Extensions

Extensions	Extensions provide building blocks to "extend" and change the behavior of individual control points, and they allow that behavior to be extended in a <u>consistent</u> manner.
	Extensions are Niagara's way of <u>adding functionality and additional processing requirements to a control point</u>. They allow plug-in functionality to support features such as **alarming, control** and **historical data** collection.
	Basic control points have no provision for alarming or historical trending. This keeps control points as lean as possible.
	NiagaraAX provides an extensible component model and rich libraries to enable programmers to <u>extend</u> and <u>enhance</u> the Niagara Framework and build their own unique applications and products.
	*These libraries are called **palettes**.*
Point extensions	⋗ Added as <u>dynamic properties</u> (slots) on a control point ⋗ <u>Process and modify the value</u> of a control point whenever it executes ⋗ Always invoked <u>in the order they are declared</u>
Proxy points	A "*proxy*" is used to locally <u>represent a remote object</u> – e.g., a control data point from a remote smart device.
	A **PROXY POINT** is any of the 8 control points (or objects). A data point from a remote platform becomes a proxy point when it is "discovered" and added to the Station database. By creating proxy points, the Niagara Framework is able to accurately represent (and manipulate) the data points available from remote devices.
Location of points	Points for any remote device reside inside the appropriate **Network** container under the **Drivers** container. They can be viewed on the <u>Nav Tree</u> and also on other views such as the <u>Wire Sheet</u> view.
Extension Categories	There are 3 categories of extensions: ▪ **Control** ▪ **Alarm** ▪ **History**
Control extensions	There are 3 types of control extensions: ▪ **DiscreteTotalizerExt:** tracks runtime and change of state count on equipment ▪ **NumericTotalizerExt:** accumulates numeric total using minutely or hourly totalization ▪ **ProxyExt:** indicates how a point's data originates, including details specific to the point's network and driver
	<u>EXAMPLE</u>: If you want to <u>track runtime on a pump</u>, you would add an extension to the object representing the pump inside a Niagara station.
	Control extensions can be found in the <u>control palette-Extensions folder</u>.

Alarm extensions	Use an alarm extension for any control point you want to <u>monitor</u> for **off-normal values**, and <u>show alarm indication</u> when a **limit or value** is met or exceeded. Examples include: • **OutOfRangeAlarmExt** • **BooleanChangeOfStateAlarmExt** • **BooleanCommandFailureAlarm** An important point to remember is that alarm extension types <u>must</u> match their parent component types. So, for example, you could not use a Boolean Change-of-State alarm extension that provides alarming based on a change of state – like ON or OFF – to alarm if pump discharge pressure drops below a specific value. In such a case, you WOULD use a Numeric Out-of-Range alarm extension. *<u>EXAMPLE</u>: If you want an **alarm to sound** <u>in the event that no pump is running</u>, you could add an alarm extension to either the pump object or the discharge pressure object. In this example, the extension – an Out-of-Range Alarm extension – originates inside the Alarm palette – under the "Extensions" folder.* *Alarm extensions can be found in the <u>alarm palette—Extensions folder</u>.*
History extensions	*<u>EXAMPLE</u>: If you want to **collect trend data** <u>every time a pump's discharge pressure changes significantly</u>, you would add an extension to the object representing the pump's discharge pressure. In this example, use a Numeric Change-of-Value or COV – extension.* *History extensions can be found in the <u>history palette—Extensions folder</u>.*