Chromeleon DDK ePanel Selectors

Topics:	ePanel Selectors
File:	Chromeleon 7 DDK ePanel Selectors.docx
Release:	Version 1.2, 09-Jan-2017
Author(s):	Ohlhaut, Marcus
Distribution:	All internal and external developers writing DDK based Chromeleon drivers
Keywords:	Chromeleon 7 DDK ePanel Selectors

Contents

Contents 1
1 Scope of this Document 1
2 References 1
3 Terminology and Abbreviations 2
4 ePanel selection engine overview 2
4.1 ePanel selector XSLT output 5
4.2 Device ePanels 6
4.3 Home ePanel pods 7
4.4 External ePanel macros 9
4.5 Localization support 10
5 Symbol Table Dump Tool 10
6 Debugging the ePanel Selector 12
7 Appendix - Priority ranges 13

1 Scope of this Document

This document is intended for developers using the Chromeleon 7 Device Driver Development Kit (CM7 DDK) to write driver packages for the Chromeleon 7 product family. Readers should already have received a face-to-face training, where the concepts illustrated in this document have been presented.

In addition to the reference documentation (see section 2), which is automatically generated from all the interfaces that the DDK provides, this document contains additional information and background to help developers implement ePanels and ePanel selectors to provide seamless integration of instrument control through Chromeleon 7 ePanels.

We intend to update this document in response to questions arising during the training sessions or sent to us via email (ddksupport@thermofisher.com), so frequent updates are to be expected. Updates will not be distributed automatically, but only on request or with each new DDK release.

2 References

1. DDK V1 Documentation:
   <ChromeleonInstallationFolder>\DDK\Documentation\DDKV1.chm

2. DDK V2 Documentation:
   <ChromeleonInstallationFolder>\DDK\Documentation\DDKV2.chm
3. Chromeleon DDK Driver Certification Requirements.pdf

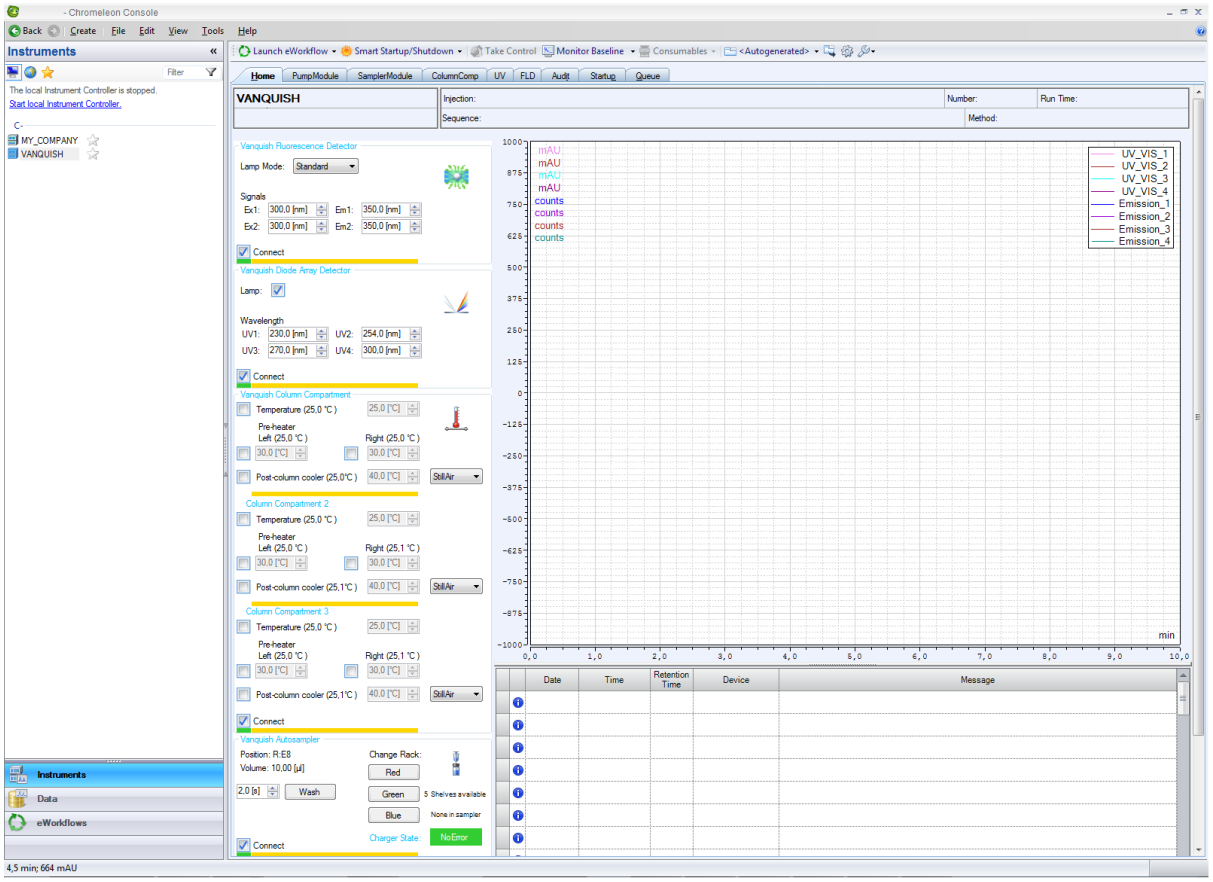
3 Terminology and Abbreviations

Term	Meaning
Instrument	Refers to a set of →Modules that share a common clock. This could be a stack built of individual UltiMate 3000 modules, as well as a stack of Agilent 1200 modules, or any other mix of modules. It could also be a single module, such as an Agilent GC, a Shimadzu LC-2010 compact system or any other module that provides a complete chromatography system in one module.
Driver	Refers to the set of software that is needed to control a →Module. Each driver can belong to one →Instrument or can be shared between instruments. Typically, a driver has a single communication resource that it uses to control its →Module. Note that for each UltiMate 3000 module we would have one driver (as each uses its own USB connection), whereas for a stack of Agilent 1200 modules we would have a single driver (as the whole stack uses a common TCP/IP connection). For an Agilent GC, we would also have one driver, as well as for a Shimadzu LC-2010 compact system.
Module	A piece of chromatographic instrumentation that can be controlled via a single communication resource. Note, that for a stack of Agilent 1200 modules, the whole stack is represented as a single →Module in Chromeleon.
Device	Functional part of a →Module. This can represent any functional group, such as a UV detector as a whole, a relay offered by a →Module, or one of the two flows delivered by a dual pump. Devices can be grouped into a hierarchy.
Script	A series of property assignments, commands (with parameters), time marks and other instructions controlling the execution order. The Chromeleon way of representing full-fledged Instrument Methods as well as single user interactions with the →Instrument.
Channel	A →Device that produces data during an Injection. It has AcqOn and AcqOff commands.
Signal	Data produced by a →Channel device. It can be 2D (such as Pump.Pressure, Oven.Temperature or UV.UV_VIS_1) or 3D such as UV.3DFIELD or FLD.3DFIELD.
CM	Chromeleon
DDK	Chromeleon Device Driver Development Kit

4 ePanel selection engine overview

In Chromeleon Console, Instruments category, Instrument View pane, an automatically determined ePanel set is presented to the user to provide UI to show instrument status details and to allow direct instrument control in an instrument configuration dependent way.

For instance, here is the <Autogenerated> ePanel set for a Vanquish instrument consisting of Vanquish binary pump, autosampler, column compartment and two detectors:



Note that there is a Home ePanel providing an overview of the whole instrument (built up from small pods, an online signal plot and an instrument audit trail grid) as well as one specific Device ePanel for each of the modules (on the PumpModule, SamplerModule, ColumnComp, UV, FLD tabs). Pods are separate ePanel files, for details see 4.3 below.

To determine which ePanels should be shown by default for a given instrument configuration, Chromeleon Console uses a set of XSL files known as ePanel selectors. These files reside in the following directory

<Chromeleon installation directory>\bin\ePanelSelectors[1]

Every supported module has an associated ePanel selector file named to match either the **DriverId** or **ModelNo** property. For example, the selector file for the driver package with **DriverId** `MyCompany.ExampleLCSystem` needs to be named `MyCompany.ExampleLCSystem.xsl`. The selector file should use UTF-8 encoding (with BOM)[2] and needs to declare a parameter called "lang" for localization purposes (see 4.5 below).

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="lang"/>
  <xsl:include href="PanelSelectorCommon.xsl" />
  <xsl:template match="//Symbols/Device[Property[@Name='DriverID' and
                                                 @Value='MyCompany.ExampleLCSystem']]">
    ...
  </xsl:template>
```

[1] For a typical installation (Windows 7 x64 en-US), this would be `C:\Program Files (x86)\Thermo\Chromeleon\bin\ePanelSelectors`.
The Chromeleon installation directory can be determined by querying the SZ_REG value `HKLM\SOFTWARE\Wow6432Node\Thermo\Chromeleon\7\Core Components\InstallFolder`.

[2] We recommend to use an advanced text editor such as Visual Studio, Code or notepad++ which allows to control which encoding is used.

```
</xsl:stylesheet>
```

Once Instrument View has successfully connected to an Instrument, the following steps will occur:

1. Retrieve current symbol table from instrument. (NB: A test harness (see section 5 below) is available that allows to retrieve an XML representation of this.) This symbol table includes all devices, structures, commands and properties, however, values are shown for a few properties[3] only. The driver must ensure that these properties are initialized early (ideally, as part of IDriver.Init); changing a value later on (e.g. as a result of IDriver.Connect) may not be picked up by a client and result in an inappropriate ePanel set being shown. The client does not update the ePanel set upon changes to the values of these properties after initially connecting to the instrument.
2. Iterate over all devices in the symbol table and, for each device, determine whether it has a **DriverId** (preferred) or **ModelNo** (fallback) property. If found, read the property value and look for an ePanel selector file having that name.
3. Run the XSLT defined by the selector file on the symbol table XML to generate a set of XML nodes.
4. From the resulting XML nodes, identify which ePanels are requested for each context (i.e. Home ePanels or Device ePanels). Try to load the corresponding PANX files from the ePanelTemplates directory. The XML nodes typically contain 'external macros' that are injected into the ePanel instance when loaded. Note that the same ePanel may be loaded more than once, but each instance may get different values for the external macros so that each can work with a specific module instance.
5. The corresponding PANX files are loaded from the ePanelTemplates directory.

[3] DeviceType, DriverID, LicenseProvider, ModelNo, ModelVariant, SampleLoadingPump, Location, ValveType, FirmwareVersion, HardwareVersion, PumpDeviceName

4.1 ePanel selector XSLT output

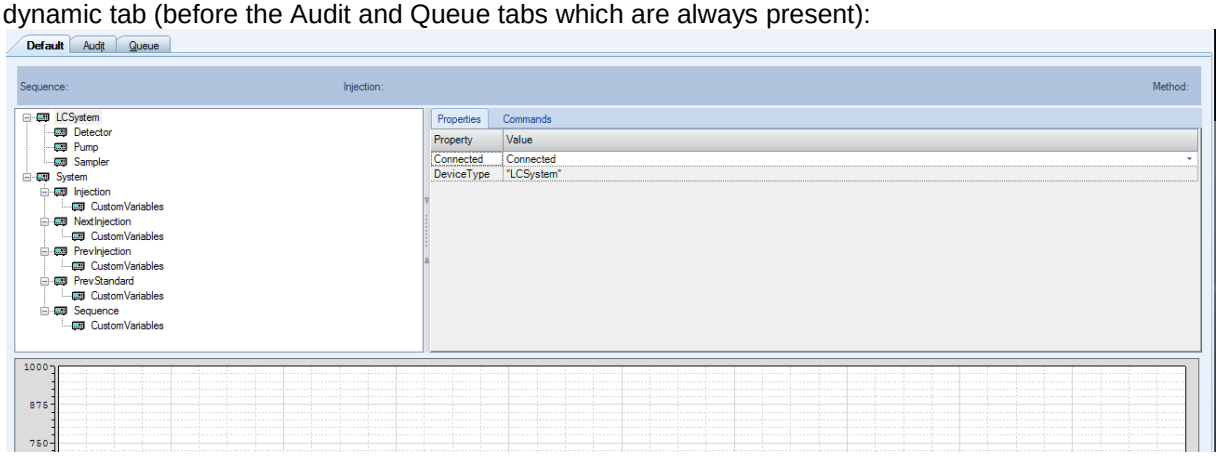
In the XML used in step 4, the requested ePanels are described in the following way:

```
<DevicePanels>
  <Panel Name="ExampleLCSystem.DevicePanel" Priority="1">
    <Macro Name="Main" DeviceName="LCSystem" />
  </Panel>
</DevicePanels>
<HomePanels>
  <Panel Name="ExampleLCSystem.HomePanel" Block="1" Priority="1">
    <Macro Name="Main" DeviceName="LCSystem" />
  </Panel>
</HomePanels>
```

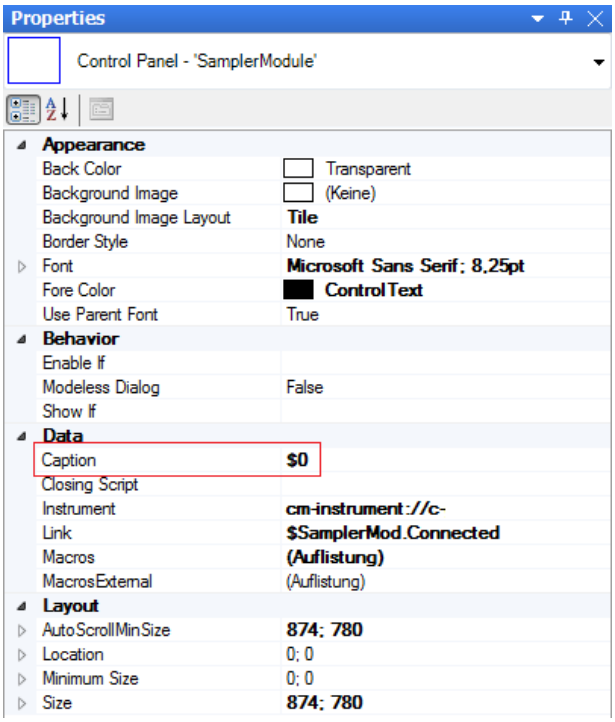
Note that the file name stated in the **Name** attribute does not include the PANX file extension. Chromeleon Console uses a set of PANX files known as ePanel templates. These files reside in the following directory

<Chromeleon installation directory>\bin\ePanelTemplates

If one or more devices have no associated selector file then a 'Default' panel is added as the rightmost dynamic tab (before the Audit and Queue tabs which are always present):



4.2 Device ePanels



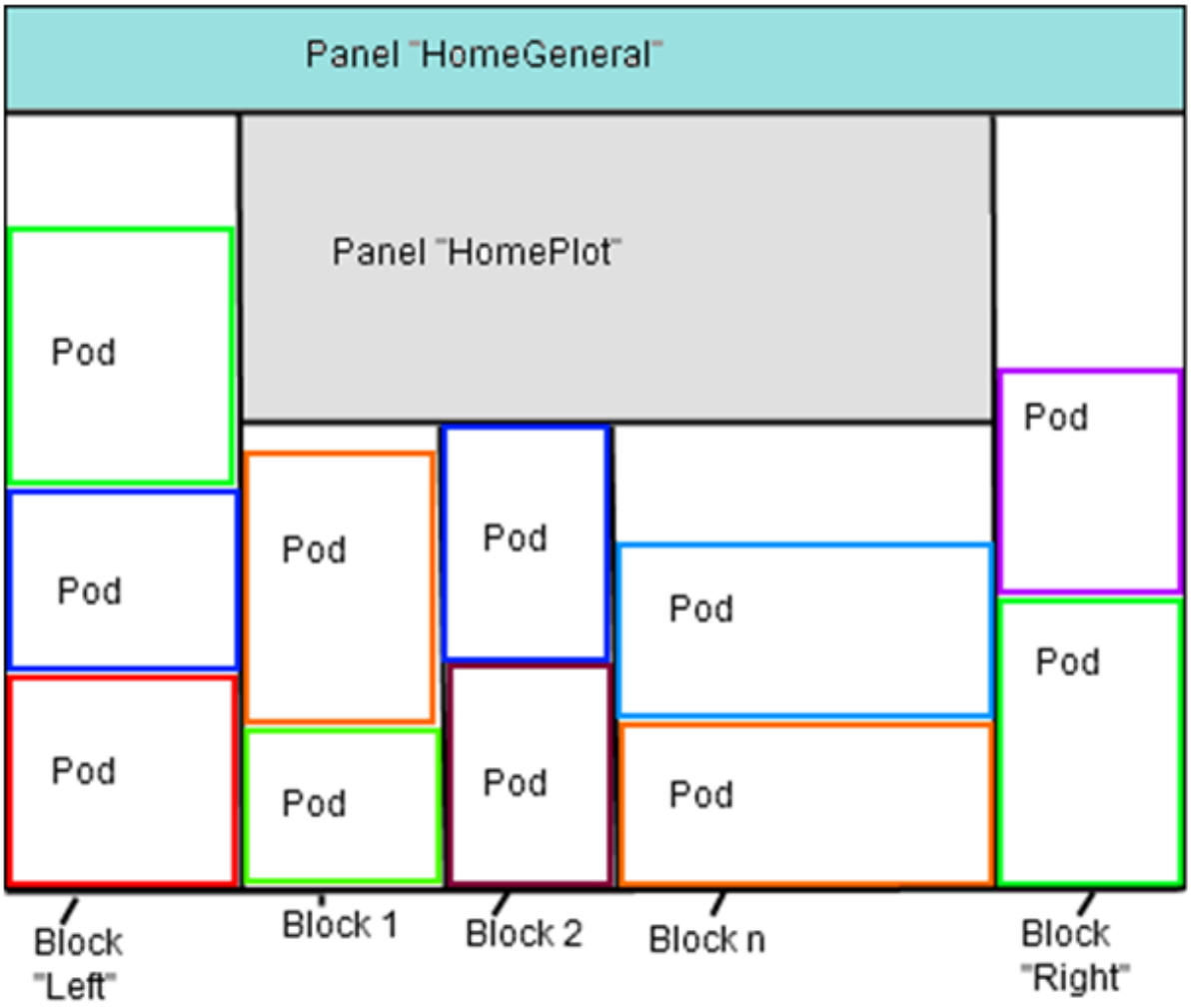
<**Panel**> child nodes of the <**DevicePanels**> node describe individual Device ePanels, each of which are presented on one tab of the <Autogenerated> ePanel set. The text on these tabs is determined by the caption of the main panel contained therein. To ensure unique captions in case a particular ePanel is present multiple times in the ePanel set, you can use the predefined macro $0 for the caption, which evaluates to the corresponding device name[4]. If the caption of the main panel is not defined or evaluates to an empty string then the name of the ePanel (as seen in the ePanel Manager dialog) is used.

The **Priority** value (a number) controls the ordering[5] and is evaluated such that ePanels with lower numbers appear to the left (i.e. Priority =”10” appears left of Priority = “20”). Should two ePanels specify the same priority, tab order depends on the appearance of the corresponding devices in the symbol table (which is not guaranteed). Users can change the tab order via context menu or drag and drop, doing so will, however, change the ePanel set from <Autogenerated> to <Last Used>. Caveat: The <Last Used> ePanel set (as any other named ePanel set) no longer provides external macros to the ePanels, so some functionality will be lost.

[4] For this to work, you must also specify an expression for the Link property that resolves to an existing (arbitrary) symbol.
[5] See Appendix for conventions on Priority.

4.3 Home ePanel pods

<**Panel**> child nodes of the <**HomePanels**> node describe individual ePanel pods, all of which are presented on the **Home** tab of the <Autogenerated> ePanel set. The **Block** value (either "Left", "Right" or a number) controls where the pod is placed; and the **Priority** value (a number) controls the order within that block:



Panels "HomeGeneral" and "HomePlot" are defined by the system and cannot be changed.

- If no pods are defined in Block "Left" then it does not appear (i.e. the "HomePlot" and "Block 1" pods extend to the left edge).
- If no pods are defined in "Block Right" then it does not appear (i.e. the "HomePlot" and "Block n" pods extend to the right edge).
- Numbered blocks are only numbered to indicate relative position (lower numbers to the left); that is, no gaps are reserved for missing numbers.
- Normally, numbered blocks are left-aligned; however, if there is a "Block Right" and no "Block Left" then the numbered blocks are right-aligned.
- The width of a block is determined by the widest pod in the block.
- The height of the numbered blocks is determined by the height of the tallest numbered block (the sum of the heights of its pods).
- Within a block, pods with lower Priority numbers appear above pods with higher Priority numbers. [6]
- Pods are bottom aligned within a block, so one may see space at the top of some blocks.

[6] See Appendix for conventions on Priority.

- The "HomePlot" pod resizes (down to a minimum size set at design time) to fill the remaining space.
- If the "HomePlot" contains a SplitContainer as its first control (e.g. to share space between a plot and an audit trail), the orientation of the splitter changes dynamically such that its panels are stacked vertically if either a "Block Left" or "Block Right" is present and horizontally otherwise.

Note to selector implementers: it is recommended that - if a given driver does not need ePanels of a given type (e.g. no HomePanels items) - an XML comment be inserted indicating this fact so that it is clear to readers of the selector file that the omission is intentional.

```
<xsl:element name="HomePanels">
    <!-- No Home Panels. -->
</xsl:element>
```

4.4 External ePanel macros

Chromeleon ePanels support macro replacement (e.g. in captions and property links) to allow ePanels designed such that changing device names in Chromeleon Instrument Configuration Manager will not break ePanel functionality. Users are free to change the default device names provided by a driver; especially, there may be a need to do so when more than one instance of a given driver is added to the same instrument to resolve duplicated device names.
ePanels can define internal macros (stored within the PANX file), which will resolve to the same value even if that ePanel will be added more than once to an ePanel set.
ePanels can also reference external macros, names and values for which are also defined in the XML output created by the ePanel selector. Each <**Macro**> node has a **Name** attribute which defines the macro's name, referenced in the ePanel designer as $Name; and a **Value** attribute which defines the replacement value. For the example given above, $Main would resolve to "LCSystem" when the ePanel is initialized.

Complex ePanels usually contain controls for both the driver's main device as well as for one or more sub-devices. Typically, one would need to provide one external ePanel macro for each sub-device that is being represented on the ePanel so that the person designing the ePanel can provide appropriate linking for all controls placed on the ePanel. Thus, the ePanel selector typically contains several macro definitions, which use XPath expressions to navigate the symbol table XML in a way appropriate for the driver's symbol table structure:

```
<xsl:call-template name="addDeviceMacro">
    <xsl:with-param name="macroName" select="string('Main')" />
    <xsl:with-param name="deviceName" select="@Name" />
</xsl:call-template>
<xsl:call-template name="addDeviceMacro">
    <xsl:with-param name="macroName" select="string('Detector')" />
    <xsl:with-param name="deviceName" select="./Device[Property[@Name='Wavelength']]/@Name" />
</xsl:call-template>
<xsl:call-template name="addDeviceMacro">
    <xsl:with-param name="macroName" select="string('Pump')" />
    <xsl:with-param name="deviceName" select="./Device[Struct[@Name='Flow']]/@Name" />
</xsl:call-template>
<xsl:call-template name="addDeviceMacro">
    <xsl:with-param name="macroName" select="string('Sampler')" />
    <xsl:with-param name="deviceName" select="./Device[Command[@Name='Inject']]/@Name" />
</xsl:call-template>
```

External macros can be used where ever ePanels accept a path to a device symbol present in the symbol table, for instance in the following places:

- Caption
- Link
- ShowIf expression
- EnableIf expression
- Scripts for Buttons (commands, properties, value expressions and command parameter expressions)
- Script for Close (commands, properties, value expressions and command parameter expressions)

Macro substitution only works for the first component of a symbol path (e.g. "*$UvChannel*.Wavelength" can be resolved to "*MyMainDevice.MyUvChannel*.Wavelength", but "*$UvChannel*_Diagnostics" or "MyMainDevice.*$UvChannel*.Wavelength" will fail to resolve.)

When both an external macro and an internal macro with the same name exist, the value provided by the external macro is being used for the replacement.

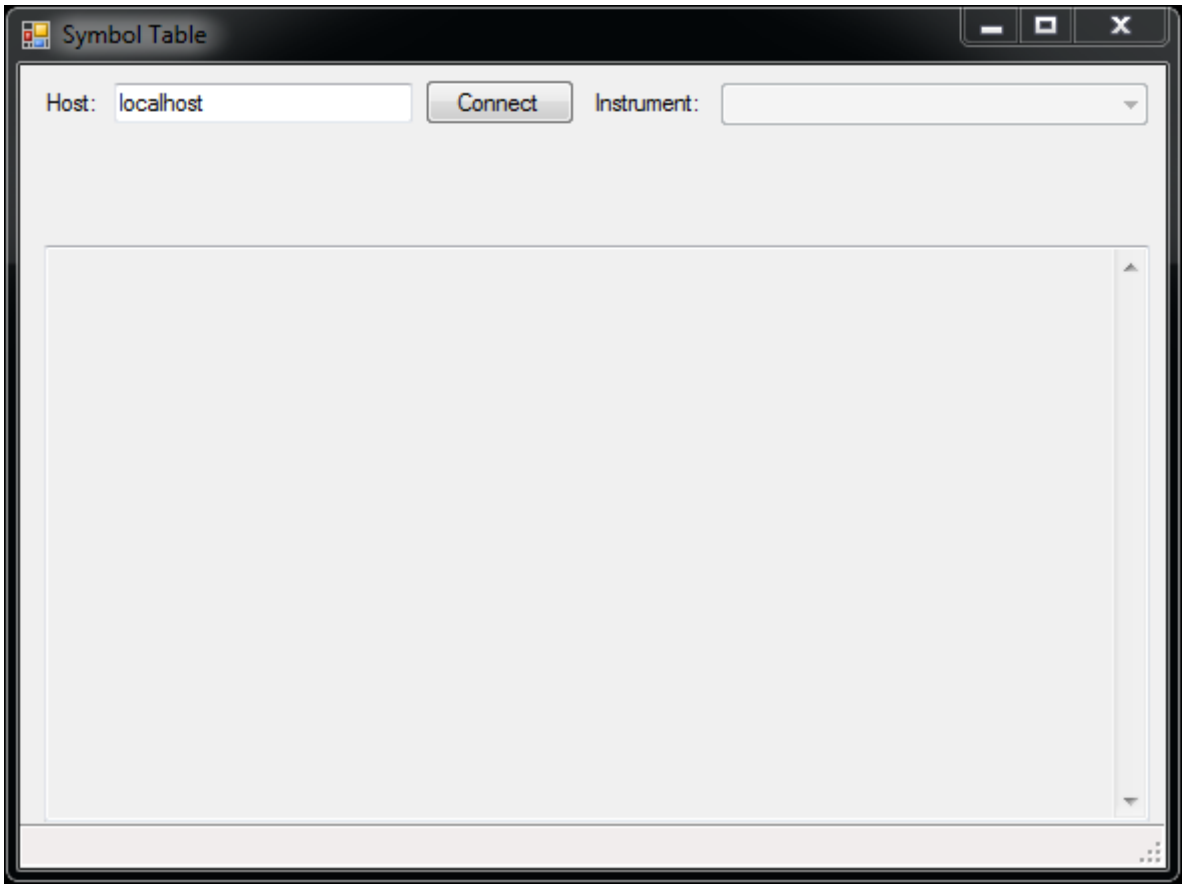
4.5 Localization support

The two-letter ISO language identifier for the current user locale is passed to the XSL transformation as a string parameter named "**lang**". This can be used to control which ePanels should be shown in case this should be language dependent:

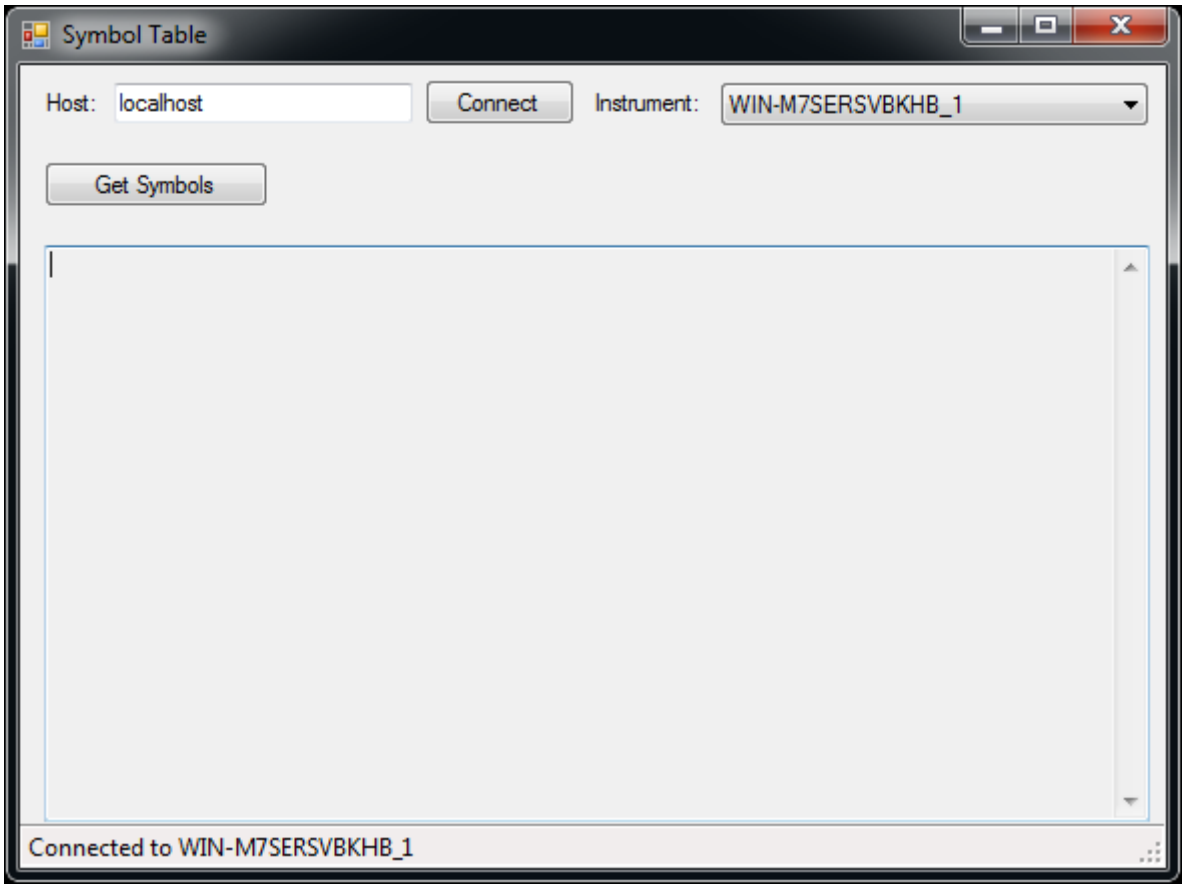
```
<xsl:choose>
    <xsl:when test="$lang='de'">
        <xsl:element name="Panel">
            <xsl:attribute name="Name">Panel_DE</xsl:attribute>
        </xsl:element>
    </xsl:when>
    <xsl:when test="$lang='fr'">
        <xsl:element name="Panel">
            <xsl:attribute name="Name">Panel_FR</xsl:attribute>
        </xsl:element>
    </xsl:when>
    <xsl:otherwise>
        <xsl:element name="Panel">
            <xsl:attribute name="Name">Panel_EN</xsl:attribute>
        </xsl:element>
    </xsl:otherwise>
</xsl:choose>
```

5 Symbol Table Dump Tool

The Chromeleon DDK provides a small tool (SymbolTreeDump.exe) to view the symbol table XML used as the input for the ePanel selection engine. It is accessible through Start Menu > Thermo Chromeleon 7 > DDK Tools > Symbol Table Dump Tool

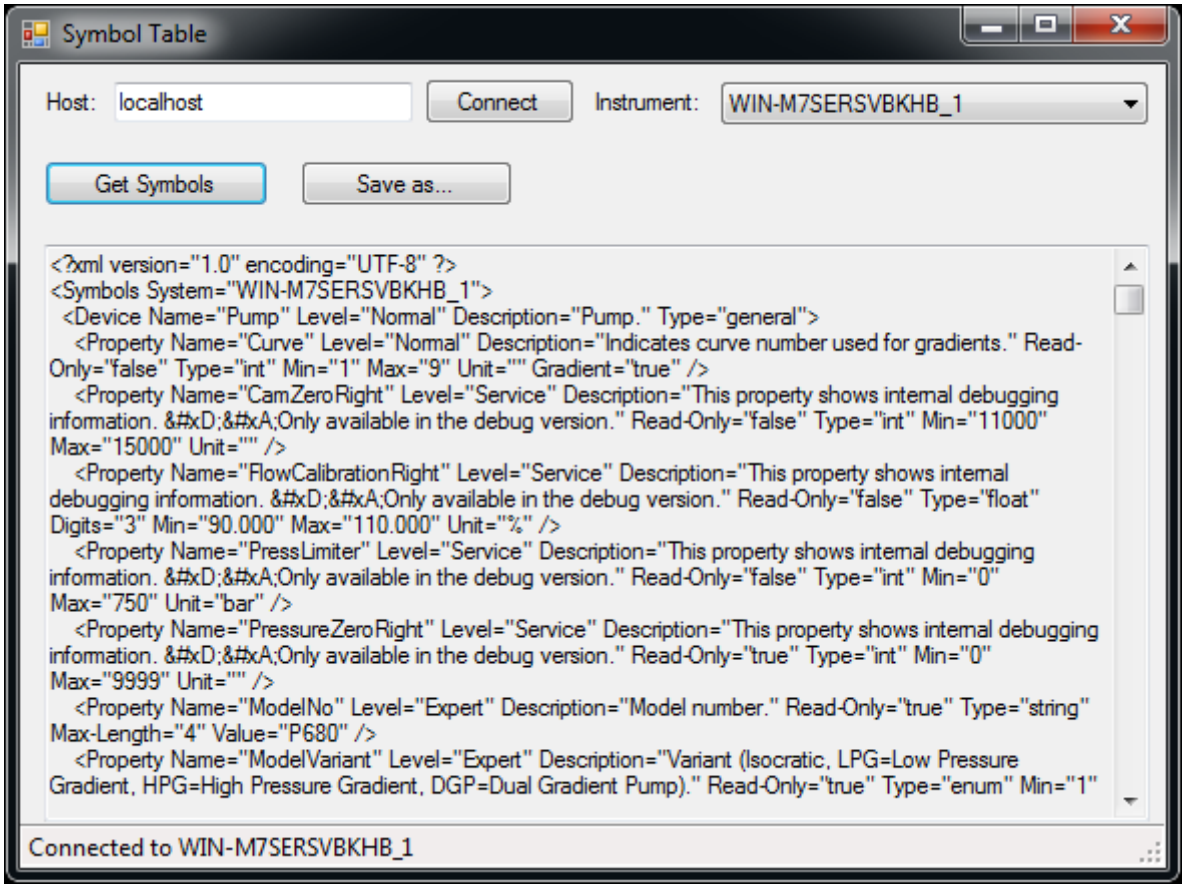


For **Host**, specify the host name of the instrument controller you want to access; click **Connect** to let the tool connect to the instrument controller via remoting. When the connect is successful, the **Instrument** drop list shows the available instruments.



Select the appropriate instrument and wait for the tool to connect to the instrument via remoting. The status line at the bottom of the window will inform you whether the connect is successful, or whether any errors have occurred.

Click **Get Symbols** to retrieve the symbol table XML:

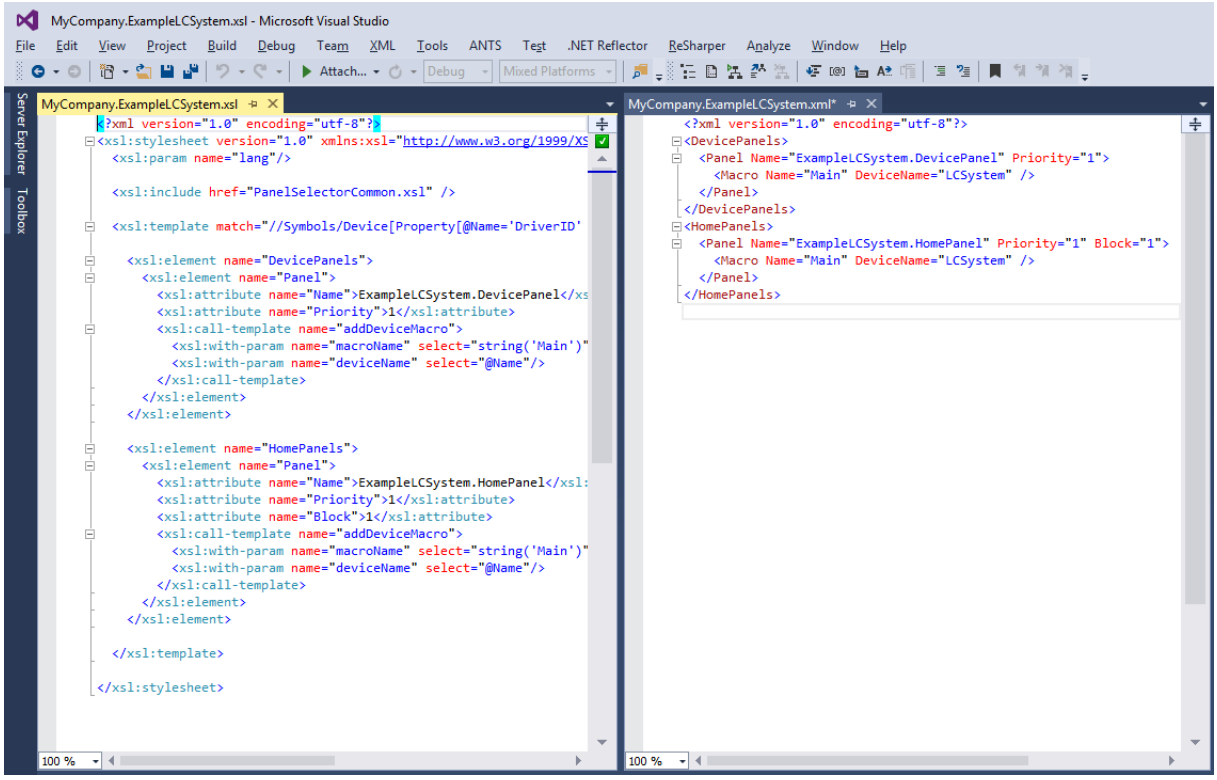


Use the Save as... button to save the data to an XML file; alternatively, you can use **Select All** and **Copy** from the text box's context menu to copy the XML out of the tool if needed.

6 Debugging the ePanel Selector

To debug the ePanel selector XSL, we recommend the following approach:
- Save symbol table XML to a file, e.g. to MyCompany.ExampleLCSystem.xml
- Open ePanel selector XSL in Visual Studio
- From Debug menu, select Start XSLT Debugging
- Visual Studio will ask to Choose Input XML Document, browse to the symbol table XML file.

- Visual Studio will process the XSLT (you can also set breakpoints and step through as usual) and show the result in a new window:



Notes:
- You may need to copy PanelSelectorCommon.xsl (from *<Chromeleon installation directory>\bin\ePanelSelectors*) next to your ePanel selector file, depending on where you test this file.
- The resulting XML is not a well-formed XML document, as it has more than one top-level node. This is, however, the format expected by the ePanel selection engine.

7 Appendix - Priority ranges

The following is how some existing ePanel/Pods have been prioritized in Chromeleon:

Home ePanel	**Horizontal/Vertical Alignment**				
Module	Left Block	Block 1	Block 2	Block 3	Right Block
UltiMate and 3rd Party LC Pump	200				
UltiMate FLM	260				
UltiMate and 3rd Party LC autosampler	400				
UltiMate and 3rd Party LC Column Compartment	500				
UltiMate and 3rd Party LC UV / PDA Detector	600 Exact assignments see below				
UltiMate and 3rd Party FLD	Exact assignments see below				
ELSD	Exact				

	assignments see below				
RI	Exact assignments see below				
Corona	700				
MS	800				
AS		130			
AS-AP		120			
AS-DV		110			
AS-HV		100			
AXP			300		
ICS-1000					311
ICS-1100					310
ICS-1500					321
ICS-1600					320
ICS-2000					331
ICS-2100					330
ICS-4000					340
ICS-900 IC System					301
DC				610	
ERC 10				611	
CD				613	
ED				614	
DP					210
SP					211
EG					250
PDA Photodiode Array Detector (PDA-100)			620		
PDA Photodiode Array Detector (ICS-3000 PDA)			621		
SP Single Pump					211
TC Thermal Compartment			511		
VWD Variable Wavelength Detector			630		

The device ePanels will be ordered in a consistent way regardless whether the instrument is an IC, LC, GC, MS or 3rd Party instrument:

IC Autosampler	100
Pumps	200
Eluent Generator	250
FLM	260
Integrated ICS-Series	300
LC Autosampler	400
Column Compartment	500

Detectors	600
DC	
ECD	
CD	
UV / PDA	
FLD	
ELS	
RI	
Corona Detectors	700
Mass Spec	800

There is also a specific order for GCs.

Home ePanel	100
GC Injector / Autosampler	200
Inlet	300
Valve (belonging to inlet)	400
Oven	500
Column	600
Detector	700
Valve / EPC Modules (if not belonging to the inlet)	800

To ensure Thermo Scientific instruments show up first when used in mixed configurations, use the following priorities:

Thermo Scientific instruments use numbers between x00 to x69

3rd Party instruments use numbers between x70 to x99