机器网关服务

机器网关服务

机器网关服务为在同一个 OSGi 框架实例上运行的机器适配器提供了集中的配置、监控监测和管理。

通过通用 API 也展示子来自多个机器适配器的实时数据流，还可以用作服务多个客户端的数据聚合器。

机器网关服务的架构灵活，可扩展，这样，开发者可以针对实时数据流自定义适配器。默认提供 MQTT、OPC-UA、Modbus 和健康状况监控监测器机器适配器。

启动开机时，机器网关查询所有已安装机器适配器的 OSGi 服务注册表，并始终追踪其状态。`IMachineGateway` 接口允许客户端服务获得机器适配器和数据节点列表，并读取和写入数据。下图显示了机器网关服务的高级组件概览视图以及与其他服务的交互方式

依赖项

数据包 `com.ge.dspmicro.machinegateway.types` 以及数据包中的类已经被转移移动到新的 `device-common` bundle 中捆绑束。为这些类添加以下下列依赖项。

```
<groupId>com.ge.dspmicro</groupId> <artifactId>device-common</artifactId> <
version>{Predix Machine version}</version>
```

使用机器网关 API

查看机器网关 Javadoc API，了解如何实现机器网关服务。

找到并打开下列 API：<SDK installation location>/docs/apidocs/index.html/com.dspmicro.machinegateway.api。

使用机器网关示例应用程序

两个示例应用程序向你演示了如何构建自定义机器网关适配器。

1. 导航至<SDK installation location>/samples/sample-apps.zip 并提取文件。
2. 打开 sample-basicmachineadapter 和 sample-sample-subscriptionmachineadapter 应用程序。

   注：

   参见示例应用程序了解示例应用程序的构建和运行。

机器网关 Ping

从机器网关、机器适配器、数据订阅、Hoover Spillway 和处理器到 river 服务，机器网关 Ping 服务都会发送 PING 消息来确认连接。

机器网关 Ping 服务实现 `IValidateRoute` 接口。在调用 `ping (IPingMessage)` 方法之前， `IPingMessage` 实例应由 `Machine Gateway Ping` 服务通过调用 `createPingMessage(Map<String, Object> params, IPongNotification pongNotification)`方法创建。创建需要两个自变量：具有密钥的三个参数、适配器、订阅和 Spillway，这些在 `E2EPingMessage` 类中定义来识别网关，以及一个能够处理网关上的 pong 消息的 pong 通知回调函数。

如果一个或多个参数无效或丢失，PONG 消息会包含有效的选项来更改参数并再次输入。

依赖项

消费此服务需要 Maven 依赖项和 OSGi 导入：

- 消费机器网关 Ping 服务需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>validateroute-api</artifactId> <version>15.4.0</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入

```
Import Package: com.ge.dspmicro.validateroute.api;version="[1.0,2)"
```

获取机器网关 Ping 服务

使用声明式服务，注入机器网关 Ping 服务。

如何注入机器网关 Ping 服务的范例：

```
@Reference(type = '*') public void setMachineGatewayPing(IValidateRoute vali
dateRoute) { if (E2EPingMessage.ROUTE_ADAPTER_TO_DATA_RIVER.equals(validate
Route.getRouteName()) { //set service here this.machineGatewayPing = validat
eRoute; } }
```

数据存储和转发

数据存储和转发服务会持续存储机器、传感器和设备生成的数据并将其转发到云端。

Predix 机器收集机器、传感器和设备生成的数据并将其转发到云端。数据存储和转发服务能够持续存储数据，避免由于系统崩溃或网络连接中断导致的数据丢失。网络恢复时，会转发临时存储的数据。

通过数据存储和转发服务，你可以在加密的 H2 数据库文件中存储数据，并将其转发到目的位置。每个数据存储都在<Predix Machine runtime container location>/appdata/datastore/<storeName>中有一个独立的 H2 数据库文件。默认的数据库用户是 "datastore"。数据作为二进制大对象（BLOB）进行存储和转发。根据数据生产者和数据消费者之间的合同进行二进制数据的序列化和反序列化。

在默认情况下，数据库文件是加密和压缩文件。

使用数据存储和转发的 Spillway

数据存储和转发服务提供了当机器无法连接到目的位置时临时存储由机器适配器读取数据的方式~~，无法连接到目的位置时由机器适配器读取~~。当~~在~~连接恢复~~到目的位置~~之后，可以持续将数据转发到目的位置，便于被接收。

数据存储和转发是一项通用服务。Hoover 中的每个 Spillway 实例都有一个数据存储和转发实例，将数据存储并转发到 river 中。通过使用数据存储和转发服务的 Spillway 查看 Hoover 服务的以下数据流。

1. 每个 Spillway 都和一个数据存储和转发服务实例相关，对数据进行备份。
2. 一个 Spillway 只能使用~~对应~~一个数据存储和转发服务实例。如果引~~注~~入数据存储和转发实例，Spillway 会添加其转发回调函数。
3. 在处理器之后，发送到 river 之前，Spillway 始终将数据保存在数据存储中。
4. 转发线程从数据库中请求现有数据，如果可用的话，调用转发回调函数将数据转发到 river 中。
    a. 数据成功发出后，回调函数返回 SUCCESSFUL， 转发线程从数据存储中删除数据。转发线程继续将数据转发到接收的位置，直到数据库被清空。
    b. 如果 river 运行停止~~停机~~，回调函数返回 FAILED，数据还保存在数据存储中。转发线程会休眠一段配置的时间间隔之后再次尝试。同时，服务持续将新的数据保存到数据库中。

c. 如果数据无效，回调函数返回 FAILED_BUT_DELETE，记录日志~~记录信息消息~~，并从数据库中删除数据。

Spillway 和数据存储和转发服务的其他场景如下。

- 如果 Spillway 没有配置数据存储和转发服务，数据会直接转发到 river 中。
- 如果 Spillway 配置了数据存储和转发服务，但是数据存储和转发服务中没有注册转发回调函数，转发线程会记录出错信息，数据仍然存储在数据库中。
- 如果 Spillway 配置了数据存储和转发服务，但是没有找到数据存储和转发，会记录出错信息，数据被转发到 river 中。在这种情况下，如果数据存储中存在数据，无法确保数据的顺序。

下图展示了数据流。 图 1 使用数据存储和转发服务的 Spillway

限制

每个数据存储只允许有一个 `ForwardCallback` 实例。

依赖项

消费此服务需要 Maven 依赖项和 OSGI 导入包：

- 需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>storeforward-api</artifactId> <version>${dspmicro.storeforward.project.version}</version> </dependency>
```

- 消费捆绑束需要以下 OSGi 导入：

```
Import-Package: com.ge.dspmicro.storeforward.api;version="[1.0,2)"
```

获取数据存储和转发服务

使用声明式服务注入 `IStoreForward` 服务。
如何注入 `IStoreForward` 服务的范例：

```
import com.ge.dspmicro.storeforward.api; ...@Reference(type = '*') public void addStoreForward(IStoreForward storeForward) { //get instance whose name matches; }
```

配置数据存储和转发服务

你可以定制数据存储和转发服务的配置。

定制配置时，添加唯一的名称和描述，指定回调函数读取和转发数据的时间间隔。

格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 找到并打开下列文件：<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.storeforward-[n].config~~com.ge.dspmicro.httptunnel.client-[n].config file~~。
2. 设置以下属性的值：

属性	描述	类型	默认值	必需
com.ge.dspmicro.storeforward.storeName	服务的唯一名称。	字符串	"DefaultStoreForward"	是
com.ge.dspmicro.storeforward.forwardInterval	如果回调函数返回 FAILED，会显示读取和转发数据间隔的秒数。如果数据库中有数据，回调函数返回 SUCCESSFUL，转发线程会继续读取并转发数据，直到数据库被清空。	整型	I"10"	否

属性	描述	类型	默认值	必需
com.ge.dspmicro.storeforward.databasePassword	自动生成的数据库密码。用户不得更改密码。一旦更改密码，就无法打开现有数据库，数据库中的数据就会丢失。为了恢复错误密码打开数据库，必须删除 appdata/storeforward 文件夹中的数据库文件，这样才能创建新的数据库。	字符串		是
com.ge.dspmicro.storeforward.databaseEncryption	指定是否启用或禁用数据库加密。可以通过将这个属性设置为 false 或在配置文件中删除这个属性，禁用数据库文件加密。加密在 on 和 off 之间切换时，无法打开旧的数据库（如有），应删除。	布尔型	true	否
com.ge.dspmicro.storeforward.encryptionPassword	在默认的情况下，数据库文件加密密码和生成的数据库密码一致，在配置文件中加密。可以通过在配置文件中删除这一属性禁用数据库文件加密。加密在 on 和 off 之间切	字符串	和 databasePassword 相同	否

属性	描述	类型	默认值	必需
	换时，无法打开旧的数据库（如有），应删除。~~字符串和 databasePassword 相同~~			
com.ge.dspmicro.storeforward.databaseCompression	指定是否启用或禁用数据库压缩。可以通过将这个属性设置为 false 或在配置文件中删除这个属性，禁用数据库压缩。	布尔型	true	否
警告： 如果 GE 客户支持没有提出明确指导~~说明~~，请勿~~不得~~更改本警告下方的参数。				
com.ge.dspmicro.storeforward.dbSizeLimit	数据存储和转发服务停止~~向文件中~~写入数据时的~~的~~数据~~库~~库文件大小限值。默认值为 -1，代表无限的文件大小。	整型	l"-1"	否
com.dspmicro.storeforward.dbWriteResumeLimitMB	文件大小超过 dbSizeLimitMB 时，恢复写入数据库的数据库大小限值，暂停写入数据库。只有在文件大小低于 dbWriteResumeLimitMB 的值时，才能恢复写入。根据定义，这一数值必须低	整型	l"-1"	否

属性	描述	类型	默认值	必需
	于或等于 dbSizeLimitMB 的值。默认值匹配 dbSizeLimitMB 的值。			
com.ge.dspmicro.storeforward.compactIntervalSec	定义数据存储和转发服务检查文件大小的频率，如有必要，压缩数据库。默认值为 60 秒。	整型	I"60"	否
com.ge.dspmicro.storeforward.dbSizeLimitMBCompact	数据存储和转发服务尝试压缩数据文件的~~库文件的数据库文件~~大小限值。	整型	I"1"	否
com.ge.dspmicro.storeforward.purgeOnFullPercentage	数据库已满时，每个 compactIntervalSec 周期要放弃的最旧消息比例。	整型	I"0"	否
com.ge.dspmicro.storeforward.maxRetryLimit	在消息被丢弃前尝试发送消息的最多次数。默认为无限。	整型	I"-1"	否

使用数据存储和转发 API

查看数据存储和转发 Javadoc API 了解如何全面实现数据存储和转发服务。

1. 导航至<SDK installation location>/docs/apidocs.zip，并提取所有文件
2. 导航至数据存储和转发 Javadoc API<SDK installation location>/docs/apidocs/index.html/com.ge.dspmicro.storeforward.api。

使用数据存储和转发示例应用程序

为示范如何使用此服务，特提供一个数据存储和转发示例应用程序。

1. 导航至<SDK installation location>/samples/sample-apps.zip 并提取文件。
2. 打开 sample-datastoreforwardclient 应用程序。

   注：

   参见示例应用程序了解示例应用程序的构建和运行。

管理数据存储和转发实例

你可以在 Predix 机器网页控制台中管理数据存储和转发实例。

存储和转发页面列出了机器文件系统中出现的所有活动~~有效~~和非活动~~无效~~数据存储。

- 活动~~有效~~数据存储和运行的 `StoreForward` 服务实例相关。
- 非活动~~无效~~数据存储和运行的 `StoreForward` 服务实例不相关。

可以执行 Predix 机器网页控制台中的以下任务：

- 删除活动~~有效~~数据存储中最旧的条目
- 删除活动~~有效~~数据存储中的所有条目
- 删除非活动~~无效~~数据存储

删除数据存储中最旧的条目

启动你的 Predix 机器调试容器，打开 Predix 机器网页控制台。
你可以删除活动~~有效~~数据存储中最旧的条目。

1. 在 Predix 机器网页控制台中的 Predix 菜单上，点击存储和转发。
2. 在活动~~有效~~数据存储部分，点击数据存储。

   存储内容部分会出现数据存储中的最后 10 个数据条目。

3. 点击查看最旧的条目在数据存储中查看最旧条目的详细信息。

   只会显示数据的第一个 KB；如果数据无法读取，~~无法读取数据，~~就不会显示。

4. 点击删除最旧的条目。

删除活动~~有效~~数据存储中的所有条目

如果活动~~有效~~数据存储中的数据过于陈旧，或已经停止发送数据，可以删除数据存储中的内容。

启动你的 Predix 机器调试容器，打开 Predix 机器网页控制台。

1. 在 Predix 机器网页控制台中的 Predix 菜单上，点击存储和转发。
2. 在有效数据存储部分，点击数据存储。

   会出现数据存储中最旧的 10 个数据条目。

3. 点击删除最旧的条目删除数据存储中的所有数据条目。

删除非活动~~无效~~数据存储

删除非活动~~无效~~数据存储会永久性删除数据存储。

启动你的 Predix 机器调试容器，打开 Predix 机器网页控制台。

1. 在 Predix 机器 网页控制台中的 Predix 菜单上，点击存储和转发。
2. 在非活动~~无效~~数据存储部分点击~~无效~~数据存储的名称删除并点击删除数据存储。

   出现确认删除选项。

3. 点击删除来删除无效数据存储。

数据存储和转发 Ping

通过数据存储和转发 Ping 服务，你可以向数据存储中发送 PING 消息，将其保存在数据存储中，并转发到转发回调函数中来验证数据存储和转发路径。

`StoreForward` 服务实现 `IValidateRoute` 接口。在调用 Ping (`IPingMessage`) 方法之前，应由数据存储转发 Ping 服务创建一个 `IPingMessage` 实例，通过调用 `createPingMessage(Map<String,Object> params, IPongNotification pongNotification)`方法实现。 数据存储和转发路径不需要参数。需要 Pong 通知回调函数在路径上处理 PONG 消息。

依赖项

消费此服务需要 Maven 依赖项和 OSGi 导入：

- 需要下列 Maven 依赖项来消费数据存储和转发 Ping 服务：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>validateroute-api</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import Package: com.ge.dspmicro.validateroute.api;version="[1.0,2)"
```

获取数据存储和转发 Ping 服务

可以通过下列方式获取数据存储和转发 Ping 服务。

- 获取数据存储和转发 Ping 服务实例，然后调用 `StoreForward` 接口方法 `getValidateRoute()`。
- 使用声明式服务，注入数据存储和转发 Ping 服务。在默认情况下，具体的 `StoreForward` 实例的路径名称是存储名称和 `Route` 字符串的组合。

  下列代码展示了如何注入数据存储和转发 Ping 服务：

```
Example on how to inject the Store Forward Ping Service: @Reference(type = '*') public void setStoreForwardPing(IValidateRoute validateRoute) { if (YourStoreForwardRouteName.equals(validateRoute.getRouteName()) { //set service here this.storeForwardPing = validateRoute; } }
```

网关验证

使用网关验证服务从一个点到另一个点来验证任何需要有效网关连接的组件的网关连接性。(如，机器到机器或机器到云端）

功能

网关验证服务能够验证需要网关连接性和验证的组件之间的网关连接。它实现 `IValidateRoute` 接口。所有 River 服务都可以使用网关验证服务。

依赖项

在开始使用网关验证服务之前，你必须确保满足下列依赖项：

- Maven 依赖项

  消费网关验证服务需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>validateroute-api</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- OSGi Manifest.MF 导入

  消费捆绑束需要下列 OSGi 导入：

```
Import-Package: com.ge.dspmicro.validateroute.api;version="[1.0,2)"
```

创建网关验证 Ping 请求

你可以在 Predix 机器网页控制台上使用网关验证服务。网关验证服务能够使用 PING 验证路径。

注：

生成的 Predix 机器运行时容器必须是 Predix 机器调试版本，或你必须使用 Predix SDK 手动添加网页控制台功能。

1. 启动 Predix 机器。
   a. 通过命令行导航至<Predix Machine runtime container location>/bin 目录。
   b. 使用以下其中一条命令启动 Predix 机器：

      start_predixmachine.bat（Windows）

      start_predixmachine.sh（Linux 和 Mac）

2. 启动 Predix 机器网页控制台。
   a. 打开网页浏览器并输入下列 URL：

```
https://localhost:8443/system/console/
```

出现登录提示。

b. 输入你的用户名和密码。Predix 机器使用以下默认值：

用户名：`predix`

密码：`predix2machine`

注：

首次登录时，你必须更改密码。新密码必须为：

- 长度至少为 8 个字符
- 至少包含一个小写字母
- 至少包含两个数字
- 密码中不得包含用户名
- 以字母开始和结束

3. 在 Predix 菜单上点击网关验证。

4. 点击+图标创建 PING 请求。

5. 完成设置 PING 请求后，点击 Ping 按钮。

你会收到 SUCCESSFUL 或 FAILED 状态消息。查看 FAILED 消息的内容对 PING 相关问题进行故障排除。

6. 如果要删除 PING 请求，可以在 网关 Ping 请求部分点击 图标。

7. 如果要重复 PING，可以点击 图标。

健康状况监测器机器适配器

健康状况监测器机器适配器是一款在 Predix 机器运行时容器内作为 OSGI 捆绑束运行的数据获取适配器，在监测机器上 OS 级健康状况的衡量标准。

它实现 `ISubscriptionMachineAdapter` 接口，由机器网关服务管理。

下图显示了整体机器网关和适配器架构，包括健康状况监测器机器适配器： 图 2 健康状况监测器机器适配器架构

消费健康状况监测器机器适配器

和其他适配器一样，健康状况监测器机器适配器同样被机器网关服务~~和其他适配器一样~~消费~~健康状况监测器机器适配器~~。你可以通过使用 `AdapterType` 属性在机器网关适配器的返回列表中搜索适配器，该属性在 `MachineAdapterInfo` 类中。

注：

参见机器网关了解如何获取适配器。
使用 `ISubscriptionMachineAdapter` 接口与健康状况监测器机器适配器进行交互。

限制

健康状况监测器机器适配器为以下健康状况类别返回数据：

- CPU 使用。
- 内存使用。
- 硬盘使用（通过文件系统）。
- ProSyst Predix 机器 JVM 内存使用。

配置健康状况监测器机器适配器

你可以配置健康状况监测器机器适配器的实现。

你必须指定健康状况监测器机器适配器 XML 文件的路径，并提供适配器的名称和描述。格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型

F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 找打并打开下列文件：<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.predixcloud.identity.config。
2. 设置以下属性的值：

属性	**描述**	**类型**	**默认值**	**必需**
com.ge.dspmicro.machineadapter.healthmonitor.configFile	健康状况监测器机器适配器XML文件的路径。 如果这个属性为空	字符串	"<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.machineadapter.healthmonitor.xml"	否，但如果这个属性为空，适配器被禁用。

属性	描述	类型	默认值	必需
	，适配器被禁用。			
com.ge.dspmicro.machineadapter.healthmonitor.name	适配器的人类可读名称。	字符串	"healthmonitor"	?
com.ge.dspmicro.machineadapter.healthmonitor.description	适配器的人类可读描述。	字符串	"Health\ Monitor"	?

健康状况监测器机器适配器 XML 配置文件描述了适配器连接节点和订阅。同时提供示例文件和容器。其读取节点和订阅是静态配置的。

下列范例是健康状况监测器机器适配器 XML 配置文件。 `nodeTypes` 是当前支持的唯一类型，不得改变这些数值。你可以改变 `updateInterval` 值。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <healthMonitorMachi
neAdapterConfig> <name>Machine health monitor nodes</name> <description>Mac
hine health monitor nodes</description> <dataNodeConfigs> <dataNode name="No
de-1-1" nodeType="SYSTEM_CPU_USAGE" dataType="DOUBLE" description="CPU Usag
e" /> <dataNode name="Node-2-1" nodeType="SYSTEM_MEMORY_USAGE" dataType="DO
```

```
UBLE" description="Memory Usage" /> <!-- If only one file system is availabl
e, you do not have to specified drive attribute.By default, the first one is
 returned.--> <dataNode name="Node-3-1" nodeType="SYSTEM_DISK_USAGE" dataTy
pe="DOUBLE" description="Disk Usage Example" /> <!-- For Windows, drive name
 is constructed as <drive friendly name><space><(drivename:)> For Linux, dri
ve name is constructed as <mounted name><space><(Filesystem name)> --> <!--
The followings are examples on how to specify drive attribute <dataNode name
="Node-3-1" nodeType="SYSTEM_DISK_USAGE" dataType="DOUBLE" drive="Windows7
(C:)" description="Windows Disk Usage Example" /> <dataNode name="Node-3-2"
nodeType="SYSTEM_DISK_USAGE" dataType="DOUBLE" drive="/ (/dev/mapper/ubuntu
--vg-root)" description="Linux Disk_Usage Example" /> <dataNode name="Node-3
-3" nodeType="SYSTEM_DISK_USAGE" dataType="DOUBLE" drive="/mnt (192.168.30.
66:/export)" description="Linux Disk_Usage Example" /> --> <dataNode name="N
ode-4-1" nodeType="JVM_MEMORY_USAGE" dataType="DOUBLE" description="JVM Mem
ory Usage" /> </dataNodeConfigs> <!-- updateInterval is in seconds.--> <data
SubscriptionConfigs> <dataSubscriptionConfig name="CPU Memory Usage" update
Interval="60" > <nodeName>Node-1-1</nodeName> <nodeName>Node-2-1</nodeName>
 <nodeName>Node-4-1</nodeName> </dataSubscriptionConfig> <dataSubscriptionC
onfig name="Disk Usage" updateInterval="1200" > <nodeName>Node-3-1</nodeNam
e> </dataSubscriptionConfig> </dataSubscriptionConfigs> </healthMonitorMach
ineAdapterConfig>
```

使用健康状况机器适配器示例应用程序

提供健康状况及其适配器示例应用程序的目的是展示如何使用这项服务。

1. 导航至<SDK installation location>/samples/sample-apps.zip 并提取文件。
2. 找到并打开<SDK installation location/samples/sample-apps/sample/sample-healthmachineadapter 应用程序。

   注：

   参见示例应用程序了解示例应用程序的构建和运行。

Hoover 服务

Hoover 捆绑束管理多个 Spillways 从机器网关收集数据，然后运用处理器处理数据并通过 river 发送到目的位置，或通过 HTTP River 发送到 Predix 云端。

Hoover Spillway 从机器网关收集数据，然后进行处理，决定是通过 river 还是 HTTP River 发送数据。如果 Hoover Spillway 配置规定了 HTTP River 或 WebSocket River 目的位置，通过 HTTP River 或 WebSocket ~~网络套接字~~ River 将数据发送到 HTTP 数据服务或 Predix 云端的时间序列数据库。

处理器是一个在发送之前处理数据的例程。处理包括筛选、改变数据格式和转换数据。

可以通过不同的 `SubscriptionMachineAdapter` 类型从不同的传感器中收集数据。数据的类型和格式可能各不相同。

下图展示了 Hoover 捆绑束如何处理并向 HTTP River 发送数据。

使用 Hoover API

查看 Hoover 处理器和 Hoover Spillway Javadoc API 了解如何实现 Hoover 服务。

1. 导航至<SDK installation location>/docs/apidocs.zip，并提取所有文件
2. 导航至处理 API<SDK installation location>/docs/apidocs/index.html/com.ge.dspmicro.hoover.api.processor 和 <installation location>/docs/apidocs/index.html/com.ge.dspmicro.hoover.api.spillway。
3. 导航至 Spillway API<SDK installation location>/docs/apidocs/index.html/com.ge.dspmicro.hoover.api.spillway。

配置 Hoover 服务

你可以通过指定 Spillway 的名称和描述，列出数据订阅者和目标 river 的名称来定制 Hoover 服务的配置。

格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 找到并打开配置文件：<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.hoover.spillway-{n}.config
2. 设置以下属性的值：

属性	描述	类型	默认值
com.ge.dspmicro.hoover.spillway.name	spillway 唯一的名称。	字符串	"DefaultSpillway"
com.ge.dspmicro.hoover.spillway.description	（可选）spillway 的简短描述	字符串	Simple test spillway
com.ge.dspmicro.hoover.spillway.dataSubscriptions	用管道符("|")分隔的 DataSubscription 名称列表。	字符串列表	Temperature_Subscription | Pressure_Subscription | OPCUA_Subscription_1 | OPCA_Subscription_2
com.ge.dspmicro.hoover.spillway.destination	river 的名称。	字符串	Sender Service
com.ge.dspmicro.hoover.spillway.process Type	（可选）处理器中定义的处理数据的业务逻辑类型。	字符串	
com.ge.dspmicro.hoover.spillway.storeforward	与此 spillway 相关的 StoreForward 的名称。	字符串	DefaultStoreForward

使用 Hoover 示例应用程序

为了示范如何使用该服务，特提供一个 Hoover 示例应用程序。

1. 导航至<SDK installation location>/samples/sample-apps.zip 并提取文件。
2. 在 sample-apps/sample 文件夹内，打开 sample-hoover 应用程序。

注：

参见示例应用程序了解示例的构建和运行。

Modbus 机器适配器

Modbus 机器适配器是一款基于 Modbus 标准的数据采集组件。

它会发现并连接到 Modbus 节点，浏览节点的地址空间，采用不同的数据访问机制读/写数据。

提示：

了解 Modbus 更多信息，参见 http://www.modbus.org/。

作为 OSGi 服务，Modbus 机器适配器在 Predix 机器运行时容器中的项部运行，并支持 `ISubscriptionMachineAdapter` 接口。Modbus 机器适配器由机器网关服务管理。和其他适配器一样，可以通过机器网关服务消费。可以使用 `Adapter Type` 属性在机器网关适配器的返回列表中找到，该属性在 `MachineAdapterInfo` 类中。

Modbus 标准协议支持 TCP/IP 和串口行通信，但是对于在 Predix 机器中支持串口行通信的 Modbus 机器适配器，你必须将 RXTX 及其内嵌的本地库部署到 Predix 机器运行时容器中。*

*RXTX 库不支持一些平台。下表指出了与某些平台共同合作的第三方软件库的构建位置：

平台	下载链接	说明
Windows x64	https://bitbucket.org/jlauer/mfz-cdn/downloads/mfz-rxtx-2.2-20081207-win-x64.zip	参见 http://rxtx.qbang.org 了解兼容性。
Raspberry Pi II (ARM7 Linux)	https://blogs.oracle.com/jtc/resource/rxtx-arm7l/librxtxSerial-2.1-7.so	本链接只提供本地库。RXTXComm.jar 可以运行。 可将本地库重命名（或软链接至）为 librxtxSerial.so

1. 从 http://rxtx.qbang.org/pub/rxtx/rxtx-2.2pre2-bins.zip 下载 RXTX 库压缩文件。
2. 在下列位置创建一个 rxtx 目录：<Predix Machine runtime container location>machine/lib
3. 将文件解压至<Predix Machine runtime container location>machine/lib/rxtx 文件夹中。

   在 rxtx 目录中创建 rxtx-2.2pre2-bins 目录。会产生这样的目录结构：

   注：

   你只需为你想支持的平台添加目录。例如，如果你想在 Linux 上运行，无需包括 mac-10.5、win64 或 win32 目录。你必须始终包括 RXTXcomm.jar 文件。

下图说明了如何在机器网关的整体架构中安装 Modbus 机器适配器。 图 3Modbus 机器适配器架构

Modbus 机器适配器功能性

Modbus 机器适配器支持下列功能：

- 下载 Modbus 机器适配器 XML 配置文件读取下列信息：
  - 连接信息
  - 数据节点
  - 订阅节点
  - 发布间隔
- 用机器网关注册：
  - 展示配置。
  - 展示适配器状态。
- 使用 TCP 和串口行传输连接到 Modbus 节点。
- 数据读写操作。
- 订阅寄存器值属性。

限制

由于 Modbus 协议的性质，没有安全功能。如果不是绝对必要，解决方案中不应使用 Modbus。

- 注意，数据没有加密，没有基于角色的访问控制或认证机制。
- 确保不允许外部访问 Modbus 服务器。
- 确保环境与外部网络访问隔离开来。

使用 Modbus 机器适配器 API

查看 Modbus 机器适配器 API，了解如何实现 Modbus 机器适配器。

1. 导航至<SDK installation location>/docs/apidocs.zip，并提取所有文件
2. 导航至 Modbus Javadoc API<SDK installation location>/docs/apidocs/index.html/com.ge.dspmicro.machineadapter.modbus.api 。

配置 Modbus 机器适配器

你可以在 XML 配置文件中配置连接节点和订阅。

你可以在每个实例的配置（.config) 文件中配置 XML 配置的名称、描述和位置。

格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 为了配置 Modbus 机器适配器连接节点和订阅的实例，编辑 XML 配置文件：<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.machineadapter.modbus-[n].xml。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <modbusMachin
eAdapterConfig> <name>Onsite monitor modbus nodes</name> <descriptio
n>Onsite monitor modbus nodes</description> <dataNodeConfigs> <!-- RE
MOVE THIS LINE FOR TCP/IP <channel protocol="TCP_IP" tcpIpAddress="12
7.0.0.1" tcpIpPort="502"> <unit id="1"> <register name="Node-1-1" dat
aType="INTEGER" address="10" registerType="HOLDING" description="temp
erature"/> <register name="Node-1-2" dataType="DECIMAL" address="11"
registerType="HOLDING" description="pressure"/> </unit> <unit id="2">
 <register name="Node-2-1" dataType="INTEGER" address="20" registerTy
pe="HOLDING" description="temperature"/> <register name="Node-2-2" da
taType="INTEGER" address="21" registerType="HOLDING" description="pre
```

```
ssure"/> </unit> </channel> REMOVE THIS LINE --> <!-- REMOVE THIS LIN
E FOR SERIAL <channel protocol="SERIAL" encoding="RTU" portName="COM1
" ba4udRate="9600" parity="NONE"> <unit id="3"> <register name="Node-
3-1" dataType="INTEGER" address="30" registerType="HOLDING" descripti
on="temperature"/> <register name="Node-3-2" dataType="INTEGER" addre
ss="31" registerType="HOLDING" description="pressure"/> </unit> <unit
 id="4"> <register name="Node-4-1" dataType="INTEGER" address="40" re
gisterType="INPUT" description="temperature"/> <register name="Node-4
-2" dataType="INTEGER" address="41" registerType="INPUT" description=
"pressure"/> </unit> </channel> REMOVE THIS LINE --> </dataNodeConfig
s> <!-- updateInterval 和 startPointOffset are in seconds.If startPoin
tOffset == -1, it will start immediately --> <dataSubscriptionConfig
s> <!-- REMOVE THIS LINE FOR TCP/IP <dataSubscriptionConfig name="Tem
perature_Subscription" updateInterval="60" startPointUnit="MINUTES" s
tartPointOffset="10"> <nodeName>Node-2-1</nodeName> <nodeName>Node-1-
1</nodeName> </dataSubscriptionConfig> REMOVE THIS LINE --> <!-- REMO
VE THIS LINE FOR SERIAL <dataSubscriptionConfig name="Pressure_Subscr
iption" updateInterval="3600" startPointUnit="HOURS" startPointOffset
="600"> <nodeName>Node-3-2</nodeName> <nodeName>Node-4-2</nodeName> <
/dataSubscriptionConfig> REMOVE THIS LINE --> </dataSubscriptionConfi
gs> </modbusMachineAdapterConfig>
```

2. 如果要给Modbus机器适配器的每个实例配置XML配置文件的名称、描述和位置，可以在<Predix Machine runtime container location>/configuration/machine/com.ge.dspmicro.machineadapter.modbus-[n].config 文件中编辑属性。

属性	描述	类型	默认值	必需
com.ge.dspmicro.machineadpater.modbus.description	适配器的人类可读描述。	字符串	“支持 Modbus 节点的基本读/写功能。支持订阅一组 Modbus 节点。"	否
com.ge.dspmicro.machineadapter.modbus.name	适配器的	字符串	"Modbus 机器适配器"	是

属性	描述	类型	默认值	必需
	名称。			
com.ge.dspmicro.machineadapter.modbus.nodeCommMaxTries	在抛出异常之前尝试读/写至设备的最大数目。最小数值为2，这样，如果读写失败，可以回收	整型	I"2"	否

属性	描述	类型	默认值	必需
	缓存连接。 如果重新输入后无法到达设备，之前检索到的缓存数据会返回BAD质量。 如果没有缓存			

属性	描述	类型	默认值	必需
	的数据，记录异常和数值0，返回BAD质量。			
com.ge.dspmicro.machineadapter.modbus.configFile	Modbus XML配置文件的路径。 重要提示：如	字符串	"configuration/machine/com.ge.dspmicro.machineadapter.modbus-0.xml"	是

属性	描述	类型	默认值	必需
	果此属性保留为空，适配器被禁用。			
com.ge.dspmicro.machineadapter.modbus.subscriptionThreadPoolSize	处理订阅从节点检索值的线程池大小。最小值为1。	整型	I"5"	否

通用~~普通~~信道配置

下列属性通常用于定义如何访问设备：

通用普通信道属性	默认值	有效值	必需	备注
protocol		TCP_IP, SERIAL	是	**Tcp/Ip 与串行通信协议** 指定将用于连接 Modbus 设备的通信协议。
regBaseAddress	0	0,1	否	**基于 0 与 1 的寻址** 在设备的配置文件中规定寄存器地址的编号约定。从 0 或 1 开始。如果是基于 1 的寻址，当帧被构造成与 Modbus 设备通信时，用户输入的地址将减去 1。
bitBaseAddress	0	0,1	否	**寄存器内基于 0 与 1 的位寻址** 允许寄存器内的位（16 位）作为布尔值引用的内存类型。寄存器内的位寻址提供了在给定寄存器中位寻址的两种方法：基于 0 和基于 1。寄存器内基于 0 的位寻址代表第一位从 0 开始。寄存器内基于 1 的位寻址代表第一位从 1 开始。
defaultModbusByteOrder	true	true, false	否	**默认 Modbus 字节顺序** 通过这个选项，Modbus 适配器使用的字节顺序可以从默认的 Modbus 字节排序（大端）变为 Intel 字节排序（小端）。默认情况下，这是 Modbus 兼容设备的正常设置。如果设备使用 Intel 字节排序，将此选项设置为 false 能够让 Modbus 驱动程序正确读取 Intel 格式的数据。
first16BitLow	true	true, false	否	**32 位中前 16 位低位的数**

通用普通信道属性	默认值	有效值	必需	备注
				据类型 Modbus 设备中的两个连续寄存器地址用于 32 位数据类型，如 Integer 和 Float。可以指定适配器是否应该假定前 16 位是 32 位值的低位还是高位字。默认情况下，第一个字低位，接下来是 Modicon Modsoft 编程软件约定。这也适用于 64 位中两个 32 位数据的数据类型，如 Long 和 Double。
`first32BitLow`	true	true, false	否	**64 位中前 32 位低位的数据类型** Modbus 设备中的四个连续寄存器地址用于 64 位数据类型，如 Long 和 Double。可以指定驱动程序是否应该假定前 32 位是 64 位值的低位还是高位字。在默认情况下，前 32 位是低位，遵循默认的 32 位数据类型约定。
`mostSigBit`	false	true, false	否	**使用 Modicon 位排序** 指定最有效位是否是位 0/1。当这个选项为 true 时，适配器会在读写寄存器上调转位顺序，遵循 Modicon Modsoft 编程软件约定。例如，启用这个选项时，写入地址 40001.0/1 会影响到设备中的位 15/16。这个选项默认为 false。例如，`bitBaseAddress=0` 和 `mostSigBit=true` 对于值 0x1357，第 1 位是 0，第 15 位是 1。

通用~~普通~~信道属性	默认值	有效值	必需	备注
				例如，`bitBaseAddress=0` 和 `mostSigBit=true` 对于值 0x1357，第 1 位是 0，第 15 位是 1。 例如，`bitBaseAddress=0` 和 `mostSigBit=false` 对于值 0x1357，第 1 位是 1，第 15 位是 0。

TCP/IP-相关~~具体~~信道配置

下列属性专用于 TCP/IP 通信。

TCP/IP 信道属性	必需	备注
`tcpIpAddress`	是	规定 Modbus 从属设备的 IP 地址或主机名称。
`tcpIpPort`	是	规定 Modbus 从属设备的端口编号。通常会使用 502。

串行-相关~~具体~~信道配置

下表中列出了串行通信的属性。

串行信道属性	默认值	有效值	必需	备注
`encoding`	RTU	RTU，ASCII	否	串行 Modbus 连接可以使用两种基本传输模式，ASCII 或远程终端(RTU) 。串行通信中的传输模式定义了 Modbus 消息的编码方式。通过 Modbus/ASCII， 消息是可读的 ASCII 格式。Modbus/RTU 格式采用二进制编码，在监测时无法读取消息但是却缩减了每个消息的大小。这样可以同时交换更多数据。一个 Modbus 网络段上的所有节点必须使用相同的串行传输模式。配置使用 Modbus/ASCII 的设备无法理解

<table>
<tr><th>串行信道属性</th><th>默认值</th><th>有效值</th><th>必需</th><th colspan="5">备注</th></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="5">Modbus/RTU 中的消息，反之亦然。根据编码模式，可以根据下列方式设置起始位和数据位。根据奇偶校验值，可根据下表设置停止位。</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>RTU</td><td colspan="2">ASCII</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>字符</td><td>二进制 0 ...255</td><td colspan="2">ASCII 0..9 和 A..F</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>起始位</td><td>1</td><td colspan="2">1</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>数据位</td><td>8</td><td colspan="2">7</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>奇偶校验</td><td>EVEN/ODD</td><td>无</td><td>EVEN/ODD</td><td>无</td></tr>
<tr><td></td><td></td><td></td><td></td><td>停止位</td><td>1</td><td>2</td><td>1</td><td>2</td></tr>
<tr><td>portName</td><td></td><td></td><td>是</td><td colspan="5">指定通信的串行端口名称。</td></tr>
<tr><td>baudRate</td><td></td><td></td><td>是</td><td colspan="5">指定在通信信道中传输信息的速率。</td></tr>
<tr><td>奇偶校验</td><td>无</td><td>ODD, EVEN, NONE</td><td>否</td><td colspan="5">一个奇偶校验位、校验位或添加到二进制代码字符串后面的位，可以显示值为 1 的字符串的位数是奇还是偶。</td></tr>
</table>

寄存器配置

使用以下属性配置节点。

寄存器属性	默认值	有效值	必需	备注	
name			是	节点唯一的名称	

<table>
<tr><th>寄存器属性</th><th>默认值</th><th>有效值</th><th>必需</th><th colspan="4">备注</th></tr>
<tr><td>dataType</td><td></td><td>BOOLEAN, BYTE, SHORT, INTEGER, LONG, FLOAT, DOUBLE, STRING</td><td>是</td><td colspan="4">节点的数据类型。</td></tr>
<tr><td>address</td><td></td><td></td><td>是</td><td colspan="4">寄存器地址。取决于 regBaseAddress 的值，地址可以从 0 或 1 开始。</td></tr>
<tr><td rowspan="4">registerType</td><td rowspan="4"></td><td rowspan="4">HOLDING, INPUT, COIL, DISCRETE</td><td rowspan="4">是</td><td>寄存器类型</td><td>对象类型</td><td>类型</td><td>备注</td></tr>
<tr><td>HOLDING</td><td>一个或多个 16 位字</td><td>读-写</td><td>这种数据类型可以由应用程序更改。</td></tr>
<tr><td>INPUT</td><td>一个或多个 16 位字</td><td>只读</td><td>这种数据类型可以由 I/O 系统提供。</td></tr>
<tr><td>COIL</td><td>单独位</td><td>读-写</td><td>这种数据类型可以由应用程</td></tr>
</table>

寄存器属性	默认值	有效值	必需	备注			
							序更改。
				DISCRETE	单独位	只读	这种数据类型可以由 I/O 系统提供。
description			否	节点的描述。			
count	1		否	仅针对 STRING 数据类型有效，规定访问字符串的寄存器数量。			
bitIndex	1	0~15（如果 bitBaseAddress=0）；1~16（如果 biBaseAddress=1）	否	仅针对 HOLDING 和 INPUT 寄存器的 BOOLEAN 数据类型有效。			

寄存器和数据类型

Modbus 机器适配器支持下列寄存器和数据类型：

寄存器类型	访问	数据类型	位	寄存器数量	备注
HOLDING	读 写	布尔型	1	1	bitIndex 属性规定访问哪一位
		字节	8	1	访问最低有效字节
		短整型	16	1	
		整型	32	2	
		长整型	64	4	

寄存器类型	访问	数据类型	位	寄存器数量	备注
		浮点数	32	2	
		双精度型	64	4	
		字符串		取决于 `count` 属性的值	
INPUT	只读	布尔型	1	1	`bitIndex` 属性规定访问哪一位
		字节	8	1	访问最低有效字节
		短整型	16	1	
		整型	32	2	
		长整型	64	4	
		浮点数	32	2	
		双精度型	64	4	
		字符串		取决于 `count` 属性的值	
COIL	读 写	布尔型	1		
DISCRETE	只读	布尔型	1		

MQTT 客户端

消息队列遥测传输 （MQTT）是发布/订阅模型中使用的简单、轻量的消息协议。

专为受限设备和低带宽、高延迟或不可靠的网络而设计。使用 MQTT 客户端向代理发布消息或订阅一个主题来接收消息。

注：

了解 MQTT 的更多信息，参见 http://mqtt.org/。
MQTT 客户端展示了 Paho Java 客户端中定义的所有接口。Paho Java 客户端是 Java 用来开发在 Java 虚拟机（JVM）或其他 Java 兼容平台上运行的应用程序而写入的 MQTT 客户端库。Paho Java 客户端提供了两个 API：

- `MqttAsyncClient`：一个完全异步的 API，通过注册的回调函数通知活动完成。
- `MqttClient`：一个关于 `MqttAsyncClient` 的同步封装函数，让函数与应用程序同步。

依赖项

消费 MQTT 客户端需要 Maven 依赖项和 OSGi 导入：

- 消费 MQTT 客户端需要下列 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>mqtt-client</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import-Package: org.eclipse.paho.client.mqttv3;version="[1.0,2.0)", org.eclipse.paho.client.mqttv3.persist;version="[1.0,2.0)"
```

使用 MQTT 客户端示例应用程序

为示范如何使用此服务，特提供一个 MQTT 客户端示例应用程序。

1. 导航至<SDK installation location>//sample/sample-apps.zip 并提取文件。
2. 在 sample-apps/sample 文件夹内，打开 sample-mqttclient 应用程序。

   注：

   关于构建和运行示例应用的说明参见在 Predix SDK 内构建示例和运行示例。

MQTT 机器适配器

MQTT 机器适配器在被机器网关消费之前从 MQTT 代理接收数据。

根据 MQTT 协议，MQTT 机器适配器作为一个 OSGi 服务运行在 Predix 机器运行时容器中~~的项部作为 OSGi 服务运行~~，并支持 `ISubscriptionMachineAdapter` 接口。使用 `Adapter Type` 属性，MQTT 机器适配器出现在机器网关适配器的返回列表中，该属性在 `MachineAdapterInfo` 类中。

MQTT 机器适配器订阅 MQTT 代理接收消息。代理将消息发布到适配器并通知其订阅者。MQTT 代理将消息转换并发送到 MQTT 机适配器，作为 `PDataValue` 类中的 JAON 字符串。

因为 MQTT 代理在通知订阅听众之前发布至适配器，所以不支持从适配器直接读取。此外，不支持订阅时间间隔。

下图显示了架构概览以及从 MQTT 代理到订阅者，然后到 MQTT 适配器的消息流~~高级架构和流程~~。

配置 MQTT 机器适配器

如果要配置 MQTT 机器适配器，可以在.XML 配置文件中定义其连接节点和订阅。容器内有一个示例。

- 在该文件中静态配置所有节点和订阅。
- 每个节点包括 `name`、 `topic`、 `qos`。你也可以规定收到的数据是否是 `PDataValue` 列表中的 `serializedData`
    - 如果数据是序列化的数据，MQTT 机器适配器–会对数据进行反序列化，并告诉听众 `serializedData`。
    - 在默认的情况下， `serializedData` 的值为 `false`，这时，MQTT 机器适配器将数据作为字符串处理。

为了配置 XML 配置文件的名称、描述和位置，在配置 (.CONFIG) 文件中为每个实例设置属性值。

1. 在 XML 配置文件中配置 MQTT 机器适配器连接节点和订阅的实例，<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.machineadapter.mqtt-[n].xml。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <mqttMachineA
dapterConfig> <name>MQTT nodes</name> <description>MQTT nodes and sub
scriptions</description> <dataNodeConfigs> <dataNode name="Node-1" to
pic="TestTopic1" qos="0" serializedData="false" description="TestTopi
c 1 with Qos 0" /> <dataNode name="Node-2" topic="TestTopic2" qos="1"
 description="TestTopic 2 with Qos 1" /> <dataNode name="Node-3" topi
c="TestTopic3" qos="2" description="TestTopic 3 with Qos 2" /> </data
NodeConfigs> <dataSubscriptionConfigs> <dataSubscriptionConfig name="
Analytics Subscription"> <nodeName>Node-1</nodeName> <nodeName>Node-2
</nodeName> </dataSubscriptionConfig> </dataSubscriptionConfigs> </mq
ttMachineAdapterConfig>
```

2. 通过在配置文件中设置属性值，为适配器的每个实例配置 XML 配置文件的名称、描述和位置，<Predix Machine runtime container location>/configuration/machine/com.ge.dspmicro.machineadapter.mqtt-[n].config。

   格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

   数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"]. 清晰起见，可使用反斜杠分隔。

属性	**描述**	**类型**	**默认值**	**必需**
`com.ge.dspmicro.machineadapter.mqtt.configFile`	MQTT 适配器 XML 配置文件的路径。 重要提示： 如果此属性保留为空，适配器被禁用。	字符串	"configuration/machine/com.ge.dspmicro.machineadapter.mqtt-0.xml"	是
`com.ge.dspmicro.machineadapter.mqtt.name`	适配器唯一的名称。	字符串	"Mqtt Machine Adapter"	是
`com.ge.dspmicro.machineadapter.mqtt.description`	适配器的描述。	字符串	支持 Mqtt 节点的基本读/写功能。支持一组 Mqtt 节点的订阅。	否
`com.ge.dspmicro.machineadapter.mqtt.broker`	代理主要负责接收并筛选所有消息，决	字符串		是

属性	描述	类型	默认值	必需
	定谁对这些消息感兴趣，然后将消息发送给所有订阅的客户端。（例如，tcp://hostname:port 或 ssl://hostname:port）。			
com.ge.dspmicro.machineadapter.mqtt.clientId	连接至代理的 MQTT 客户端 ID。MQTT 适配器的每个实例应使用不同的客户端 ID。	字符串		否
com.ge.dspmicro.machineadapter.mqtt.interval	检查连接的心跳间隔（单位：秒）。如果断开，会尝试再次连接。	整型	I"2"	否
com.ge.dspmicro.mqtt.userName	如果代理要求，用于身份验证的用户名。	字符串		否
com.ge.dspmicro.mqtt.password	如果代理要求，用于身份验证的密码。	字符串		否

MQTT River

MQTT River 在边缘设备和 Predix 云端之间提供发布/订阅通信方法。

边缘设备将数据传输至MQTT代理，代理接收并筛选所有消息，然后，将消息发送到在Predix云端或本地网络中的 MQTT 客户端订阅者。

你可以用 MQTT River 通过 MQTT 代理将一个客户端的数据或命令发送到另一个。

信息流通过机器网关从数据源流向 Hoover spillway 和处理器，然后到达 MQTT River，最

终到达下图所示的 MQTT 代理。 图 4 使用 Hoover Spillway 和处理器的 MQTT River

注：

使用 SSL 连接至代理时，你必须将 MQTT 代理服务器证书导入到客户端 TrustStore, machine_client_truststore.jks。

依赖项

消费 MQTT River 需要 Maven 依赖项和 OSGi 导入。

- 消费 MQTT River 需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>mqttriver
-send</artifactId> <version>{Predix Machine version}</version> </depe
ndency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import-Package: com.ge.dspmicro.mqttriver.send.api;version="[1.0,2)"
```

获取 MQTT River 服务

使用声明式服务注入 `IMqttRiverSend` 服务。
下例展示了如何使用声明式服务注入 MQTT River 发送服务。

```
import com.ge.dspmicro.mqttriver.send.api.IMqttRiverSend; ...@Reference pub
lic void setService(IMqttRiverSend service) { //set service here }
```

配置 MQTT River

你可以通过对 MQTT River 进行命名并指定 MQTT 代理 URI、MQTT 客户端 ID 和话题进行配置。

你也可以提供 MQTT 代理可能需要的身份验证信息。

下列情况可以确定是否使用某些配置属性：

- 没有在 `MqttRiverSendRequest` 中设置 `topic` 和 `Qos` 时，MQTT River 用这些默认配置属性发送数据。
- 如果 `MqttRiverSendRequest` 中的 `topic` 和 `Qos` 已经初始化，MQTT River 用它们发送数据。
- 当使用 `transfer(InputStream data, Map<String, String> properties)` 方法传输数据时，如果 `topic` 和 `Qos` 属性未通过 `MqttRiverSendRequest.PROPKEY_MQTT_TOPIC` 和 `MqttRiverSendRequest.PROPKEY_MQTT_QOS` 项分别配置，MQTT River 使用配置文件中的默认 `topic` 和 `Qos` 发送数据。

格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 导航至<Predix Machine runtime container location>/configuration/machine。
2. 打开 com.ge.dspmicro.mqttriver.send-[n].config 文件。
3. 设置以下属性的值：

属性	描述	类型	默认值	必需
`com.ge.dspmicro.mqttriver.send.river.uuid`	river 的 ID。启动捆绑束时自动生成并保存。	UUID	（系统生成）	是
`com.ge.dspmicro.mqttriver.send.river.name`	river 的唯一名称。用于链接 river 到其他服务，如	字符串	"Mqtt Sender Servic	是

属性	描述	类型	默认值	必需
	Spillway。		e"	
com.ge.dspmicro.mqttriver.send.broker	MQTT 代理 URI。例如，tcp://hostname:port 或 ssl://hostname.port	字符串		是
com.ge.dspmicro.mqttriver.send.clientId	用来连接代理的 MQTT 客户端的客户端 ID。	字符串		否
com.ge.dspmicro.mqttriver.send.topic	代理用来筛选每个已连接客户端的消息的 UTF-8 字符串。	字符串	"Default MQTT Sender Topic"	是
com.ge.dspmicro.mqttriver.send.qos	关于保证消息传递的消息发送者和接收者之间的协议。有效值是： 0 至多一次传递。接收方不发送响应，接收方接受消息的所有权。 1 至少一次传递。确保至少一次消息到达接收方。 2 确切的一次传	整型	l"2"	是

属性	描述	类型	默认值	必需
	递。消息不得丢失或复制。			
com.ge.dspmicro.mqttriver.send.interval	检查 MQTT 连接的心跳间隔（单位：秒）。如果断开，会尝试再次连接。	整型	I"2"	否
com.ge.dspmicro.mqttriver.send.userName	如果 MQTT 代理要求，用于身份验证的用户名。	字符串		否
com.ge.dspmicro.mqttriver.send.password	如果 MQTT 代理要求，用于身份验证的密码。	字符串		否

使用 MQTT River API

你可以查看 MQTT River API 来了解全面实现 MQTT River 服务。

1. 导航至<SDK installation location>/docs/apidocs.zip 并提取文件。
2. 查看 MQTT River API：

   <SDK installation location>/docs/apidocs/index.html/com.ge.dspmicro.mqttriver.send.api

MQTT River Ping

用 MQTT River Ping 服务将 PING 消息发送到 MQTT 代理，并检查 MQTT 代理的连接性来验证 MQTT River 路径。

MQTT Ping 服务实现 `IValidateRoute` 接口。在调用 `ping(IPingMessage)` 方法之前，应由 MQTT River Ping 服务创建一个 `IPingMessage` 实例，通过调用 `createPingMessage(Map<String, Object> params, IPongNotification pongNotification)` 方法实现。如果要指定 MQTT river 的名称，MQTT River 路径需要带有 `com.ge.dspmicro.river.api.IPingConstants.RIVER_NAME_KEY` 项的参数。xxxx 需要 `pong` 通知回调函数在路径上处理 PONG 消息。

依赖项

消费此服务需要 Maven 依赖项和 OSGI 导入。

- 消费 MQTT River Ping 服务需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>validateroute-api</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import Package: com.ge.dspmicro.validateroute.api;version="[1.0,2)"
```

获取 MQTT River Ping 服务

可以通过下列方式获取 MQTT River Ping 服务：

- 获取 MQTT River Ping 服务实例，然后调用 River 接口方法 `getvalidateRoute()`。
- 使用声明式服务，注入 MQTT River Ping 服务。

本示例显示了如何注入 MQTT River Ping 服务：

```
@Reference(type = '*') public void setMqttRiverPing(IValidateRoute validateRoute) { if (YourMqttRiverRouteName.equals(validateRoute.getRouteName()) { //set service here this.mqttRiverPing = validateRoute; } }
```

OPC-UA 适配器

OPC-UA 机器适配器是一款基于 OPC-UA 标准的数据采集组件。适配器的目的是发现并连接到 OPC-UA 服务器，从特定的节点读写，并订阅至特定的节点。

作为 OSGi 服务，OPC-UA 机器适配器在 Predix 机器运行时容器中~~的顶部~~运行，并支持 `IMachineAdapter` 接口。和其他适配器一样，它通过 `Machine Gateway` 服务进行消费。

注：

参见机器网关了解如何获取适配器。
OPC-UA 适配器可以在 `Machine Gateway` 的适配器返回列表中找到，使用 `AdapterType` 属性，该属性在 `MachineAdapterInfo` 类中。
下图显示了高级架构：

图 5OPC-UA 架构

在设置工业资产时，应明确配置 OPC-UA 适配器连接的第三方 OPC-UA 服务器以启用或禁用写入访问。本适配器具有写入功能。如果用例不需要写入访问，最好从服务器端禁用这一权限。

OPC-UA 适配器和 OPC-UA 服务器的通信模式

OPCUA 机器适配器订阅有两种与 OPCUA 服务器通信的模式。

默认模式为订阅。下表描述了这两种模式。

模式	**描述**
订阅	OPCUA 适配器寄存器监控 OPCUA 服务器的节点。根据数据变更筛选器以及在订阅或节点中配置的间隔时间，OPCUA 服务器会将数据变更推送至适配器。
请求	OPCUA 适配器根据订阅时间间隔定期从 OPCUA 服务器中请求数据，无论数据是否在服务器上变更。在这种模式下数据变更筛选器不适用。

配置 OPC-UA 适配器

你可以执行以下操作：

- 在<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.machineadapter.opcua-[n].xml 文件中配置 OPC-UA 适配器连接节点和订阅的实例。

  OPC-UA 适配器使用 OPC-UA 客户端订阅来监测数据。在数据变更时，OPC-UA 客户端订阅只通知 OPC-UA 适配器。XML 文件中规定的发布时间间隔是检查数据是否变更的时间间隔。如果没有数据变更，不会有数据推送到 Hoover 中。

- 通过在<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.machineadapter.opcua-[n].config 文件中设置属性值定制 OPC-UA 适配器的实现。

1. 在<Predix Machine runtime container location>/configuration/machine 中配置 OPC-UA 适配器的连接节点和订阅。在此文件中静态配置所有读写节点和订阅。

XML 支持两个标签来显示 OPC-UA 节点标识符。*StringId* 标签仅支持字符串标识符。例如 -

```
<DataNode> <Name>Node-1</Name> <StringId>2:temperature</StringId> <DataNode>
```

标签 *NodeId* 支持 4 个 OPCUA 节点标识符类型。使用这些类型的标准表示如下：

```
ns=<namespace>;<type>=<text>
```

*namespace ~~命名空间~~*是命名空间的索引，*type ~~类型~~*可能是 `'s' (string)`、 `'i' (numeric)`、 `'g' (GUID)` 或 `'b' (opaque)`。第二个'=' 符号之后的文本是节点 id 的文本表示，是一个字符串、整型数字、GUID （没有大括号）或 base ~~基本~~ 64 编码不透明数据。

StringId 标签会持续支持向后兼容性。如果在 *DataNode* 下面找到了两个标签，*NodeId* 标签优先。

2. 为了指定发送到订阅本节点的 spillway 数据包中包含哪些时间戳，在 `DataSubscriptionConfig` 下面的 `TimestampOrigin` 元素中设置以下其中一个区分大小写的值。

TimestampOrigin	**描述**
`Adapter`	~~将该时间戳用作~~使用机器适配器（运行 Predix 机器的计算机）的时间戳
`Server`	~~在 OPC-UA 服务器中~~使用 OPC-UA 服务器（运行 OPC-UA 服务器服务数据的计算机。）中的~~该~~时间戳。~~（运行 OPC-UA 服务器服务数据的计算机。）~~ 注： 如果请求 `Server` 时间戳，但是在接收的数据中找不到，则使用 `Source` 时间戳。
Source	~~在数据源中~~使用数据源（生成数据的设备）中~~该~~的时间戳。~~（生成数据的设备）~~ 注： 如果请求 `Source` 时间戳，但是在接收的数据中找不到，则使用 `Server` 时间戳。

TimestampOrigin	描述
	如果 Source 和 Server 时间戳均未找到，则使用适配器时间戳。

3. 如果要筛选触发数据变更通知的数据，可以在 XML 文件中配置 DataChangeFilter 用于订阅和/或节点设置：

注：

如果订阅中的节点都有特定的数据筛选器，会覆盖订阅数据的筛选器。如果订阅中的节点没有特定的数据筛选器，只使用订阅数据筛选器。

设置	描述	默认值
trigger	触发通知消息的变更类型。 有效值是： **Status** 数值的状态 **StatusValue** 数值或数值的状态。 **StatusValueTimestamp** 数据源时间戳，数值或数值的状态。	StatusValue
deadbandType	表示是否使用了 deadband ~~死区~~，规定了使用的 deadband ~~死区~~类型。 deadband ~~死区~~是不会触发数据变更通知的数值的允许变更范围。订阅变量时，可以将 deadband ~~死区~~用作筛选器，可以用来避免升级客户端时不必要的噪音信号。 有效值是： **None** 未规定 deadband ~~死区~~。 **Absolute** 基于绝对值变更的 deadband ~~死区~~。 **Percent** 基于数据范围百分比的	None

设置	描述	默认值
	deadband ~~死区~~。	
deadbandValue	规定了 deadbandTypes 的数值。有效值是： **Absolute** 包括触发要生成通知的数字数据值的绝对值变更。如果 abs （最后值-新值） > DeadbandValue，触发数值变更。 **Percent** EURange 的百分比。仅适用于有 EURange 属性的变量。如果变更值超过了配置值范围的百分比，触发数值。	

XML 配置文件范例如下：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <OPCUAMachine
AdapterConfig> <OPCUAClientConfig> <AppName>PredixMachine_OPCUAAdapte
r</AppName> <ServerUri>opc.tcp://localhost:4841</ServerUri> <AppUri>u
rn:localhost:UA:PredixMachine_OPCUAAdapter</AppUri> <ProductUri>urn:g
e.com:UA:PredixMachine_OPCUAAdapter</ProductUri> </OPCUAClientConfig>
 <DataNodeConfigs> <DataNode> <Name>Node 1</Name> <StringId>2:Demo.Si
mulationSpeed</StringId> </DataNode> <DataNode> <Name>Node 2</Name> <
StringId>2:Demo.Dynamic.Scalar.Int32</StringId> </DataNode> <DataNod
e> <Name>New_Node 1</Name> <StringId>2:Demo.Massfolder_Dynamic.Variab
le0009</StringId> </DataNode> <DataNode> <Name>Node 3</Name> <NodeId>
ns=2;s=Temperature</NodeId> </DataNode> <DataNode> <Name>Node 4</Nam
e> <NodeId>ns=2;i=4294967294</NodeId> </DataNode> <DataNode> <Name>No
de 5</Name> <NodeId>ns=2;g=22bc1f2f-e598-4aac-b7a9-bede30b7711f</Node
Id> </DataNode> <DataNode> <Name>Node 6</Name> <NodeId>ns=2;b=YWJjZA=
=</NodeId> </DataNode> </DataNodeConfigs> <DataSubscriptionConfigs> <
DataSubscriptionConfig> <Name>OPCUA_Subscription_1</Name> <TimestampO
rigin>Source</TimestampOrigin> <PublishingInterval>1000</PublishingIn
terval> <DataChangeFilter trigger="StatusValue" deadbandType="Absolut
e" deadbandValue="1.0"/> <DataNode> <Name>Node 1</Name> <DataChangeFi
lter trigger="StatusValue" deadbandType="Absolute" deadbandValue="2.0
"/> </DataNode> <DataNode> <Name>Node 2</Name> </DataNode> </DataSubs
criptionConfig> <DataSubscriptionConfig pattern="Pull"> <Name>OPCUA_S
ubscription_2</Name> <PublishingInterval>1500</PublishingInterval> <D
ataNode> <Name>New_Node 1</Name> </DataNode> </DataSubscriptionConfi
g> </DataSubscriptionConfigs> </OPCUAMachineAdapterConfig>
```

4. 找到并打开下列文件：<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.machineadapter.opcua-[n].config。
5. 设置以下属性的值：

属性	类型	描述	默认值
com.ge.dspmicro.machineadapter.opcua.configFile	字符串	配置文件的名称。	
com.ge.dspmicro.machineadapter.opcua.name	字符串	OPC-UA 机器适配器的标识符。	"OPC-UA Machine Adapter“ ~~OPC-UA 机器适配器~~
com.ge.dspmicro.machineadapter.opcua.description	字符串	支持 OPC-UA 节点的基本读/写功能。支持一组 OPC-UA 节点的订阅。	"支持 OPC-UA 节点的基本读/写功能。支持一组 OPC-UA 节点的订阅。
com.ge.dspmicro.machineadapter.opcua.security.mode	枚举	无 BASIC128RSA15_SIGN BASIC128RSA15_SIGN_ENCRYPT BASIC256_SIGN BASIC256_SIGN_ENCRYPT BASIC256SHA256_SIGN BASIC256SHA256_SIGN_ENCRYPT	"NONE"
com.ge.dspmicro.machineadapter.opcua.keystore.path	字符串	用于 TLS 包含 OPC-UA 密钥对的 keystore 的路径（相对于 Predix 机器根目录）	"security/opcua_kesytore.jks"
com.ge.dspmicro.machineadapter.opcua.keystore.type	字符串	Keystore 的类型。	"JKS"
com.ge.dspmicro.machineadapter.opcua.keystore.password	字符串	KeyStore 的密码。自动加密值。	"dspmicro"
com.ge.dspmicro.machineadapte	字	keystore 中用于 TLS	"dspmicro"

r.opcua.key.alias	字符串	的别名引用条目	
com.ge.dspmicro.machineadapter.opcua.key.password	字符串	别名密码。自动加密值。	"dspmicro"
com.ge.dspmicro.machineadapter.opcua.truststore.path	字符串	truststore 的路径。	"security/machinegateway_truststore.jks"
com.ge.dspmicro.machineadapter.opcua.truststore.type	字符串	truststore 的类型。	"JKS"
com.ge.dspmicro.machineadapter.opcua.truststore.password	字符串	truststore 的访问密码。	"dsmpicro"
com.ge.dspmicro.machineadapter.opcua.connection.check.interval	整型	确保连接有效的时间间隔（单位：毫秒）。	I"5"
com.ge.dspmicro.machineadapter.opcua.connection.invalidation.retries	整型	在作废质量为BAD的数据之前，检查服务器连接并重试的次数（-1 关闭作废）	I"-1"
com.ge.dspmicro.machineadapter.opcua.opcua.server.username	字符串	用于服务器验证的用户名	
com.ge.dspmicro.machineadapter.opcua.server.password	字符串	用户的密码。	

OPC-UA 服务器

OPC-UA 服务器允许启用 Predix 机器的应用程序通过 OPC-UA 协议公开数据，该协议是一个机器到机器的常见协议。

服务支持不同端口的多个实例，允许定义应用程序信息、端口类型和安全模式。

启用 Predix 机器的应用程序控制 OPC-UA 节点结构。

OPC-UA 客户端能够连接至作为工业设备的 Predix 机器，因为服务器能够通过机器网关提取连接的设备。

使用范例

OPC-UA 服务器支持下列用例示例：

- 启用 Predix 机器的应用程序能够在 OPC-UA 服务器中匹配机器网关设备和节点结构。Predix 机器作为外部 OPC-UA 客户端的网关连接至本服务器。
- OPC-UA River 将 OPC-UA 用作数据传输协议。

限制

OPC-UA 服务器仅从机器网关 API 接受 `PDataValue` 格式。通过机器网关寻址定义服务器地址空间。

依赖项

消费此服务需要 Maven 依赖项和 OSGi 导入。

- 消费 OPC-UA 服务器需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>opcua-server</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import-Package: com.ge.dspmicro.opcua.server.api;version="[1.0,2)"
```

获取 IDataManager 服务

使用声明式服务注入 `IDataManager` 服务。
下例展示了如何注入 `IDataManager` 服务。

```
import com.ge.dspmicro.opcua.server.api.IDataManager; ...@Reference public void setService(IDataManager service) { //set service here }
```

配置 OPC-UA 服务器

通过在 com.ge.dspmicro.opcua.server-[n].config 文件中设置属性值，

你可以配置 OPC-UA 服务器的实现。

格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 找到并打开<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.opcua.server-[n].config 文件。
2. 设置以下属性的值：

属性	描述	类型	默认值	必需
com.ge.dspmicro.opcua.server.name	OPC-UA 服务器名称	字符串	"DefaultOpcuaServer"	是
com.ge.dspmicro.opcua.server.uri	OPC-UA 服务器 URI	字符串	~~不适用~~ N/A	是
com.ge.dspmicro.opcua.server.producturi	OPC-UA 服务器产品 URI	字符串	N/A ~~不适用~~	是
com.ge.dspmicro.opcua.server.namespaceUri	OPC-UA 服务器名称空间 URI	字符串	"http://www.ge.com/OPCUA/PredixMachineAddressSpace	是
com.ge.dspmicro.opcua.server.discoveryServerUri	OPC-UA 发现服务器 URI	字符串		
com.ge.dspmicro.opcua.server.port.tcp	暴露~~展示~~ OPC-UA 服务器的 TCP 端口。 保持~~保持~~这个值为	整型	I"48010"	否

属性	描述	类型	默认值	必需
	空，会禁用 TCP。			
com.ge.dspmicro.opcua.server.port.http	暴露~~展示~~OPC-UA 服务器的 HTTP 端口。 保持这个值为空，禁用 HTTP。	整型	I"48080"	否
com.ge.dspmicro.opcua.server.port.https	暴露~~展示~~OPC-UA 服务器的 HTTPS 端口。	整型	I"48443"	
com.ge.dspmicro.opcua.server.security.mode	用管道符"|"分隔的安全模式列表。选择服务器使用的模式。 无 BASIC128RSA15_SIGN BASIC128RSA15_SIGN_ENCRYPT BASIC256_SIGN BASIC256_SIGN_ENCRYPT BASIC256SHA256_SIGN BASIC256SHA_SIGN_ENCRYPT	枚举	"NONE"	否。如果为空，默认选择。
com.ge.dspmicro.opcua.server.keystore.path	用于 TLS 的 keystore 的文件夹路径。	字符串	"security/opcuaserver_keystore.jks"	是
com.ge.dspmicro.opcua.server.keystore.type	keystore 的类型。	字符串，支持 JCE keystore	"JKS"	是

属性	描述	类型	默认值	必需
		类型。		
com.ge.dspmicro.opcua.server.keystore.password	keystore 的密码。	字符串	"dspmicro"	是
com.ge.dspmicro.opcua.server.keystore.key.alias	配置的 keystore 中使用的密钥的别名。	字符串	"dspmicro"	是
com.ge.dspmicro.opcua.server.keystore.key.password	密钥的密码。	字符串	"dspmicro"	是

OPC-UA River

使用 OPC-UA river，你可以将数据从边缘设备传输到 Predix 机器中定义的 OPC-UA 服务器。

你可以从各种不同的边缘设备收集数据，在 OPC-UA 服务器上存储数据进行聚合和处理。

下图展示了机器网关适配器的数据通过机器网关流向 OPC-UA river，最终到达 OPC-UA 服务器的过程。 图 6 从网关适配器到 OPC-UA 服务器的数据流

依赖项

使用 OPC-UA river 需要 Maven 依赖项和 OSGi 导入：

- 消费此服务需要下列 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>opcuariver-send</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import-Package: com.ge.dspmicro.opcuariver.send.api;version="[1.0,2)"
```

限制

机器网关适配器的 JSON 数据使用特定的格式。如果你有定制的处理器或定制的 `PDataValue.toBytes()` 实现，可将适配器中的数据转换为不同的格式，OPC-UA river 无法解析数据并将其存储到 OPC-UA 服务器。*参见数据转换*。

数据转换

机器网关适配器将以下 JSON 格式的数据发送到 OPC-UA River：

```
{ "timestamp":1440608242631, "category":"REAL", "address":"com.ge.dspmicro.
machineadapter.modbus://127.0.0.1:502/2/20", "name":"Node-2-1", "quality":"
NOT_SUPPORTED (20000000) ", "value":211, "datatype":"INTEGER" }
```

OPC-UA River 解析地址信息以在 OPC-UA 服务器上创建文件夹和数据节点。在 OPC-UA 服务器上的 JSON 数据到数据节点之间的转换如下：

- 协议（通常为适配器的服务 PID）转换为 OPC-UA 服务器上的对象下的文件夹。
- 主机名称和端口转换为协议文件夹下的设备。
- 将端口和最后一个元素（独占）之间的路径转换为设备下的文件夹列表。
- 整个地址用作节点的节点标识符。
- 节点名用作变量名。
- 数据值转换为 JSON 中规定的数据类型，设置为 OPC-UA 服务器上的值。

下图展示了在 OPC-UA 服务器上如何将数据转换成文件夹结构：图 7OPC-UA 服务器

获取 OPC-UA River 服务

使用声明式服务，注入 OPC-UA River 服务。
以下示例显示了如何注入服务。

```
import com.ge.dspmicro.opcuariver.send.api.IOpcuaRiverSend; ...@Reference p
ublic void setService(IOpcuaRiverSend service) { //set service here }
```

使用 OPC-UA River API

查看 OPC-UA River API 了解如何全面实现 OPC-UA River。

1. 导航至<SDK installation location>/docs/apidocs.zip，并提取所有文件

2. 找到并打开下列 API：<SDK installation location>/docs/apidocs/index.html/com.ge.dspmicro.opcuariver.send.api。

配置 OPC-UA River

通过规定 OPC-UA River 的名称和 ID 以及提供 OPC-UA 服务器实例的名称，你可以配置 OPC-UA River 的实现。

格式化.config 文件（不是.cfg 文件）内的属性值时，使用类型字符，后跟引用的值字符串表示。例如，一个布尔型属性=B"true"。小写类型字符表示基本数据。字符串的类型可以省略。下表列出了类型和对应的类型字符：

T = 字符串型	I = 整型	L = 长整型
F = 浮点型	D = 双精度型	X = 字节型
S = 短整型	C = 字符型	B = 布尔型

数组格式为属性=<类型>[”<值 1>”,”<值 2>”]。例如，整型数组属性=I["1", "2", "3"].清晰起见，可使用反斜杠分隔。

1. 找到并打开下列文件：<Predix Machine runtime container location>/configuration/machine /com.ge.dspmicro.opcuariver.send-[n].config。
2. 设置以下属性的值：

属性	描述	类型	默认值	必需
com.ge.dspmicro.opcuariver.send.river.uuid	系统为 OPC-UA River 生成的唯一标识符。	UUID	~~不适用~~ N/A	是
com.ge.dspmicro.opcuariver.send.river.name	OPC-UA River 的唯	字符串	"Opcua Sender Service"	是

属性	描述	类型	默认值	必需
	一名称。			
com.ge.dspmicro.opcuariver.send.opcua.server.name	OPC-UA 服务器实例的名称。	字符串	"DefaultOpcuaServer"	是

OPC-UA River Ping

用 OPC-UA River Ping 服务发送 PING 消息至 OPC-UA River，并检查 OPC-UA 服务器的连接性，验证 OPC-UA River 路径。

OPC-UA Ping 服务实现 `IValidateRoute` 接口。在调用 `ping(IPingMessage)` 方法之前，由 OPC-UA River Ping 服务创建一个 `IPingMessage` 实例，通过调用 `createPingMessage(Map<String, Object> params, IPongNotification pongNotification)` 方法实现。OPC-UA River 路径需要一个带有 `com.ge.dspmicro.river.api.IPingConstants.RIVER_NAME_KEY` 项的参数来规定 OPC-UA river 的名称。需要一个 `pongNotification` 回调函数在整个路径上处理 `PONG` 消息。

依赖项

消费此服务需要 Maven 依赖项和 OSGi 导入。

- 消费 OPC-UA River Ping 服务需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>validateroute-api</artifactId> <version>{Predix Machine version}</version> </dependency>
```

- 消费捆绑束需要下列 OSGi 导入：

```
Import Package: com.ge.dspmicro.validateroute.api;version="[1.0,2)"
```

获取 OPC-UA River Ping 服务

你可以通过下列方式获取 OPC-UA River Ping 服务：

- 获取 OPC-UA River 服务的实例并调用 `IRiverSend` 接口方法 `getValidateRoute()`。
- 使用声明式服务，注入 OPC-UA River Ping 服务。

  以下示例显示了如何注入 OPC-UA River Ping 服务：

```
@Reference(type = '*') public void setOpcuaRiverPing(IValidateRoute v
alidateRoute) { if (YourOpcuaRiverRouteName.equals(validateRoute.getR
outeName()) { //set service here this.opcuaRiverPing = validateRoute;
 } }
```

XML 配置监测和通知

XML 配置监测和通知服务监测 XML 配置文件确定是否被修改、添加或删除，然后通过 XML 配置文件通知 Predix 机器服务这些变更。

回调这些服务在注册的 XML 文件中对变更做出响应。

使用范例

Predix 机器 OPC-UA 机器网关适配器参考 OPC-UA 服务器连接信息的 XML 配置文件和 OPC-UA 服务器中节点和订阅的元数据。XML 配置变更时，OPC-UA 机器网关适配器实现 `IXmlListener` 接口来转播通知。OPC-UA 适配器能够再次对自身进行初始化来适应变更。

功能

XML 配置监测和通知进程包括下列步骤：

1. Predix 机器运行时容器中的 Apache `Fileinstall` 服务配置文件（`felix.fileinstall.filter=.*\\.(cfg|config|xml)`）允许 `Fileinstall` 服务通过扩展名.xml 查看这些变更。
2. `Fileinstall` 服务使用实现 `org.apache.felix.fileinstall.ArtifactListener` 接口的 XML 配置监测和通知服务确定<Predix Machine runtime container location>/configuration/machine 文件夹中的 XML 文件已经被修改。XML 配置监测和通知服务也可以实现

`org.apache.felix.fileinstall.ArtifactInstaller` 服务，以接收配置文件夹中所有 XML 的更新通知。

3. 需要 XML 配置文件变更通知的 Predix 机器服务实现 `com.ge.dspmicro.device.api.xmlmonitor.IXmlListener` 服务。
    a. Predix 机器服务启动时，声明式服务将监听器添加到 XML 配置监测和通知服务中。
    b. XML 配置监测和通知服务添加 Predix 机器服务时，查询新服务的 XML 文件列表来接收通知。
    c. 更新或删除文件时，调用相应的函数，让 Predix 机器服务有机会适应这些变更。

```
package com.ge.dspmicro.device.api.xmlmonitor; public interfac
e IXmlListener { void xmlFileAdded(String xmlFile); void xmlFil
eUpdated(String xmlFile); void xmlFileDeleted(String xmlFile);
 String[] getXmlFilesList(); }
```

假设

- XML 配置监测和通知服务依赖~~取决~~于注册回调函数的每项服务用以~~，~~提供要监测的文件名。
- 对于共享同一个 XML 配置文件的不同服务或可以使用的 XML 文件数目没有限制。XML 配置监测和通知服务为每个回调函数存储了一个文件列表。更新时，该列表提供回调函数中更新文件的名称。

限制

- XML 配置监测和通知服务需要修改 `FileInstall` 配置来监视 XML 文件。如果未配置，服务不会被调用~~任何更新~~。
- 如果服务提供的文件名在监测的配置文件夹中不存在~~修改的配置文件夹中不存在的文件名~~，则 `FileInstall` 服务不会提供更新。
- 如果服务提供的文件名不存在于硬盘上~~不存在的文件名~~，则 `FileInstall` 也不会提供更新。
- 只支持小写的.xml 扩展名。

依赖项

消费此服务需要 Maven 依赖项和 OSGi 导入。

- 消费此服务需要以下 Maven 依赖项：

```
<dependency> <groupId>com.ge.dspmicro</groupId> <artifactId>device-common</artifactId> <version>{Predix Machine Version}</version> </dependency>
```

- 消费捆绑束需要以下 OSGi 导入：

```
Import-Package: com.ge.dspmicro.device.api.xmlmonitor;version="[1.0,2)"
```

获取 XML 监测和通知服务

> 和使用声明式服务注入的大多数服务不同，在 XML 监测和通知服务中~~士~~不会获取并使用任何服务。
>
> 1. 识别~~包括并应该~~监测 XML 配置文件变更的服务。
> 2. 针对每项服务，提供 `com.ge.dspmicro.device.api.xmlmonitor.IXmlListener` 服务。
>
> 当这些服务被激活~~有效~~时，XML 监测和通知服务在这~~那~~些服务中添加引用，当这些服务无效~~无效~~时，~~删除~~引用被删除。