog. When interfacing programmatically, this can be accomplished by appending **Environment.NewLine** to the command string.
- Commands that execute successfully will be prefixed with the response **OK**.
- Commands that fail to execute will be prefixed with the response **ERROR (code):text:[optionalParameters]**, where code is an integer error code, text is the corresponding description of the error, and optional parameters may be returned based on the error. See the section “Error Codes” on page 26 for a list of all possible error codes, text, and optional parameters.
- If Umetrix Video stops responding, restart Umetrix Video and the socket connection, and then try again.

CONFIGURE CHANNEL

Command Format	**CONFIGURE CHANNEL: ChannelIndex, ChannelDescription, FITTFrames, VideoContentFramerate, [VideoStimulusFramerate]**
Purpose	Sets up a channel configuration for recording. This step is required for all enabled channels before starting a capture.
Parameter(s)	• ChannelIndex • ChannelDescription • FITTFrames (must be between 1 and 12 inclusive) • VideoContentFramerate (must be between 1 and 60 inclusive) • VideoStimulusFramerate [optional - must be between 1 and 60 inclusive. If specified, indicates this is Video Chat].
Example	This example configures the channel at index 0 with the channel description of “phone A”, for video chat with FITT=6, content framerate=15 and stimulus framerate =20. **CONFIGURE CHANNEL: 0, phone A, 6 ,20, 15**
Umetrix Video Response	• Successful Response: “OK: Channel ChannelIndex configured” • Possible Failure Responses: ▪ “Error (8): Recording is in progress” ▪ “Error (2): Parameter string cannot be empty” ▪ “Error(3): Parameter string not formatted properly” ▪ “Error(4): Parameter string does not contain enough arguments” ▪ “Error(6): Channel at this index is not enabled” ▪ “Error(9): Invalid content description name” ▪ ERROR (28):FITT FRAMES MUST BE BETWEEN 1 AND 12 INCLUSIVE. ▪ ERROR (29):CONTENT FRAME RATE MUST BE BETWEEN 1 AND 60 INCLUSIVE. ▪ ERROR (30):STIMULUS FRAME RATE MUST BE BETWEEN 1 AND 60 INCLUSIVE.

GET CHANNEL CONFIGURATION

Command Format	**GET CHANNEL CONFIGURATION: ChannelIndex**
Purpose	Sets up a channel configuration for recording. This step is required for all enabled channels before starting a capture.
Parameter(s)	ChannelIndex
Example	This example gets the configuration for the channel at index 0: **GET CHANNEL CONFIGURATION: 0**
Umetrix Video Response	• Successful Response: "OK: Channel Configuration: channelIndex,ChannelDescription,VideoContentName" • Possible Failure Responses: ▪ "Error (8): Recording is in progress" ▪ "Error (2): Parameter string cannot be empty" ▪ "Error (3): Parameter string not formatted properly" ▪ "Error (4): Parameter string does not contain enough arguments" ▪ "Error (6): Channel at this index is not enabled" ▪ "Error (9): Invalid content description name"

START CAPTURE AUTOREPORT

Command Format	**START CAPTURE AUTOREPORT**
Purpose	Enable real-time reporting of capture status.
Parameter(s)	None
Example	**START CAPTURE AUTOREPORT**
Umetrix Video Response	• Successful Response: "OK: Capture status reporting enabled" • Possible Failure Responses: This method just sets a flag internally, and will not return an error. With capture autoreport enabled, Umetrix Video will report the current duration of the capture every 10 seconds in the following format. This example represents a status of 10 seconds into a test. DURATION 00:00:10/00:01:00

STOP CAPTURE AUTOREPORT

Command Format	**STOP CAPTURE AUTOREPORT**
Purpose	Disables real-time reporting of capture status.
Parameter(s)	None
Example	**STOP CAPTURE AUTOREPORT**
Umetrix Video Response	• Successful Response: "OK: Capture status reporting disabled" • Possible Failure Responses: This method just sets a flag internally, and will not return an error.

START CAPTURE FIXED

Command Format	**START CAPTURE FIXED: SessionDescription, DurationInSeconds**
Purpose	Starts a capture of defined fixed duration.
Parameter(s)	• SessionDescription – a description for the capture session as a whole. This is the same text that a user running Umetrix Video from the UI normally enters when they stop the capture. • DurationInSeconds
Example	This command starts a 60 second capture with the description of "location b session": **START CAPTURE FIXED: location b session, 60**
Umetrix Video Response	• Successful Response: "OK: Capture for numSeconds started to:captureFilePath" • Possible Failure Responses ▪ "Error (8): Recording is in progress" ▪ "Error (2): Parameter string cannot be empty" ▪ "Error (3): Parameter string not formatted properly" ▪ "Error (4): Parameter string does not contain enough arguments" ▪ "Error (12): Not all enabled channels are configured" ▪ "Error (11): Value for duration is invalid"

STOP CAPTURE

Command Format	**STOP CAPTURE: SessionDescription**
Purpose	Stops any capture currently running. This can be called to preemptively stop a capture of fixed duration, but does not need to be. If it is called to stop a capture session of fixed duration, then the session description provided in this call will override that provided in the START CAPTURE FIXED command.
Parameter(s)	SessionDescription - a description for the capture session as a whole.
Example	This example stops the currently running capture session and gives it a session description of "my first session": **STOP CAPTURE: my first session**
Umetrix Video Response	• Successful Response: "OK: Capture completed:CaptureFilePath" • Possible Failure Responses: ▪ "Error(13): Recording is not in progress" ▪ "Error(14): Description cannot be empty"

START PROCESS AUTOREPORT

Command Format	**START PROCESS AUTOREPORT**
Purpose	Enables real-time reporting of processing status.
Parameter(s)	None
Example	**START PROCESS AUTOREPORT**
Umetrix Video Response	• Successful Response: "OK: Processing status reporting enabled" • Possible Failure Responses: This method just sets a flag internally, and will therefore not return an error. With process autoreport enabled, Umetrix Video will report the current percent complete of the processing as it advances in the following format: Status 20% complete

STOP PROCESS AUTOREPORT

Command Format	**STOP PROCESS AUTOREPORT**
Purpose	Disables real-time reporting of processing status.
Parameter(s)	None
Example	**STOP PROCESS AUTOREPORT**
Umetrix Video Response	• Successful Response: "OK: Processing status reporting disabled" • Possible Failure Responses: This method just sets a flag internally and will therefore not return an error.

START PROCESS

Command Format	**START PROCESS: CapturePath, ChannelIndex**
Purpose	Executes post processing on the referenced channel in a session.
Parameter(s)	CapturePath – The command will work with either the captureinfo.xml full file path as returned by the start capture command, or just the path to the parent folder of the captureinfo.xml file. ChannelIndex
Example	This example starts processing on channel 0 of the session stored in C:\Users\chromatic_user\AppData\Local\Metrico Wireless\Chromatic\captures\2011-03-18-131509 **START PROCESS: C:\Users\chromatic_user\AppData\Local\ Metrico Wireless\Chromatic\captures\2011-03-18-131509, 0**
Umetrix Video Response	• Successful Response: "OK: Processing started" • Possible Failure Responses: ▪ "Error(8): Recording is in progress" ▪ "Error(2): Parameter string cannot be empty" ▪ "Error(3): Parameter string not formatted properly" ▪ "Error(4): Parameter string does not contain enough arguments" ▪ "Error(5): Invalid channel index" ▪ "Error(17): Unable to locate session" ▪ "Error(15): Session does not contain channel at given index" When processing completes, one of the following status messages will be reported: • "Processing Completed" –returned when processing completes with no errors. • "Processing Aborted" – returned if processing is cancelled, either from the remote control interface, or from the user interface. • "Processing Completed with Error in Frame Processing" – returned if processing completed, but there was an error in frame processing. • "Processing Completed with Error in Post-Processing" – returned if processing completed, but there was an error in post-processing.

CANCEL PROCESS

Command Format	**CANCEL PROCESS**
Purpose	Cancels any currently running post-processing tasks.
Parameter(s)	None
Example	**CANCEL PROCESS**
Umetrix Video Response	• Successful Response: OK: Processing Aborted • Possible Failure Responses: “Error(16): Processing not in progress”

AUTOPROCESS

Command Format	**AUTOPROCESS: ChannelIndex, ChannelDescription, FITTFrames, VideoContentFramerate, [VideoStimulusFramerate], CaptureFolderPath, [CaptureFilename], CaptureLength**
Purpose	Configures a channel and performs a fixed duration capture on the selected channel index for the supplied capture length in seconds. After capture completes, data captured video is automatically post-processed.
Parameter(s)	• ChannelIndex • ChannelDescription • FITTFrames (must be between 1 and 10) • VideoContentFramerate (must be between 1 and 30) • VideoStimulusFramerate [optional - must be between 1 and 30] • CaptureFolderPath • CaptureFilename - Optional • CaptureLength
Example	**AUTOPROCESS: 0,Capture 1,6,30,,C:\ChromaticCaptures\1\,,60**
Umetrix Video Response	• Successful Response: OK: CaptureFolderPath • Possible Failure Responses: ▪ “Error(27): There are more channels enabled than this command supports” ▪ “Error(5): Parameter string does not contain enough arguments” ▪ “Error(26): A previous capture’s data exists at location” ▪ ERROR (28):FITT FRAMES MUST BE BETWEEN 1 AND 12 INCLUSIVE. ▪ ERROR (29):CONTENT FRAME RATE MUST BE BETWEEN 1 AND 60 INCLUSIVE. ▪ ERROR (30):STIMULUS FRAME RATE MUST BE BETWEEN 1 AND 60 INCLUSIVE.

CREATE REPORT

Command Format	**CREATE REPORT: ChannelIndex, CaptureFolderPath, PlaytimeUTC,[ReportFileName]**
Purpose	Create a Umetrix Video Excel report from the Capture in CaptureFolderPath.
Parameter(s)	• ChannelIndex • CaptureFolderPath • PlaytimeUTC • ReportFileName
Example	**CREATE REPORT:0,C:\ChromaticCaptures\1,1/1/2012 1:00:00pm**
Umetrix Video Response	• Successful Response: OK: Report filepath • Possible Failure Responses: ▪ "Error(5): Parameter string does not contain enough arguments" ▪ "Error(22): Capture not found" ▪ "Error(3): Parameter string not formatted properly" ▪ "Error(23): Capture has not been processed"

GET UTC TIMESTAMP

Command Format	**GET UTC TIMESTAMP**
Purpose	Returns a string representing the Umetrix Video system's current UTC time. String format is: "yyyy/MM/dd HH:mm:ss.fff"
Parameter(s)	None
Example	**GET UTC TIMESTAMP**
Umetrix Video Response (example)	"2013/11/15 12:24:30.254"

TIME SYNC INTERNET

Command Format	**TIME SYNC INTERNET**
Purpose	Attempts to synchronize the test PC clock to the Audio Server using internet time synchronization.
Parameter(s)	None
Example	**TIME SYNC INTERNET**
Umetrix Video Response	• Successful Response: "OK: Internet time sync started" • Possible Failure Responses: "Error(20): Could not start Internet time sync"

TIME SYNC GPS

Command Format	**TIME SYNC GPS**
Purpose	Attempts to synchronize the test PC clock to the Audio Server using GPS time synchronization.
Parameter(s)	None
Example	**TIME SYNC GPS**
Umetrix Video Response	• Successful Response: "OK: GPS Time sync started" • Possible Failure Responses: "Error(19): Could not start GPS time sync"

VERSION

Command Format	**VERSION**
Purpose	Returns the version of the currently running Umetrix Video application.
Parameter(s)	None
Example	**VERSION**
Umetrix Video Response	"Chromatic Version: version number"

GET DISK INFO

Command Format	**GET DISK INFO: infoType**
Purpose	Returns the value in bytes on the main system hard drive for the corresponding infoType requested.
Parameter(s)	infoType: Either ‘total’ or ‘free’ to represent the information needed.
Example	**GET DISK INFO: total**
Umetrix Video Response	• Successful Responses: “OK: 512002879488“ • Possible Failure Responses: “Error (3): Parameter string not formatted properly“

ARCHIVE

Command Format	**ARCHIVE: ExternalLocation,CapturePath[,CapturePath]**
Purpose	Create a ZIP archive of captures and move them to a designated location.
Parameter(s)	• ExternalLocation: The complete file path to the location to move the created archive. • CapturePath: One or more file paths pointing to Umetrix Video captures.
Example	**ARCHIVE: E:\ChromaticArchives\1.zip, C:\ChromaticCaptures\1**
Umetrix Video Response	• Successful Responses: OK: ExternalLocation • Possible Failure Responses: ▪ Error (2): Parameter string cannot be empty ▪ Error (3): Parameter string not formatted properly ▪ Error (22): Capture not found, CapturePath ▪ Error (25): ExternalLocation not found or inaccessible

RESTART

Command Format	**RESTART**
Purpose	Restarts the Umetrix Video application.
Parameter(s)	None
Example	**RESTART**
Umetrix Video Response	There is no response from Umetrix Video for this command. The Umetrix Video application will automatically close and reopen.

SET GED CIRCLE

Command Format	**SET GED CIRCLE: Channel Index (0 or 1), Circle Count (4 or 5)**
Purpose	This sets the number of circles to be used for testing. By default, 5 circles will be used for each channel. 4 circles cannot be used for a 60fps configuration.
Parameter(s)	• Channel Index – Channel to be configured. • Circle Count – Number of circles to be used for the test.
Example	**SET GED CIRCLE: 0, 4**
Umetrix Video Response	• Successful Response: "OK: Channel 0/1 configured with 4/5 circles" • Possible Failure Responses: ▪ "Invalid Ged Circle Count (Valid 4 or 5)" ▪ "4 Circles cannot be used for 60fps Error(14): Description cannot be empty."

SET AVSYNC OFFSET

Command Format	**SET AVSYNC OFFSET: Offset value**
Purpose	Updates the AVSYNC value in settings. This offset value is used in the calculation of AV sync.
Parameter(s)	Offset value - a description for the AVSYNC as a whole.
Example	**SET AVSYNC OFFSET: 200**
Umetrix Video Response	• Successful Response: "OK: AV Sync value successfully saved" • Possible Failure Responses: "Invalid AV Sync value entered"

SET AUTO PROCESSING

Command Format	**SET AUTO PROCESSING: Auto Process Value to be set (true or false)**
Purpose	By setting the value to true, the session will be automatically processed after the capture of session is complete.
Parameter(s)	AutoProcess value – value to be set for the sessions to be processed.
Example	**SET AUTO PROCESSING: true/false**
Umetrix Video Response	• Successful Response: ▪ "OK: Auto Processing Session after Capture is ENABLED" ▪ "Auto Processing Session after Capture is DISABLED" • Possible Failure Responses: "Invalid arguments (true/false)"

GET AUTO PROCESSING STATUS

Command Format	**GET AUTO PROCESSING STATUS**
Purpose	Get the current setting for AUTO PROCESSING.
Parameter(s)	None
Example	**GET AUTO PROCESSING STATUS**
Umetrix Video Response	Successful Response: "OK: AUTO PROCESSING SESSION: ENABLED/DISABLED"

Error Codes

Errors will be returned in the format ERROR([code]):[text]:[parameters], where code is the integer code representing the error, text is the textual description of the error, and parameters is a placeholder for optional parameters depending on the error returned.

The following is an example of an error returned when attempting to configure a non-existent channel:

ERROR (6):CHANNEL AT THIS INDEX IS NOT ENABLED:4

In this example, the error code is 6, and the optional parameter returned, '4', is the user supplied invalid channel index.

A complete listing of error codes is generated at run time and saved to the file **AutomationErrors.xml** in the Application data directory (for example, **C:\Users\chromatic_user\AppData\Local\Metrico Wireless\Chromatic\AutomationErrors.xml**). In the event that this document becomes out of sync with the Umetrix Video code base (if new error codes are added, and the document has yet to be formally modified and released), the **AutomationErrors.xml** file will always contain the most up-to- date listing of all error codes.

Code	Text	Parameters
0	No Error	n/a
1	Unknown Command	Received command text
2	Parameter string cannot be empty	n/a
3	Parameter string not formatted properly	n/a
4	Parameter string does not contain enough arguments	n/a
5	Invalid Channel Index	User provided value for channel index
6	Channel at this index is not	User provided channel index
7	Channel not configured	User provided channel index
8	Recording is in progress	n/a
9	Invalid content description name	
10	Processing in Progress	n/a
11	Value for duration is invalid	User provided invalid duration
12	Not all enabled channels are configured	n/a
13	Recording is not in progress	n/a

Code	Text	Parameters
14	Description cannot be empty	n/a
15	Session does not contain channel at given index	User provided channel index
16	Processing not in progress	n/a
17	Unable to locate session	n/a
18	Channel already processed	n/a
19	Could not start GPS Time Sync	n/a
20	Could not start Internet Time Sync	n/a
21	Invalid KPI index	User provided KPI Index
22	Capture not found	User provided CapturePath
23	Capture has not been processed	n/a
24	First moving frame not found	n/a
25	Location not found or inaccessible	User provided location
26	A previous capture's data exists at location	n/a
27	There are more channels enabled than this command supports	n/a
28	An unknown error has occurred	n/a

4. Sample Remote Control Command Session

Follow the steps in this section to learn how to control Umetrix Video remotely for GED tests.

The sample command sequence presented here captures a 30 second clip from channel A (index 0). Before performing these steps, use the user interface of Umetrix Video to make sure that:

- Channel B is disabled/
- Channel A is properly configured (from the **Config** button on the **Capture** tab).

Perform the following steps:

1. At the PuTTY or HyperTerminal prompt, type the following command to configure channel A:
   CONFIGURE CHANNEL: 0, cameraA, 6, 30
2. Press the ENTER key.

   The command and the response to the command will be displayed in the window.

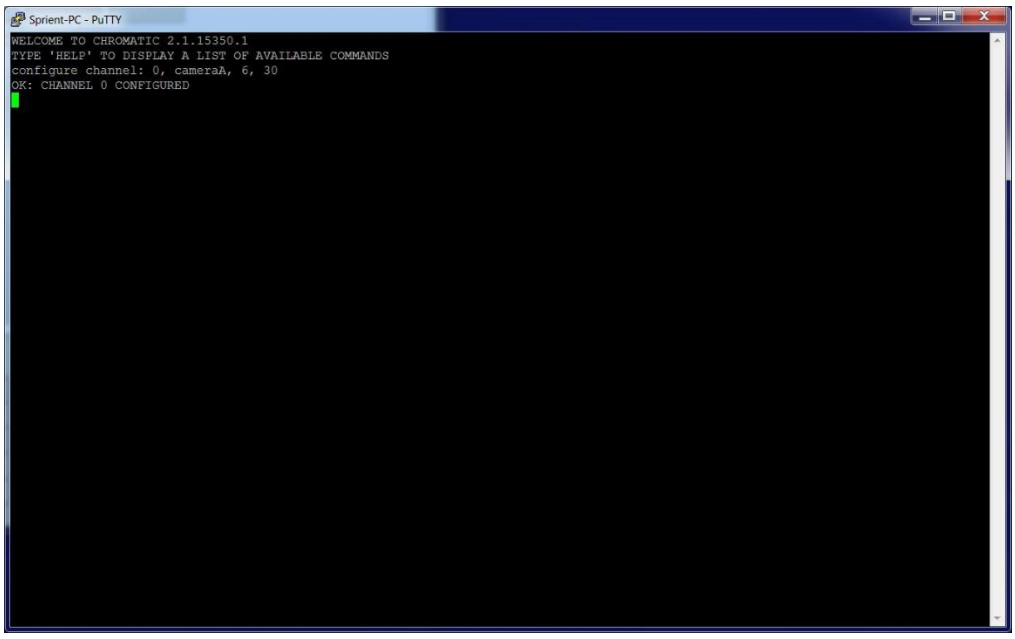

Figure 5. CONFIGURE CHANNEL command and response in PuTTY window.

This command configures channel A (index 0) with the following properties:

- ChannelIndex = 0
- ChannelDescription = camera
- FITTFrames = 6
- VideoContentFramerate = 30
- VideoStimulusFramerate [optional] = not used

3. To configure the second channel, type
CONFIGURE CHANNEL: 1, cameraB, 5, 25
and press the ENTER key.
4. To start a fixed length capture test session, type
START CAPTURE FIXED: My Test Capture, 20
and press the ENTER key.

   This command executes a fixed length capture on the configured channel, with the following properties:

   - Session Description: My Test Capture
   - Session Duration: 20 seconds

   The command and response will be displayed in the window. The initial response indicates that the capture was started and provides the path to the captureinfo.xml file associated with the capture session. When the capture completes, another message is sent that indicates that the capture completed and provides the path to the captureinfo.xml file associated with the capture session. (See the following figure.)

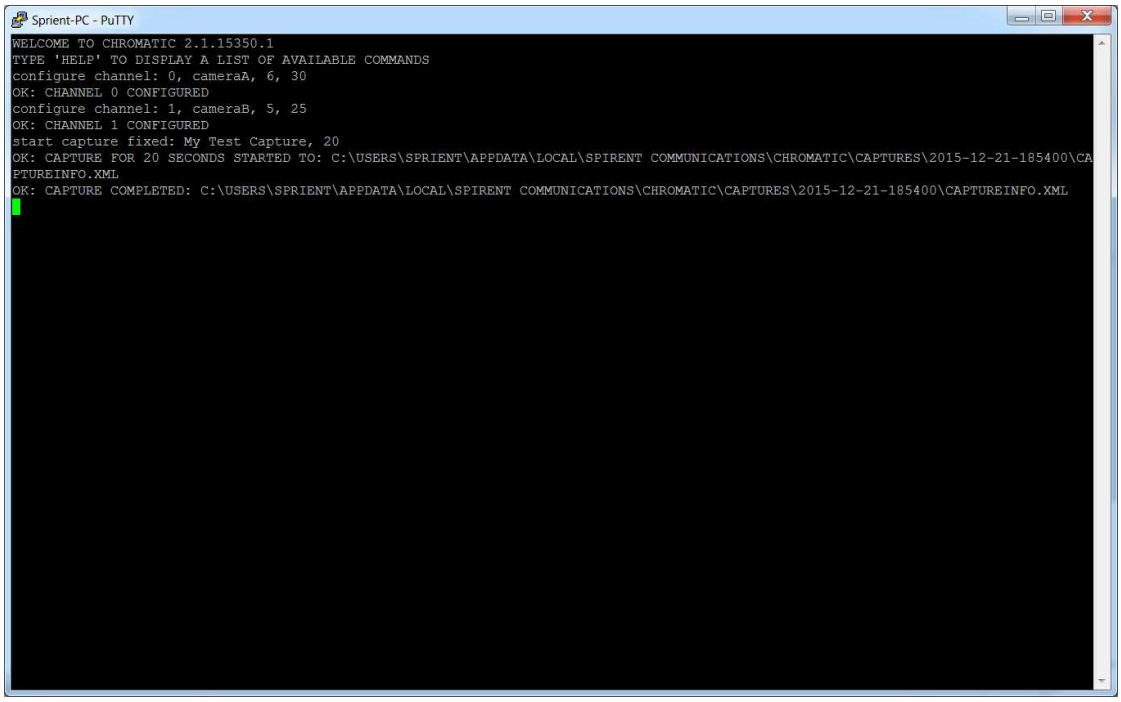

Figure 6. START CAPTURE FIXED command and response.

5. Interfacing Programmatically

This section is not intended to represent a fully functional interface to the Umetrix Video GED Remote Control API. The following code represents a basic WPF application to send commands to Umetrix Video and display return values and should be considered a base starting point for developing an application interface.

The following is the code behind for Window1.xaml:

```
/// <summary>
    /// Interaction logic for Window1.xaml
    /// </summary>
    public partial class Window1 : Window
    {
        private TcpClient _client; private
        NetworkStream _stream; private
        byte[] _receiveBuffer; private
        StringBuilder _resultsSB; private
        bool _stillRunning = true; private
        Thread _listen;

        public Window1()
        {
            InitializeComponent();
            this._client = new TcpClient("127.0.0.1", 7073);
            this._stream = this._client.GetStream();
            //this._stream.ReadTimeout = 1000;
            this._receiveBuffer = new byte[this._client.ReceiveBufferSize];
            this._resultsSB = new StringBuilder();
            this._listen = new Thread(new ThreadStart(ListenThread));
            this._listen.Start();
        }

        private void SetResults(string results)
        {
            this.results.Text = results;
        }

        private void ListenThread()
        {
            while(true)
            {
                if (this._stream != null && this._stream.DataAvailable)
                {
                    int lData = this._stream.Read(this._receiveBuffer, 0,
this._client.ReceiveBufferSize);
                    if (lData > 0)
                    {
                        string message =
Encoding.ASCII.GetString(this._receiveBuffer);
```

```
                        message = message.Substring(0, lData);
                        this._resultsSB.Append("RCV: " + message);
                        var str = this._resultsSB.ToString();
                        Dispatcher.Invoke(DispatcherPriority.Normal, new
Action<string>(SetResults), str);
                    }
                }
                else
                {
                    //sleep for a bit as to not tie up processing
                    Thread.Sleep(200);

                }
                if (!this._stillRunning)
                    break;
            }
        }

        private void Send_Click(object sender, RoutedEventArgs e)
        {
            string commandVal = this.command.Text + Environment.NewLine;

this._stream.Write(Encoding.ASCII.GetBytes(commandVal.ToCharArray()),0,commandVal.Leng
th);
            this._resultsSB.Append("SND: " + commandVal);
            this.command.Text = string.Empty;
        }

        private void OK_Click(object sender, RoutedEventArgs e)
        {
            this._stillRunning = false;

            while(this._listen.IsAlive)
            {
                Thread.Sleep(200);
            }
            this._stream.Close();
            this._client.Close();
            this.Close();
        }
    }
```

The following is the code for Window1.xaml:

```
<Window x:Class="AutomationTestApp.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Window1"
        Height="300"
        Width="500">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>

        <!--Input-->
        <StackPanel Orientation="Horizontal"
                    Grid.Row="0"
                    HorizontalAlignment="Stretch">
            <TextBox x:Name="command" Margin="5"
                     Width="400" />
            <Button Margin="5"
                    Padding="10,0"
                    Content="Send"
                    Click="Send_Click" />
        </StackPanel>

        <!--Results-->
        <TextBox x:Name="results"
                    Grid.Row="1"
                 Margin="5"
                 />

        <!--Buttons-->
        <StackPanel Grid.Row="2"
                    Orientation="Horizontal"
                    HorizontalAlignment="Right">
            <Button Margin="5" Padding="10,0"
                    Content="OK"
                    Command="{Binding OKCommand}"
                    Click="OK_Click" />
        </StackPanel>
    </Grid>
</Window>
```