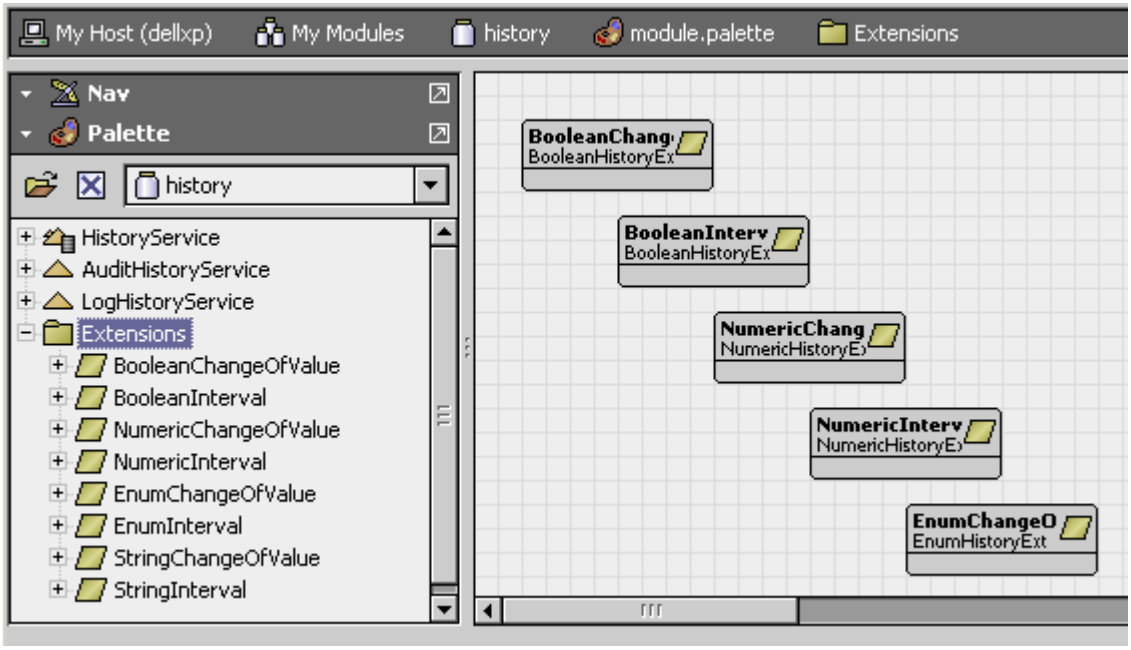
历史扩展（history extension）

添加历史扩展到任何您想要记录其历史数据的 point，也就是收集历史（history）。在 point 的 history 中，每个“Out” slot 收集的样本都有一个 date-timestamp（日期时间戳）、point status 和值（value）。Point 的历史数据（history data）既可以以表格（table）的形式查看，又可以以图表（chart）的形式查看。

您可以在调色板 **history：Extensions** 中找到历史扩展（history extension）。

图 92 History extensions（历史扩展）



每个历史扩展（history extension）具有的各种属性主要分成两类：

• Collector（收集器）属性，决定了怎样以及何时收集数据，比如 active time，以及（如适用）change tolerance 或收集间隔（interval）时间。

• HistoryConfig（历史配置）属性，决定了系统如何存储数据。这些属性包括历史记录名称（history name）、容量（capacity）以及完整策略（full policy）。

下表列出了所有历史扩展类型（history extension type）以及使用的 point 父节点。

表 12 历史扩展类型及其适用的点

历史扩展类型

适用的 point 类型

一般描述

Read-only（只读）

可写（writable）

BooleanChangeOfValue

BooleanPoint

BooleanWritable

根据布尔值和状态（status）的每次变化进行收集。

-

任何只有一个 Boolean Out 的对象，例如，kitControl： Logic 对象类型“And”

BooleanInterval

同上

同上

按照设置，根据重复时间间隔进行收集。

NumericChangeOfValue

NumericPoint

NumericWritable

根据数字量变化（在指定 tolerance 之外）或状态改变进行收集。

任何只有一个 Numeric Out 的对象，例如 kitControl：Math 对象类型“Add”

NumericInterval

同上

同上

按照设置，根据重复时间间隔进行收集。

EnumChangeOfValue

EnumPoint

EnumWritable

根据每次枚举的 state 或 status 变化进行收集。

—

任何只有一个 Enum Out 的对象

EnumInterval

同上

同上

按照设置，根据重复时间间隔进行收集。

StringChangeOfValue

StringPoint

StringWritable

根据字符串值或 status 变化进行收集。

-

任何只有一个 String Out 的对象

StringInterval

同上

同上

按照设置，根据重复时间间隔进行收集。
所有历史扩展（history extension）都具有一个可用的 Update History（更新历史）操作。该弹出菜单项目提供了重命名后刷新 History Id 的一种方法。它将 Name Format 数据区的 formatting 属性（用%号包围进行指定）应用到 History Config Id 的 Id。例如，如果 history 扩展下面的 History Name 属性被设置为%parent.name%，那么 History Config Id 最初会根据父节点的这个显示名称（parent display name）被命名。然而，如果您重命名了母节点（parent），History Config Id 属性却不会自动或立即改变。UpdateHistory Id action 会根据 formatting 属性触发对 HistoryConfig Id 的重命名，因此如果母组件（parent component）（在本例中）被修改了，那么 Update History Id action 将会修改 Id 属性。而且，如果与 history name 不一样，这也会导致对 history name 的修改。