MODULE 4.1: KEY FEATURES

- ***Workbench***
- ***Presentation XML***
- ***Object resolution Descriptor (ORD)***

Objectives

- Describe the purpose of the NiagaraAX Workbench.
- Describe the purpose and benefits of the NiagaraAX Presentation XML.
- Explain the basic structure of an Object Resolution Descriptor (ORD), and recognize how an ORD can be viewed in the Workbench.

Workbench

This Windows®-based software application is the engineering tool that is used by most systems integrators to create a custom control solution for each customer. With the Workbench, a powerful user interface can be designed, without the need for line-by-line program code writing.

The Workbench defines the framework for building standardized user interfaces, and provides features such as:

- Standard layout with menu, toolbar, sidebars and view
- Standard browser-based navigation
- Bookmarking
- Tabbed browsing
- Ability to customize
- Both desktop and browser-based applications

Many Workbench functions can be performed through a web browser as well. However, a common operator's interface may only give a user the right to monitor and control real-time data and system status.

As needed, certain operators can also be granted rights to perform overrides and configuration changes to the system.

Presentation XML (Px)

Rather than giving a User access to the standard Workbench views, you may choose to design custom graphical pages using Niagara's Presentation XML module – "Px" for short.

- The **Px Editor** mode facilitates the custom design and configuration of all graphical elements, or "Widgets," which are highly customizable.
- The **Px Viewer** mode presents the real-time display to the End User. Hyperlinks make it possible for an Operator to seamlessly navigate throughout the system, no matter how large.

Presentation is a term that describes how Niagara enables information to be visualized across various media, such as: the Workbench, web browsers, handheld devices, pdf documents, etc.

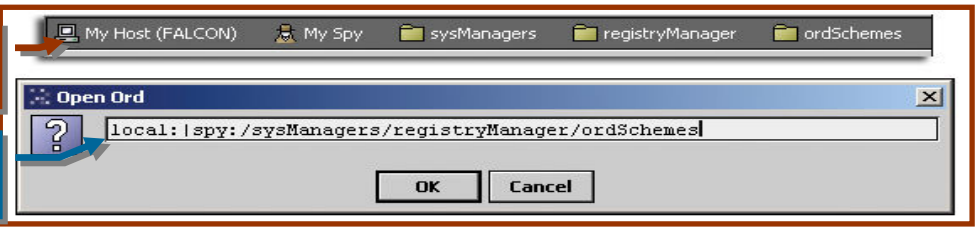
Widgets

Widgets are components that provide this visualization. You can use the Px Editor to work with widget properties in defining user interface functions for control and information display.

Visualization	Visualization refers to the display of information in the form of graphics, tables, charts, text fields, check boxes, trees, etc.
Make Widget Wizard	When you are designing Px views, a powerful tool called the "*Make Widget Wizard*" helps you to define the functional characteristics of a widget. This wizard-based presentation editor will save costly engineering time, especially when you make liberal use of standard naming conventions and screen layouts. This is the Reusability advantage that was discussed earlier in this course. The *Make Widget Wizard* launches whenever you drag and drop a Niagara object that has data onto the Px Editor workspace, and automatically binds the widget to the underlying data or control point with an ORD.
Object resolution Descriptor (ORD)	The fundamental navigation model of the Workbench is like a web browser. A web browser always has a current URL. As the current URL is changed, it obtains and displays the contents of that URL. A history of URLs is remembered allowing back & forward navigation. Most web browsers also allow the user to bookmark URLs for quick retrieval. The ORD is Niagara's universal identification system and is used throughout the Niagara framework. It unifies and standardizes access to all information, and is designed to combine different naming systems into a single string:

Open ORD Locator – visually displayed

Open ORD Dialog – textually displayed



With an ORD you can refer to the precise location of any object, file, view, or other resource.

Absolute ORD Format		
3 queries separated by vertical "pipe" symbols		
Host	**Session**	**Space**
local:	**fox: station:**	**slot:/Drivers/NiagaraNetwork/NewNiagaraStation/alarms**

TOOLS

Make Widget Wizard
Px Editor
Px Viewer
ORD Locator (Bar)

LAB

You will have opportunity to work with Ords and use the Make Widget Wizard in building a Px view in the Niagara Technical Certification Program on-site training course.

JOB AIDS

Key Features

LINKS

- ***Topic 4.1, Niagara Key Features & Building Blocks: Building Blocks***
- ***Module7, Workbench GUI (all topics)***

QUESTIONS FOR REVIEW:

1. *TRUE OR FALSE? Using the Workbench to design custom control solutions requires knowledge of some programming.*
2. *Which Px mode provides design and configuration capability?*
3. *What Workbench tool defines functional characteristics of visualization components?*
4. *What does the acronym ORD stand for?*
5. *TRUE OR FALSE? An ORD combines different naming systems into a single string.*
6. *TRUE OR FALSE? An ORD can be displayed either textually or visually.*
7. *Where can an ORD be viewed?*

MODULE 4.2: BUILDING BLOCKS

- ***Modules***
- ***Components***
- ***Views***
- ***Drivers***
- ***Stations***
- ***Platforms***
- ***.bog files***
- ***Palettes***

Objectives

- State the importance of Niagara's modular design approach.
- Define what components are and recognize their use in implementing NiagaraAX.
- Describe how to view components used to build NiagaraAX solutions using the Workbench.
- Explain what a driver is and how it enables the NiagaraAX framework to communicate and interact with specific external devices and networks.
- Differentiate between a station and a platform and state their basic roles in the NiagaraAX framework.

Module

The module is the most basic unit of software in the NiagaraAX framework.

This modular approach to Niagara's development improves the tracking of deployment and versioning of Niagara's features and new releases, and takes up less space because you can install only the modules you want.

- Used to release new versions of the Niagara software – as "release" modules
- Can deploy new modules using a *New Module Wizard*
- Updated modules can be distributed independently between major software releases

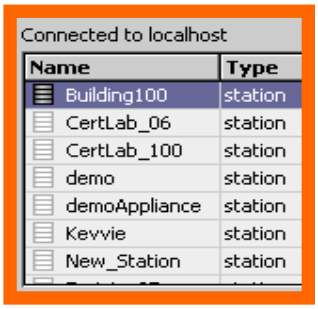
Component

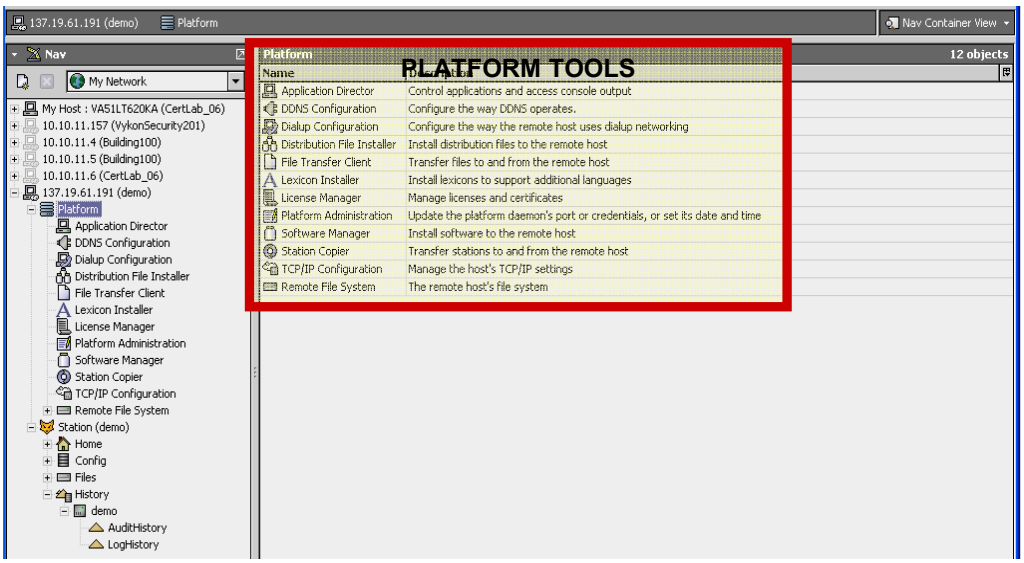
Don't confuse components with modules. Modules make up the Niagara software itself. Components are used to build Niagara solutions using the Workbench.

For the purposes of this course, any of the following objects are considered "components" --

- Any of the **8 object types** of the Common Object Model:
  Boolean, Numeric, Enum, and String (both read-only and writable)
- An object that acts as a **control point**, such as a pump or fan that an operator can override
- **Extensions** – that act to extend the functionality of a point
- **Triggers** – that provide periodic action
- A **device** that can be monitored and/or controlled such as a Lon or Bacnet controller used to control a pump or valve
- **Logic or Math components** – that process an input value and provide an output value
- A **network** such as a Lon or Bacnet network
- A **container** or **folder** that contains other components

Slots	Niagara components are defined as a series of **slots** that define both the characteristics (Properties) of a component and how those components will behave (Actions) when a user invokes a command or an event occurs.
View	The Workbench **"View Pane"** can display any component you select. Different views of components can provide different editing options. There are many ways to visualize your system and its components. A view is just a "visualization" of a component. These are but a few of the most common examples: **Wire Sheet View** **Property Sheet View** **Category Sheet View** **Slot Sheet View** **Link Sheet View** **Manager Views** **Px Viewer/Editor Views** **Other Views** Each component has a default view that appears whenever you activate a component (double-clicking, for example) without specifying a particular view.
Driver	A driver is a software program used to enable the Niagara^{AX} framework to communicate and interact with specific external devices and networks. Drivers handle device-specific features so that the operating system is freed from the burden of having to understand – and support – the needs of individual hardware devices. **All stations start with one driver network – the Niagara Network.** If you click the Drivers container in the View Pane or if you double-click the Drivers container in the Nav Tree, the graphical interface that appears is called the **Driver Manager.** It provides more details about the drivers that are currently in the station. It also allows you to add new driver networks to the station. A **driver network** represents a field-bus and a protocol through which the station will communicate with other equipment located across a field-bus, network, or communications port to which the station is connected. To create a new driver network, in the Driver Manager, click the "**New**" button and select the type of network desired from the drop-down menu. *This list of possible networks is determined based on the Niagara jar files present on your workstation.*

Station	A Niagara[AX] station is a program that runs on a Niagara[AX] platform; it is the main unit of server processing in the Niagara[AX] architecture. The station usually starts running immediately after the Niagara[AX] platform boots, and continues to run forever or until someone or something causes the platform to be turned off or rebooted.  Connected to localhost Name / Type Building100 station CertLab_06 station CertLab_100 station demo station demoAppliance station Kevvie station New_Station station  SERVICES DRIVERS CONTROL LOGIC
Config container	The **Config container** is visible in the Nav Tree and contains the services, drivers (networks and devices), and control logic (that you build). **SERVICES:** Services are essentially mini-programs that run within a station. These include a User Service, Category Service, Alarm Service, History Service, Backup Service, and Weather Service. Each station also has a limited set of platform services that provide a number of configuration platform views that do not require an active connection to the platform – just a station connection. **DRIVERS:** Though many drivers are available, the Drivers container in a specific station will only contain the network types actually used to implement the building automation and controls for that project. **CONTROL LOGIC:** Control logic is part of the software architecture that controls what the program will do. Custom control logic is required to provide building automation, energy management and access control for a specific project. The Workbench is used to design such custom controls.
Online vs. offline	The Workbench is used to both create and configure stations that are either online or offline. **ONLINE STATION:** A station that is currently running on a Niagara[AX] platform – a JACE controller, [AX]Supervisor PC or workstation **OFFLINE STATION:** A station that is not currently running (idle) on a Niagara[AX] platform
Running station	A running station provides access for client browsers to view and control the components (systems, sub-systems and devices) managed by the Niagara framework. The primary parts of a station include components and services. It is the combination of a database, a web server, and a control engine. The station either runs on a workstation, an [AX]Supervisor PC, or a JACE controller. **To see if a station is RUNNING or to START/STOP a station →** *Platform > Application director > Installed Applications area (top)*

Platform	A Niagara^{AX} platform" is the name for everything that is installed on a Niagara^{AX} host that is not part of a Niagara^{AX} station. The platform interface provides a way to address all the support tasks that allow you to setup, support and troubleshoot a Niagara^{AX} host. 
.bog file	A station database is defined by a single .bog file: **file: | station / {name} / config.bog** Stations are booted from their **config.bog file** into a single process on the host machine. A .**bog file** is a special file that describes and organizes components in a database for a variety of purposes. It can be a complete database or any collection of components. **Baja Object Graph** • Reside in the File space of a station • Typically given a ".bog" extension, although the ".palette" extension can be used to distinguish a bog designed for use as a palette; other than the extension, bog and palette files are identical in structure. A **config.bog file** describes a station's configuration.

Palette	A palette file is a bog file with a .palette extension. It is a **collection (or library) of components** in a hierarchical view that can be copied and pasted where needed – Wire Sheet, Property Sheet, Px View or Palette side bar. There are two types of palettes: **standard** and **personal**. STANDARD: Many standard modules in Niagara have their own palettes. Three standard modules with palettes that we saw earlier in the Extensibility topic are the control, alarm and history modules. PERSONAL: Niagara also gives you the ability to save previously developed solutions into **"personal" palettes** that can even be shared with co-workers. They are custom collections of components that you create and save for viewing in the palette side bar and that can be reused over and over. A personal palette allows you to save: ▪ Niagara objects ▪ Collections of objects ▪ Configurations ▪ Values ▪ Views ▪ Devices ▪ Widgets *Can you imagine having to recreate previous solutions for each project that had similar requirements? WHAT A TIME-SAVER TO BE ABLE TO SAVE PREVIOUSLY CONFIGURED SOLUTIONS – ALL IN A SINGLE LOCATION - **- YOUR PERSONAL PALETTE!*** *Both (standard and personal palettes) are viewed/accessed from the Palette Side Bar.* *Your personal palette will always be available in the Palette Side Bar, along with any standard module palettes (e.g., alarm, control, history) that you previously opened.*
 **TOOLS**	▪ ***Software Manager*** ▪ ***View Pane*** ▪ ***Interacting With Components*** ▪ ***Palette sidebar***
 **LAB**	*You will have opportunity to practice working with components, .bog files, drivers and palettes; using different Workbench views; connecting to, starting/stopping and creating new stations and platforms; and in the Niagara Technical Certification Program on-site training course.*
 **JOB AIDS**	***Niagara^{AX} Building Blocks*** ***Niagara Objects (Module 2.2)***

LINKS

- ***Topic 3.5, Niagara System Advantages: Reusability***
- ***Topic 5.1, Platforms & Stations***
- ***Topic 5.2, Platform Daemon***
- ***Topic 5.3, Connections: Platform vs. Station***
- ***Topic 5.4: Establishing a Platform Connection***
- ***Topic 5.5, Establishing a Station Connection***
- ***Topic 5.6, The Demo Station***
- ***Topic 5.7, Creating a New Station***
- ***Topic 6.2.1, Platform Tools: Application Director***
- ***Topic 7.3, Workbench GUI: View Pane***
- ***Topic 7.7, Workbench GUI: Locator Bar***

QUESTIONS FOR REVIEW:

1. *TRUE OR FALSE? A module is the same thing as a component.*
2. *Which of the following is an example of a component? Boolean object, extension, container, device, network*
3. *TRUE or FALSE? A slot assigns properties and actions to an object or component*
4. *TRUE OR FALSE? Drivers handle device-specific features so that the operating system is freed from the burden of having to understand and support the need of individual hardware devices.*
5. *TRUE OR FALSE? The Workbench can only configure stations that are currently running.*
6. *What are the primary parts of a station?*
7. *What is a platform? How does it differ from a station?*
8. *What is the purpose of a Config.bog file?*
9. *Where are personal palettes typically stored?*