PELCO

by Schneider Electric

ASCII Integration Service

Contents

Overview

The ASCII Integration Service is capable of listening for ASCII commands on a serial port or Ethernet port or optionally both simultaneously. Commands are translated from ASCII into the protocol used by VideoXpert. The ASCII command set used by Pelco has evolved over the years and has several different flavors depending upon the device implementing the command set. The ASCII Integration Service interprets a limited command set defined in the `ASCIICommandConfiguration.xml` file.

Installation

PREREQUISITES

Prior to installation, the following should be installed:

- .NET Framework 4.61 or later
- Microsoft Visual C++ 2015 Redistributables (x86)

PROCEDURE

Note: see **Installation > Prerequisites** for a list of requirements prior to installation. If any of the prerequisites are missing, the service will be unable to run and the installation will fail.

To install the ASCIIVideoXpertTranslatorService, run the provided `setup.msi` file and follow the on-screen prompts.

The service will install as the local system user. Once installed, you need to complete the configuration (described in **ASCII Integration Server Configuration**), and then either restart the system or manually start the service. On subsequent restarts of the system the service will automatically start.

Default XML Files

In the directory `C:\Program Files (x86)\Pelco\ASCIIVideoXpertTranslatorService\` there will be several default XML files with the text "default" preceding the file name. These files are examples of XML configuration files required by the service and should be renamed without "default" in the filename once the file has been modified for use. Each configuration file is described below.

Note that the defaultMonitorToCellMap.xml file should only be saved as MonitorToCellMap.xml if it is desired to translate incoming ASCII Monitor numbers to a VideoXPert Monitor and Cell number, otherwise leave this file alone for a one to one mapping of ASCII Monitor numbers to VideoXPert Monitor numbers.

ASCII Integration Server Configuration

The ASCII Integration Service requires the configuration information contained in the `defaultASCIIEventServerSettings.xml` file to be defined and, as with other files beginning with "default", to be renamed `ASCIIEventServerSettings.xml` once the file has been altered for use.

The `ASCIIEventServerSettings.xml` file contains the following XML elements:

```
<ASCIIEventServerSettings>

        <DebugLevel>1</DebugLevel>

        <VideoXpertCoreAddress></VideoXpertCoreAddress>

        <VideoXpertCorePort></VideoXpertCorePort>
        <VideoXpertUsername></VideoXpertUsername>
        <VideoXpertPassword></VideoXpertPassword>
        <TryPresetIndex>true</TryPresetIndex>
        <EthernetSettings>
              <Address></Address>
              <Port>9999</Port>
              <ConnectionType>UDP</ConnectionType>
        </EthernetSettings>
        <SerialPortSettings>
              <PortName>COM1</PortName>
              <BaudRate>19200</BaudRate>
              <DataBits>8</DataBits>
              <Parity>none</Parity>
              <StopBits>1</StopBits>
        </SerialPortSettings>
</ASCIIEventServerSettings>
```

Where:

- `DebugLevel` is either 0 for off, 1 for normal debug, 2 will output verbose debug messages. Output may be seen by turning off the service and starting the service from the command line or windows explorer.
- `VideoXpertCoreAddress` is a string representing the IPV4 Ethernet address of the core. Example: 192.168.1.100.
- `VideoXpertCorePort` is the port of the Core the ASCII service will be communicating with. Default 443.
- `VideoXpertUserName` is a base64 encoded string. Example: “admin” is encoded as YWRtaW4=
- `VideoXpertPassword` is a base64 encoded string.
- `TryPresetIndex` if true the ASCII Translator will attempt to call a preset by the index in the camera’s preset list if calling by the preset name fails. If false, the ASCII Translator will not attempt to call by index on failure.
- NOTE: `IntegrationId` is no longer configurable. It is assigned by core when registering the external device.
- `EthernetSettings` define the Ethernet address and port used by the ASCII Translator service to listen for ASCII commands on a UDP port. This is useful if the translator resides on a computer with multiple NIC cards. If the address is left undefined, the translator will not listen on an Ethernet port.
- `ConnectionType` defines the Ethernet connection type being used. This may be UDP, TCP or TCP MultiSession. See the Sessions section for more information on TCP MultiSession.
- `SerialPortSettings` define the Serial port settings used by the translator to listen for ASCII commands on a serial port. If the `PortName` is left empty, the translator will not attempt to open a COM port.

Sessions

As of version 2.0.1.0, session information has been added for each connection. The ASCII protocol was designed for serial communication and contains state information after each command. For example, selecting a monitor is a separate command from selecting a camera. Once a monitor is selected, it is “remembered” that any following operations will affect that monitor. Previously, if a command came in from the console to select Monitor 1, then a command came in from the serial port to select Monitor 2, any following commands from either connection would operate on Monitor 2. This is no longer the case. Each connection now remembers the last monitor, camera and monitor cell selected.

The following connections now have their own session information:

- The debug console.
- The serial port connection.
- The UDP connection.
- TCP connections.

The TCP Connection

TCP is a special case, where you are able to select whether all TCP connections share the same session information (by defining the <EthernetSettings> <ConnectionType> as "TCP") or you may select a separate session for each TCP connection (by defining the <EthernetSettings> <ConnectionType> as "TCP MultiSession").

Note that the TCP connection responds with an "AcK" and "NacK" response for every command received. AcK indicates a valid command was received and understood, but does not indicate the command was successful.

Responses

As of version 2.0.6.0, it is possible to configure the service to respond on both the serial and TCP connections. UDP has no response mechanism. A section has been added to the `defaultASCIICommandConfiguration.xml` with the XML tag Response as below:

```
        <Response>
                <!-- to require an ACK set to string value, i.e "AKa" -->
                <Ack></Ack>
                <!-- to require an NAK set to string value, i.e "NAa" -->
                <Nack></Nack>
        </Response>
```

In order to enable responses, simply fill in the string that you desire to send as a response for both the Ack and Nack elements.

TCP always responds as described above in "The TCP Connection", however the values returned by TCP can be overridden with the Response configuration.

The ASCII Command Set

The ASCII command set accepted as input is configurable to a certain degree, allowing the user to define characters or strings of choice along with the command delimiter and placement of parameter, if any. Default command settings are provided in a file named `defaultASCIICommandConfiguration.xml`. If command changes are desired, this file may be copied and renamed `ASCIICommandConfiguration.xml` which will take precedence over the default file settings. Note that all default files will be overwritten on upgrade and should be renamed without "default" in the name to keep them permanently changed. Any changes made to the command set should be to define unique command values, as the actual commands and number of parameters for the command cannot be changed. Values that are highlighted below may be changed.

The `ASCIICommandConfiguration.xml` file contains the following XML elements:

```
<ASCIICommandConfiguration>

      <Commands>

            <Command>

                  <Name>SelectMonitor</Name>

                  <Value>M</Value>

                  <Delimiter>a</Delimiter>

                  <Parameter>

                        <Type>int</Type>

                        <Min>1</Min>

                        <Max>9999</Max>

                        <Position>before</Position>

                  </Parameter>

            </Command>
```

```
    </Commands>
</ASCIICommandConfiguration>
```

Where:

- `Name` is the friendly name of the command.
- `Value` is the ASCII command value that the command interpreter will look for and identify as a "SelectMonitor" command. Value may be a single character or a string. It should be unique. Unfortunately some of the default values are not unique, but the combination of command and delimiter are unique. Command Values should not contain delimiters.
- `Delimiter` is a character or string that acts as the command delimiter. When a delimiter is seen, the interpreter looks for a command. There may be more than one delimiter in the command set. The default delimiters are 'a' and 'm'.
- The `Parameter` element tells the interpreter to look for a parameter and what the `Type`, `Min` and `Max` values for the parameter are. The `Position` element allows the user to select whether the parameter comes "before" or "after" the command.

IMPORTANT
Default commands with parameters must always have a parameter and command without must not. For example, "Select Monitor" must always be supplied a monitor number, so the parameter element must be present and must be of Type integer (int). The only elements that may be changed are the `Min`, `Max` and `Position` elements with the `Parameter`.

Also, commands cannot be added without changes to the code itself. So adding a new command to the file will not result in an action taking place. However, commands may be removed from the file to keep the command from being interpreted.

THE ESCAPE CHARACTER

Commands that take strings are able to use the escape character '\' to escape a character that is a delimiter for use in a string. If delimiters are unique strings, then the escape character is not necessary, however, the default ASCII command set uses 'a' and 'm' as delimiters, therefore to use these in a string, the escape character must precede the delimiter. So "\a" would be used in a string for 'a' to tell the interpreter that the 'a' is not a delimiter, but part of a parameter.

A further complication is that the default Command set utilizes '\' as the Command Value for the "Preset" command. This is a special case where the preset command was defined previously and the '\' character was re-used for another purpose. Repetition of character usage is not recommended and will break command interpretation under normal circumstances.

Camera Configuration

As of VideoXpert version 1.12, camera numbers may be assigned using the VideoXpert Ops Center.

Monitor Configuration

As of VideoXpert version 1.12, monitor numbers may be assigned using the VideoXpert Ops Center.

Alarm Configuration

VideoXpert lacks a method of associating an alarm number to a situation. When an alarm is triggered, the service will look up the alarm in the `AlarmConfiguration.xml` file and inject the associated Situation into VideoXpert. A file named `defaultAlarmConfiguration.xml` is provided as an example of what the `AlarmConfiguration.xml` contents should be.

The `AlarmConfiguration.xml` file contains the following XML elements:

```
<AlarmConfiguration>
```

```
  <Alarm>
      <Name>Alarm 1</Name>
      <Number>1</Number>
      <SourceDeviceId>USE_INTEGRATION_ID</SourceDeviceId>
      <Situation>
            <AlarmState>1</AlarmState>
            <Type>external/ASCII/ASCIIAlarm_active</Type>
            <Property>
                  <Key>1</Key>
                  <Value>uuid1</Value>
            </Property>
            <Property>
                  <Key>alarm_name</Key>
                  <Value>Alarm 1</Value>
            </Property>
            <ExecuteScript>1</ExecuteScript>
            <ExecuteScript>2</ExecuteScript>
      </Situation>
      <Situation>
            <AlarmState>0</AlarmState>
            <Type>external/ASCII/ASCIIAlarm_inactive</Type>
            <Property>
                  <Key>0</Key>
                  <Value>uuid1</Value>
            </Property>
            <Property>
                  <Key>alarm_name</Key>
                  <Value>Alarm 1</Value>
            </Property>
            <ExecuteScript>-1</ExecuteScript>
      </Situation>
  </Alarm>
</AlarmConfiguration>
```

Where:

- `Name` is the friendly name of the alarm.
- `Number` is the number to associate with the alarm.
- `SourceDeviceId` is the Id of the device associated to the alarm. For a pre-defined Situation this would likely be the `deviceId` of a camera that you would like to associate with the alarm. For an external situation type, this can be a unique `uuid` that you create or the keyword "USE_INTEGRATION_ID" may be used to have the service automatically use the `integrationId` that has been assigned by core to fill in the `SourceDeviceId` when an event is injected into VideoXpert.
- `AlarmState` is either 1 or 0 for ON/OFF respectively.
- `Type` is the Situation Type. See the VideoXpert documentation for pre-defined types and parameters. External Situations must be defined in the `CustomSituations.xml` file which is described later in this document.
- A list of properties may be included, which are key/value pairs. Property keys for pre-defined Situation types are pre-defined and must not be changed. External situations allow for adding any user-defined properties that are desired. Note that VideoXpert 1.11 cannot take advantage of these properties, however, future releases of VideoXpert will accept them.

- A list of scripts to execute may be included with the <ExecuteScripts> element. Any script listed will be executed if a corresponding script is found to be defined in the ASCIIScripts.xml file. Scripts are identified by the Script Number (see the Scripting section for more details).

Custom (External) Situations

It is possible to define custom situations, referred to as external situations, and place the created external situation type into the Alarm `<situationType>` of an entry in the `AlarmConfiguration.xml`.

There are several ways to view External Events in VideoXpert: Using VideoXpert Toolbox navigate to the “Events” tab or you can use the Event Viewer plugin within Ops Center.

Note that it is possible to modify External Situation properties within Toolbox, however, these settings will be overwritten by settings within the CustomSituations.xml file (below) each time the service restarts.

THE CUSTOMSITUATIONS.XML FILE

If the event is a custom event to VideoXpert, as configured above, the custom situation must be defined so that it can be sent through to the VideoXpert system. Custom Situations are defined in the file `C:\Program Files (x86)\Pelco\ASCIIVideoXpertTranslatorService\CustomSituations.xml`.

The `CustomSituations.xml` file contains the following XML elements:

```
<CustomSituations>
  <CustomSituation>
      <SituationType>external/ASCII/ASCIIAlarm_active</SituationType>
      <SourceDeviceId></SourceDeviceId>
      <Name>Ascii Alarm Active</Name>
      <Severity>1</Severity>
      <Log>1</Log>
      <Notify>1</Notify>
      <DisplayBanner>1</DisplayBanner>
      <ExpandBanner>0</ExpandBanner>
      <Audible>1</Audible>
      <AckNeeded>0</AckNeeded>
      <AutoAcknowledge>10</AutoAcknowledge>
      <SnoozeIntervals>60,300,600</SnoozeIntervals>
  </CustomSituation>
  <CustomSituation>
      <SituationType>external/ASCII/ASCIIAlarm_inactive</SituationType>
      <SourceDeviceId></SourceDeviceId>
      <Name>Ascii Alarm InActive</Name>
      <Severity>1</Severity>
      <Log>1</Log>
      <Notify>1</Notify>
      <DisplayBanner>1</DisplayBanner>
      <ExpandBanner>1</ExpandBanner>
      <Audible>1</Audible>
      <AckNeeded>0</AckNeeded>
      <AutoAcknowledge>10</AutoAcknowledge>
  </CustomSituation>
</CustomSituations>
```

Where:

- `SituationType` is a string defining the situation type. Must start with "external/" and should follow the rules defined in the VideoXpert protocol documentation.
- SourceDeviceId may be left blank or may be defined in order to act as an optional constraint on the source of events for this Situation. If specified, any events matching the Situation type MUST also match this SourceDeviceId in order for the Situation to apply. "USE_INTEGRATION_ID" may be used to have the service automatically use the `integrationId` that has been assigned by core to fill in the `SourceDeviceId`.
- `Name` is a descriptive string of the situation.
- `Severity` is an integer from 1 – 10.
- `Log` is a Boolean (0 or 1), telling VideoXpert to Log the situation or not.
- `Notify` is a Boolean, telling VideoXpert to Notify users of the situation or not.
- `DisplayBanner` is a Boolean, telling VideoXpert to display a banner in Ops Center or not.
- `ExpandBanner` is a Boolean, telling VideoXpert to expand the banner in Ops Center or not.
- `Audible` is a Boolean, telling VideoXpert to have an audible indication of the event or not.
- `AckNeeded` is a Boolean, telling VideoXpert that an Ack is needed or not.
- `AutoAcknowledge` is an integer number of seconds to automatically acknowledge the alarm.
- SnoozeIntervals is an optional parameter. The default is (60, 300, 600). If you do not wish to change the SnoozeIntervals, leave it blank or remove the element completely.

Scripting

It is possible to define a script or scripts to be ran when an alarm is triggered or cleared. This can be done using the <ExecuteScript> element within the <Alarm> <Situation> element found in the AlarmConfiguration.xml file. A file named ASCIIScripts.xml is used to define scripts that can be executed on an alarm. An example of this file can be found with the other default xml files named defaultASCIIScripts.xml. As with other default files, this file should be saved without the word "default" in the file name once configuration has been completed.

The ASCIIScripts.xml file contains the following XML elements:

```
<ASCIIScripts>
        <Script>
                <Name>Script 1</Name>
                <Number>1</Number>
                <Command>
                        <Name>SetLayout</Name>
                        <Monitor>1</Monitor>
                        <Layout>2x2</Layout>
                </Command>
                <Command>
                        <Name>DisplayCamera</Name>
                        <Monitor>1</Monitor>
                        <Cell>1</Cell>
                        <Camera>1</Camera>
                        <PreviousSeconds>60</PreviousSeconds>
                </Command>
```

```
		<Command>
			<Name>GoToPreset</Name>
			<Monitor>1</Monitor>
			<Camera>1</Camera>
			<Preset>PRESET1</Preset>
		</Command>
		< Action >
			<Name>BookMark</Name>
			<Camera>1</Camera>
			<Description>Script 1 Bookmark</Description>
		</ Action >
	</Script>
	<Script>
		<Name>Script 1</Name>
		<Number>-1</Number>
		<Command>
			<Name>DisplayCamera</Name>
			<Monitor>1</Monitor>
			<Cell>2</Cell>
			<Camera>1</Camera>
			<PreviousSeconds>0</PreviousSeconds>
		</Command>
	</Script>
</ASCIIScripts>
```

Where:

- Name is the friendly name given to the script. This element does not need to be present.
- Number is the number of the script. This field identifies the script and is used in the <ExecuteScript> element when calling the script to run.
- A list of `Commands` may be included, which are actions that will take place when the script is executed. A `Command` must contain the `Name` element and may contain the elements `Monitor, Camera, Cell, Layout, PreviousSeconds, Preset or Pattern` depending upon the `Command.`
- `Name` is the command name and may contain one of the following values: `SetLayout, DisplayCamera, DisconnectCamera, GoToPreset, RunPattern or BookMark.`
- `Monitor` is the number of the Monitor that the command should target.
- `Layout` is the layout value to use in the command. Valid values are: "1x1", "1x2", "2x1", "2x2", "2x3", "3x2", "3x3", "4x3", "4x4", "5x5", "1+12", "2+8", "3+4", "1+5", "1+7", "12+1", "8+2", "1+1+4", "1+4 (tall)", "1+4 (wide)"
- `Cell` is the number of the cell of the Monitor that the command should target. The cell number is 1 to X where X is the number of cells in the layout. For example, in the 4x4 layout, the cell may be 1 to 16.
- `Camera` is the number of the Camera to be used in the command.
- `PreviousSeconds` is a number of seconds before current time to seek the camera to. If 0 or the element is not present, the camera will be displayed live. To display the camera 5 minutes in the past, set `PreviousSeconds` to 300.
- `Preset` is the preset value to use in the command. An example of a commonly used preset value is “PRESET1”.
- `Pattern` is the pattern value to use in the command. An example of a commonly used preset value is “PATTERN1”.
- `Description` is a description for a BookMark. If this field is empty, the description of the BookMark will be “ASCII situation: “, followed by the situation type being injected (if any).

ASCII MonitorToCell Mapping

If the ASCII equipment you are integrating with is not capable of selecting a cell on a monitor, you will need to map the ASCII monitor numbers to VideoXpert Monitor and Cell numbers. A file named defaultMonitorToCellMap.xml is an example of this mapping. This file may be edited and saved as MonitorToCellMap.xml in order to map ASCII monitors to VideoXpert Monitor Cells. Below is an example of the MonitorToCellMap.xml file and the meaning of its contents:

```
<MonitorToCellMap>
        <MonitorToCell>
                <ASCIIMonitor>1</ ASCIIMonitor >
                <VxMonitor>1</ VxMonitor>
                <VxCell>1</VxCell>
        </MonitorToCell>
        <MonitorToCell>
                <ASCIIMonitor>2</ ASCIIMonitor >
                <VxMonitor>1</ VxMonitor>
                <VxCell>2</VxCell>
        </MonitorToCell>
</ MonitorToCellMap >
```

With the above mapping, any time ASCII Monitor 1 is selected, it is mapped to VideoXpert Monitor 1 Cell 1. Any time ASCII Monitor 2 is selected, it is mapped to VideoXpert Monitor 1 Cell 2.

ASCII Preset Mapping

By default (without the PresetMapping.xml file), ASCII preset numbers will call camera presets with the name “PRESET” with the ASCII number appended. So ASCII preset 1 would call “PRESET1”. Note that Scripts allow you to name the preset directly without the use of an ASCII preset number and PresetMapping.xml file has no effect upon Scripts.

Since some cameras do not allow you to rename presets or you may wish to rename them for convenience, the PresetMapping.xml file will let you map an ASCII preset number to a preset name. For instance, this allows you to map ASCII preset 1 to “Lobby Desk Preset” or any other text string your preset is named.

The defaultPresetMapping.xml file contains a sample of what the configuration might look like:

```
<!-- This example shows mapping Presets 1 and 2 to the preset names "PRESET1" and "PRESET2" respectively -->
<PresetMapping>
  <PresetMap>
        <!-- Incoming Preset Number from ASCII command -->
        <ASCIINumber>1</ASCIINumber>
        <!-- Preset String to use when calling preset -->
        <PresetString>PRESET1</PresetString>
        <!-- Camera numbers affected by this mapping, if none listed it will affect all cameras -->
        <Camera>1</Camera>
```

```
            <Camera>2</Camera>

  </PresetMap>

  <PresetMap>

            <!-- Incoming Preset Number from ASCII command -->

            <ASCIINumber>2</ASCIINumber>

            <!-- Preset String to use when calling preset -->

            <PresetString>PRESET2</PresetString>

            <!-- Camera numbers affected by this mapping, if none listed it will affect all cameras -->

            <Camera>1</Camera>

            <Camera>2</Camera>

  </PresetMap>

</PresetMapping>
```

Note that the above mapping is also the default behavior and that any preset not defined in the PresetMapping.xml file will follow the default behavior.

StringToAlarmMap

The ASCII Translator can be placed in a special mode that no longer interprets ASCII commands but looks for specific strings that will cause an alarm to be triggered or cleared. The defaultStringToAlarmMap.xml file, shown below, can be renamed StringToAlarmMap.xml and placed in the `C:\Program Files (x86)\Pelco\ASCIIVideoXpertTranslatorService\` directory to enable this special mode.

```
<!-- This example shows mapping ASCII Strings to triggering Alarms -->
<!-- Note: the presence of StringToAlarmMap.xml will place the ASCII Translator in a mode
          where it no longer interprets ASCII commands.  It will be looking for the strings
          defined here only -->
<StringToAlarmMap>
  <StringToAlarm>
            <!-- Incoming ASCII string -->
            <ASCIIString>TriggerAlarm1</ASCIIString>
            <!-- ASCII Alarm to trigger or clear -->
            <ASCIIAlarm>1</ASCIIAlarm>
            <!-- 0 to clear alarm, 1 to trigger alarm -->
            <AlarmState>1</AlarmState>
  </StringToAlarm>
  <StringToAlarm>
            <!-- Incoming ASCII string -->
            <ASCIIString>ClearAlarm1</ASCIIString>
            <!-- ASCII Alarm to trigger or clear -->
            <ASCIIAlarm>1</ASCIIAlarm>
            <!-- 0 to clear alarm, 1 to trigger alarm -->
            <AlarmState>0</AlarmState>
  </StringToAlarm>
</StringToAlarmMap>
```

In the default example, if the ASCII Translator receives the string “TriggerAlarm1” it will Trigger alarm number 1. If it receives “ClearAlarm1” it will Clear alarm number 1.

Pelco Troubleshooting Contact Information

If the instructions provided fail to solve your problem, contact Pelco Product Support at 1-800-289-9100 (USA and Canada) or +1-559-292-1981 (international) for assistance.

REVISION HISTORY

Manual Title	Date	Comments
ASCII Integration Service	05/17	Formatting changes.
ASCII Integration Service	01/17	Original version.

Pelco by Schneider Electric 3500 Pelco Way Clovis, California 93612-5699 United States
USA & Canada Tel (800) 289-9100 Fax (800) 289-9150
International Tel +1 (559) 292-1981 Fax +1 (559) 348-1120
www.pelco.com www.pelco.com/community