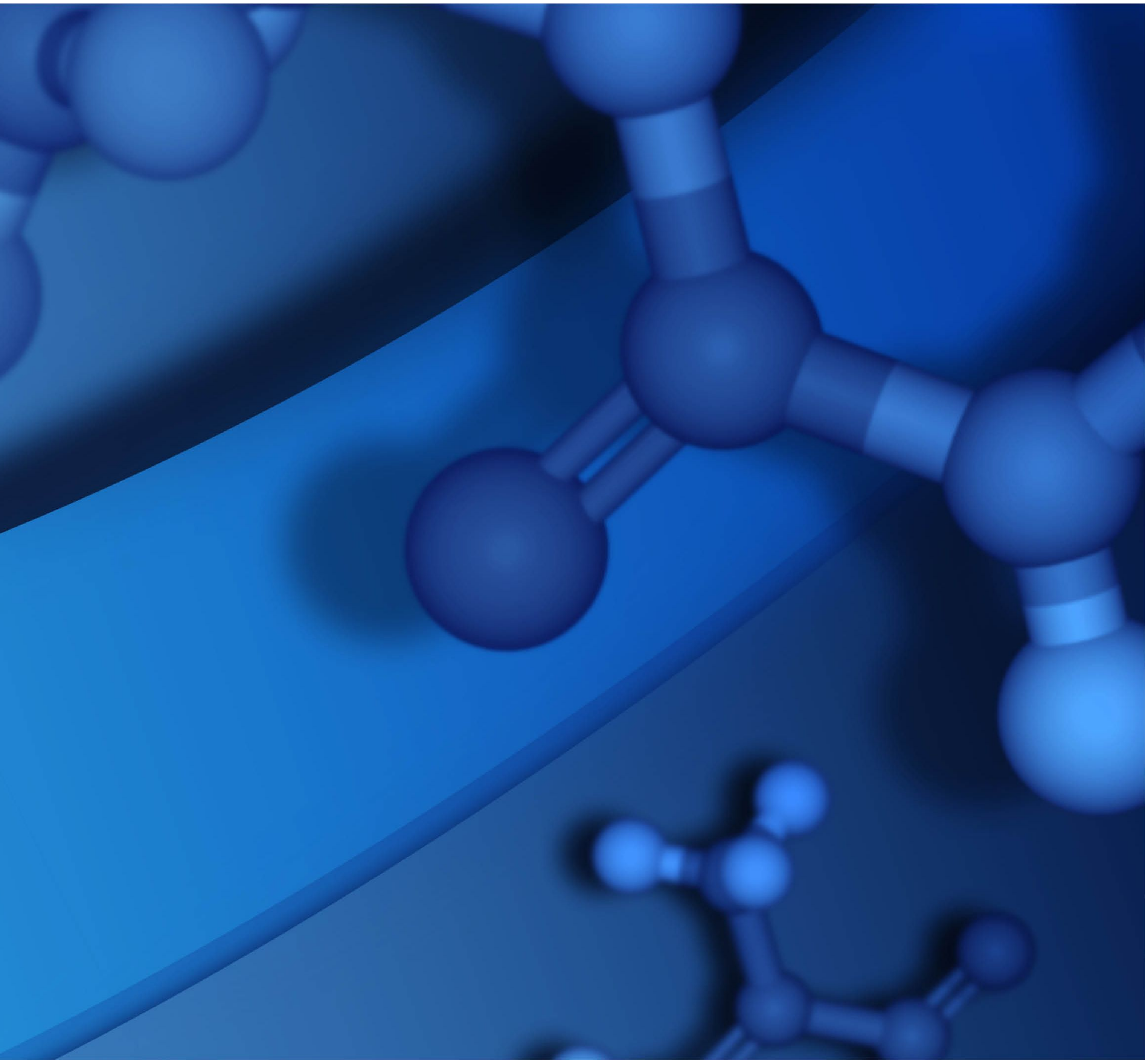
EQUIPMENT GUIDE

BIOVIA FOUNDATION HUB 2021 HF1



Copyright Notice

©2021 Dassault Systèmes. All rights reserved. 3DEXPERIENCE, the Compass icon and the 3DS logo, CATIA, SOLIDWORKS, ENOVIA, DELMIA, SIMULIA, GEOVIA, EXALEAD, 3DVIA, 3DSWYM, BIOVIA, NETVIBES, IFWE and 3DEXCITE, are commercial trademarks or registered trademarks of Dassault Systèmes, a French "société européenne" (Versailles Commercial Register # B 322 306 440), or its subsidiaries in the U.S. and/or other countries. All other trademarks are owned by their respective owners. Use of any Dassault Systèmes or its subsidiaries trademarks is subject to their express written approval.

Acknowledgments and References

To print photographs or files of computational results (figures and/or data) obtained by using Dassault Systèmes software, acknowledge the source in an appropriate format. For example:

> "Computational results were obtained by using Dassault Systèmes BIOVIA software programs. BIOVIA Foundation Hub was used to perform the calculations and to generate the graphical results."

Dassault Systèmes may grant permission to republish or reprint its copyrighted materials. Requests should be submitted to Dassault Systèmes Customer Support, either by visiting https://www.3ds.com/support/ and clicking **Call us** or **Submit a request**, or by writing to:

Dassault Systèmes Customer Support
10, Rue Marcel Dassault
78140 Vélizy-Villacoublay
FRANCE

Contents

Part 1:
Introduction

Chapter 1: Introduction to BIOVIA Equipment

BIOVIA Foundation is an integrated scientific environment to which BIOVIA applications (including Pipeline Pilot) are connected. It provides a unique infrastructure to enhance organizational efficiency, improve collaboration, and speed innovation in research-to-lab-to-plant processes.

BIOVIA Foundation includes the Pipeline Pilot protocol execution engine and implements security and user administration. It also includes BIOVIA Foundation Hub, which enables session management and single sign-on (SSO) for connected BIOVIA applications. This simplifies and reduces the cost of product integration, and also facilitates the integration of partner and third-party technologies.

You can specify resources in Foundation Hub, from which the connected applications can gather data. This guide explains how to configure and connect devices so that they can be used as a resource by connected applications.

IMPORTANT! Before starting, make sure you have set up locations for your organization. See the *BIOVIA Foundation Hub Administration Guide*.

This guide covers the following topics:

- **Managing Equipment:** Add and edit equipment classes, types, and devices.
- **Manage Equipment Status:** Manage life cycle state and status for devices.
- **Connecting and Registering Equipment:** Details for defining and setting up equipment based on connection type.
- **Managing Data Packets and Reference Ontologies:** Data Packets and Reference Ontologies map measurement data parsed from measurement files created by file-based equipment to Foundation Hub.
- **Managing the Equipment Crawler:** You can set up and use the equipment crawler to find, queue, and parse instrument-generated data files.

Intended Audience

This guide is intended for the equipment specialist or qualified administrator responsible for connecting and configuring equipment used in laboratories.

You must have the necessary administration rights in Foundation Hub to access the **Admin and Settings > Resources** menu.

Related Documentation

- *BIOVIA Foundation Hub Installation and Configuration Guide:* Detailed instructions for installing and configuring Foundation Hub. This guide is available with the Foundation Hub installation files.
- *BIOVIA Foundation Hub Administration Guide:* General administration for the Foundation Hub including security and BIOVIA application management. Click the **?** icon in the top toolbar of Foundation Hub and choose **Admin Guide**.
- *BIOVIA Foundation Hub API Reference Guide:* Reference for the Foundation Hub API. Click the **?** icon in the top toolbar of Foundation Hub and choose **API Reference Guide**.
- *BIOVIA Foundation Hub User Guide:* Guide for using the tools common to integrated BIOVIA applications that are available from the Foundation Hub home page such as notifications, tasks, and

equipment verification and calibration. Click the **?** icon in the top toolbar of Foundation Hub and choose **Help**.

- *BIOVIA Pipeline Pilot Help Center:* When configuring equipment such as file-based equipment, you will need to be familiar with Pipeline Pilot. The Pipeline Pilot Help Center is available from the Pipeline Pilot Server home page.

Getting Help

For assistance with configuring your equipment, contact Dassault Systèmes Customer Support.

Part 2:
Managing Equipment

Chapter 2: Equipment Key Concepts

All equipment used by BIOVIA applications must be registered in Foundation Hub. To register equipment, you set up the equipment definitions and then register and connect individual devices. You can then manage the maintenance, activity, and status of each device throughout its life cycle.

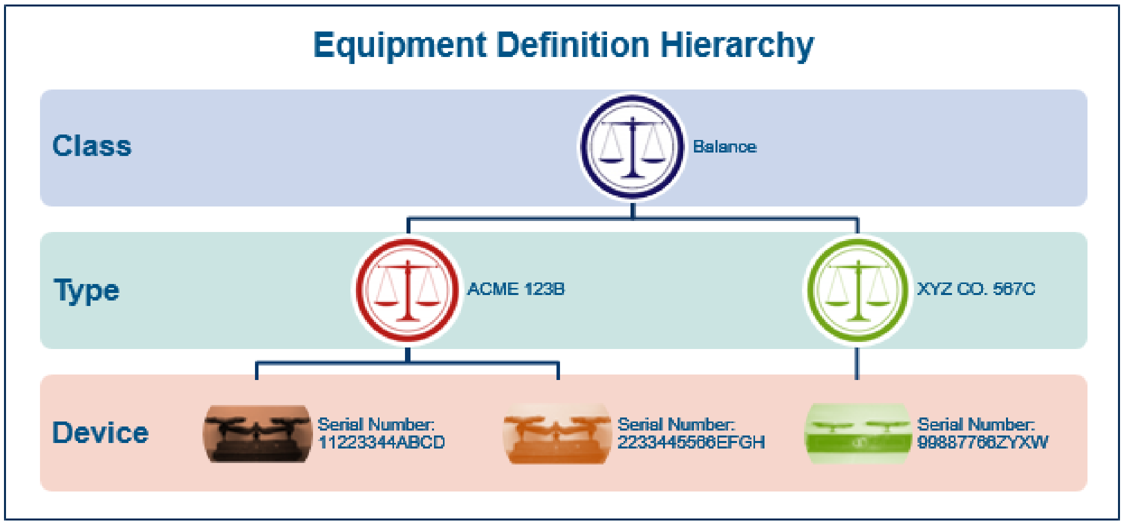
IMPORTANT! Before starting, make sure you have set up locations for your organization. See the *BIOVIA Foundation Hub Administration Guide*.

Equipment Definitions

Each device is defined by its equipment class and equipment type, and the metadata specific to that individual device.

- **Equipment Class (Category):** The equipment class defines the general group or category to which the equipment belongs (for example, balances, pH meters, and HPLCs). For more information, see Equipment Classes.
- **Equipment Type (Make/Model):** The equipment type defines the specification for a particular make and model of equipment. Where applicable, this definition can include the data that can be acquired from the equipment and the commands used to retrieve that data. For more information, see Equipment Types.
- **Device:** After the equipment classes and types have been defined, you can register devices based on them. The device also defines the unique information (for example, serial number or IP address).

  You can view a list of all registered devices in the Foundation Hub Equipment page. From **Admin and Settings**, open **Resources > Equipment**. For more information, see Devices.


Equipment Definition Hierarchy
Class
Balance
Type
ACME 123B
XYZ CO. 567C
Device
Serial Number: 11223344ABCD
Serial Number: 2233445566EFGH
Serial Number: 99887766ZYXW

Note: The Readings and Class Readings tables, which are available from the list of Related Items are not currently supported by the Parsing Framework, but will be in a future release.

Maintenance and Activity Management

You can track the maintenance, activity, life cycle state, and status for registered equipment, and make it available to other BIOVIA applications.

- **Equipment Events:** Foundation Hub allows you to manage scheduled and unscheduled maintenance events (metrology), including calibration and verification.
- **Equipment Logbook:** The equipment logbook tracks all the activity and usage history for an individual instrument or device.
- **Equipment Life Cycle and Status:** The equipment life cycle tracks equipment from set up through active life span, upgrades, and removal from active use. The equipment status is independent from its life cycle state and determines if the equipment is fit for use and if the measurements from it are valid.

Chapter 3: Equipment Classes

Equipment class is the highest level of the equipment hierarchy. They are used to categorize equipment types.

Viewing and Editing Equipment Classes

1. Open **Resources > Equipment > Related Items > Equipment Classes**.
2. Click a heading to sort by that property. Click a row to view the details.
3. The following buttons are available in the heading:
   - **Edit:** Edit the class properties.
   - **History:** View a history of changes to the class.

Adding a New Equipment Class

1. Open **Resources > Equipment > Related Items > Equipment Classes**.
2. Click the **Add Equipment Class** button.
3. Set the **Name**, **Category** (to organize the classes), and optionally, **Description**.
4. Click **Add** to choose equipment types to associate with the class. You can also choose the class for individual equipment types while editing them. See Adding a New Equipment Type.
5. Click **OK**.

Cloning an Existing Class

1. Open **Resources > Equipment > Related Items > Equipment Classes**.
2. Check the checkbox for the class you want to clone.
3. Click the **Clone Selected** button. The name will be appended with (#), and the other fields will be cloned from the original class.
4. Make changes as desired.
5. Click **OK**.

Deleting an Equipment Class

1. Open **Resources > Equipment > Related Items > Equipment Classes**.
2. Check the checkbox for the class you want to delete.
3. Click the **Delete Selected** button.

Chapter 4: Equipment Types

Equipment type defines the make and model of devices.

Viewing and Editing Equipment Types

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Review the list of equipment types. You can click a heading to sort the list.
3. Click a row to view the definition of the equipment type. The details are organized into the following categories:
   - **General:** Details about the equipment type, which vary based on type.
   - **Inventory:** List of devices defined for the type.
   - **Commands:** Varies based on the connection type:
     - **Direct-connected:** Configured commands and readings.
     - **Advanced direct-connected:** Advanced command configuration.
     - **File-based, database-connected, and adapters:** Pipeline Pilot protocol parameters.
4. Use the following buttons, as desired:
   - **Edit:** Edit the equipment type definition.
   - **History:** View a history of changes to the equipment type.
   - **Life Cycle:** Change the life cycle state. For example, you can **Activate** a draft equipment type, or **Upgrade** or **Inactivate** an active equipment type.

     Note: If the GxP level for an individual device is set to GMP, the associated equipment type must be active before the device can be used. See Editing the Equipment Life Cycle State.

Adding a New Equipment Type

Notes:
- You can also import predefined equipment types. See Importing an Equipment Type Library for more information.
- File-based equipment types require extensive configuration. See File-Based Equipment for more information.
- Make sure you have set up locations for your organization. See the *BIOVIA Foundation Hub Administration Guide* for more information.
- To add a label printer, see Configuring Label Printers.

To add a new equipment type:

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Click the **Add Equipment Type** button ⊕.
3. Set the **Name**, **Manufacturer**, **Model Number**, **Equipment Class**, and optionally, **Description**.

4. Set the **Connection Type**:
   - **Not Connected**
   - **Direct:** Connected through TCP/IP. To learn about setting up direct connected equipment, see Standard Direct Connected Equipment
   - **Advanced:** The equipment requires a more complex communication protocol to retrieve readings. To learn about setting up advanced direct connected equipment, see Advanced Direct Connected Equipment.
   - **File:** The equipment generates a file. To learn about setting up file-based equipment, see File-Based Equipment.
   - **Database:** The output of the equipment is stored in a database.
   - **Adapter:** Used to connect to an external system, such as an SDMS or CDS.
5. Click **OK**.
6. Open and **Edit** the equipment type to define additional details.

Tip: You can add new devices while creating the equipment type by clicking the **Add** button under the Equipment heading.

Extended Properties

The **Balance** equipment type has additional properties for **Precision** and **Category**. You can add extended properties to equipment type and other data objects. See the *BIOVIA Foundation Hub Administration Guide* for details.

Cloning an Existing Equipment Type

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Select the checkbox for the equipment type you want to clone.
3. Click the **Clone Selected** button . The name is appended with (#), and the other fields are cloned from the original equipment type.
4. Make changes as desired.
5. Click **OK**.

Deleting an Equipment Type

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Check the checkbox for the equipment type you want to delete.
3. Click the **Delete Selected** button .

Importing an Equipment Type Library

Foundation Hub provides a JSON-based library of equipment classes and types that you can import using a POST command. This helps when you are setting up equipment in Foundation Hub for your organization for the first time.

The default library of equipment classes and equipment types is located in the Foundation Hub installation directory:

```
<hub_install>\webapps\embedded\WEB-INF\classes\resources\data\IDS_Equipment_
Types_<release>.zip
```

Note: These files are read-only.

See the *BIOVIA Foundation Hub API Reference Guide* for more information. Click the **?** icon in the top toolbar of Foundation Hub and choose **API Reference Guide**.

Chapter 5: Devices

Viewing and Editing Devices

1. Open **Resources > Equipment**.
2. Review the list of the registered devices. You can click a heading to sort the list.
3. Click a row to view the details about the device. The details are organized into the following tabs:
   - **Metrology:** Shows records of equipment maintenance activity, such as preventative maintenance and calibration, along with metrology notes. You can execute a maintenance event for the equipment from this tab. See Managing Equipment Events.
   - **General:** Provides metadata about the equipment, which varies based on the equipment type. Depending on how the equipment is connected, the **Connection Details** section shows the following:
     - **Direct-connected:** Configured commands and readings. See Standard Direct Connected Equipment.
     - **Advanced direct-connected:** Advanced command configuration. See Advanced Direct Connected Equipment.
     - **File-based, database-connected, and adapters:** Pipeline Pilot protocol parameters. See File-Based Equipment.
   - **Components:** If the equipment consists of multiple components, lists the children components. See Creating Parent and Children Devices.
4. Use the following buttons, as desired:
   - **Edit:** Edit details for the device.
   - **History:** View a history of changes to the device.
   - **Logbook:** View the activity and usage history for the device. See Viewing the Equipment Logbook.
   - **Life Cycle:** Change the life cycle state. See Editing the Equipment Life Cycle State.

Adding a New Device

1. Before starting:
   - Check that the class and equipment type for the device have been defined. See Equipment Classes and Equipment Types.
   - File-based equipment types require extensive configuration. See File-Based Equipment.
   - To add a label printer, see Configuring Label Printers.
   - Make sure you have set up locations for your organization. See the *BIOVIA Foundation Hub Administration Guide* for details.
2. Open **Resources > Equipment**.
3. Click the **Add Equipment** button ⊕.

4. Set the equipment fields as required.

   Notes:
   - Do not include double quote characters in the name.
   - If you are planning to log equipment events for this device, the Location you choose must have the Time Zone set. See the *BIOVIA Foundation Hub Administration Guide* for details.
   - If the GxP Level for the device is set to non-GMP, only Non-GMP level equipment events can be logged.

5. Additional configuration is required based on connection type. See the following:
   - Standard Direct Connected Equipment
   - Advanced Direct Connected Equipment
   - File-Based Equipment
   - Chromatography Data System Equipment
6. Click **OK**.

Cloning an Existing Device

1. Open **Resources > Equipment**.
2. Select the checkbox for the equipment you want to clone.
3. Click the **Clone Selected** button . The name is appended with (#), and the other fields are cloned from the original device.
4. Make changes as desired.
5. Click **OK**.

Deleting a Device

1. Open **Resources > Equipment**.
2. Select the checkbox for the device you want to delete.
3. Click the **Delete Selected** button .

Creating Parent and Children Devices

Foundation Hub supports adding a device that consists of multiple devices. You can set up a parent device description and then add children to it.

1. Create the parent and children equipment. See Devices.

   Note: You cannot set the parent while adding a new child device. You must add the parent device first and then edit it to set the relationship.

2. Edit the equipment that you want as children to be associated with the parent:
   a. Open **Resources > Equipment**, and click on the device name of one of the child devices to view the details.
   b. Click **Edit**.
   c. Navigate to the **General tab**, set **Parent Equipment**, and then **Save**. Repeat for the remaining child devices.
3. After setting the parent for the children, you can check the parent for a list of all the children devices. Open **Resources > Equipment**, open the device, and then open the **Components tab**. Alternatively,

from **Resources > Equipment**, you can sort and filter by the **Parent** column to see all of the children for a parent device.

Chapter 6: Managing Equipment Events

Foundation Hub supports keeping records of equipment activity, such as preventative maintenance, calibration, cleaning, and verification. You can set up types of metrology events for different equipment types, and then execute them as you perform the tasks.

Adding an Equipment Event Type

You can set up metrology event types based on the type of equipment. When an event is executed for an individual instrument or device, the results are tracked in the logbook for that device.

IMPORTANT!

- If the **GxP Level** for the device is set to non-GMP, only Non-GMP level events can be logged.
- The **Location** for the device must have the **Time Zone** set to log events.
- If you add a Time Zone to the Site location, the children locations will inherit the Time Zone. This is the time zone of the location for the equipment and will be used for metrology expiration times.

1. Open **Resources > Equipment > Equipment Types**.
2. Click a row to open the equipment type details.
3. Click **Edit**.
4. From the General tab, click **Add** for Equipment Event Types.
5. Define the new equipment event type:
   - **Name:** Enter a name for the event type.
   - **Maintenance Event Type:** Choose Verification, Calibration, Cleaning, or Preventative Maintenance. A Preventative Maintenance event can be an ad-hoc event that can occur on demand, such as cleaning, and does not affect the Fit for use status of the equipment.
   - **Interval and Interval Unit:** Choose a number and a unit (Daily, Weekly, Monthly, or Yearly) for the interval. For example, 2 times Monthly.
   - **Method:** Choose the name of the Compose recipe you want to use for event execution, if applicable. The list shows Compose recipes with a category of Metrology. Leave this field blank if you do not want to use a Compose recipe.
   - **GxP Level:** Choose GMP or non-GMP. The event applies only to devices with the selected GxP level.

Deleting an Equipment Event Type

1. Open **Resources > Equipment > Equipment Types**.
2. Click the **Name** to open that equipment type.
3. Click **Edit**.
4. From the General tab, hover over the row and click the **Delete Item** button ⛔.

Executing Equipment Events

After you set up event types for scheduled maintenance, you can execute an event for an individual device. You can also add events to record unscheduled maintenance.

For events that do not use a Compose recipe for execution, you can upload a file attachment as you perform the maintenance task.

Executing a Scheduled Equipment Event

You can execute a metrology event for an individual device from the Admin and Settings area.

Tip: You can also create a task plan to manage and execute events that use a Compose recipe. For more information, see the *BIOVIA Foundation Hub User Guide*.

1. Open **Resources > Equipment**, and then open the device for which to execute an event.
2. Open the **Metrology** tab from the links on the left.
3. Find the event you want to execute, and do one of the following:
   - If the event uses a Compose recipe for execution, click **Run**. Then, follow the recipe workflow to collect the values for each parameter and enter the results.
   - If the event does not use a Compose recipe, click **Edit**. Then, set the **Last Performed** date and time, **Last Technician** name, **Event Outcome**. Add **Remarks** or upload a file attachment, if desired. Click **Save**.

Note: When you execute an event for an individual instrument or device, the results are tracked in the logbook for that device. See Viewing the Equipment Logbook.

Adding an Unscheduled Equipment Event

You can add a logbook entry to document an event that does not correspond to a scheduled equipment event.

1. Open **Resources > Equipment**, and then open the device for which to log an event.
2. Open the **Metrology** tab from the links on the left.
3. Click **Add**.
4. Set the **Last Performed** date and time, **Last Technician** name, and **Event Outcome**. Add **Remarks**, if desired.
5. Click **OK**.

Uploading File Attachments

When you are executing an event that does not use a Compose recipe, you can upload a file attachment. For example, if the maintenance was performed by a third-party vendor, you might want to attach proof of what was performed (IQ, OQ, preventative maintenance, repairs).

1. When you are editing the event, click **Upload** next to the **Attachment** field.
2. Navigate to the file on your local disk, and click **Open**.

   Notes:
   - The maximum file size is 50 MB.
   - If you selected the incorrect file, you can replace it by clicking **Re-Upload**.
   - You can upload only one file for each event. To attach multiple files, you can add all the desired files to a ZIP file and then upload the ZIP file.

3. Finish entering information and save the event.

   A link to the file is included in the logbook entry and on the Metrology tab for the equipment (for a scheduled event). You can download the file by clicking the link.

Chapter 7: Viewing the Equipment Logbook

The equipment logbook tracks all the activity and usage history for each device. You can use this comprehensive device history to diagnose equipment problems and for other lab management activities. For example, an auditor can review where the equipment was used and whether it was functional at the time of use.

Logbook Entries

The logbook tracks these types of activity:

- **Scheduled maintenance events**: Execution of any scheduled preventative maintenance, calibration, cleaning, and verification for the device. These events are defined based on the type of equipment. See Managing Equipment Events.
- **Unscheduled maintenance entries**: Execution of any unscheduled maintenance event that an equipment administrator logs manually. See Adding an Unscheduled Equipment Event.
- **Data acquisition events**: Use of the device to perform a measurement. Because a single data acquisition event can relate to multiple tasks and measurements, it can result in multiple logbook entries. The logbook contains at least one entry for each related task and, if a single task has samples mapped to more than one measurement, the logbook contains an entry for each measurement.
- **Commands**: Execution of a command on a connected device. The types of commands vary based on how the equipment is connected. See Connecting and Registering Equipment.
- **Life cycle changes**: Changes to the life cycle state of the device. See Editing the Equipment Life Cycle State.
- **Equipment metadata updates**: Edits to the equipment definitions, such as equipment status and metrology notes. See Devices.

Viewing the Logbook for a Device

1. Open **Resources > Equipment**, and then open the device for which you want to view the logbook.
2. Click **Logbook**.
3. Review the logbook entries for the device. You can filter and sort the entries, as desired.

The **Logbook** shows the following columns by default.

Column	Description
Event Time	The date and time when the system logged the event.
Event Type	A short description of the event. For example, *Measure* for a data acquisition, *Parse* for the execution of a file parsing command, or *Calibration* for a calibration maintenance event.
Outcome	The result of the event. For example, *Pass* or *Fail* for an equipment verification or *Record Updated* for an equipment metadata update.
Event Context	A link to access additional, relevant information about the event. For example, see the values captured and the limits for an equipment verification event. Depending on the type of event, the link might:

Column	Description
	■ Open the recipe execution or data acquisition task that resulted in the entry. **Note:** To open the link, you must have permission to view the source recipe or task. ■ Download a file attachment, if available, for a manually entered maintenance event.
Remarks	Additional information about the event. For example, the username and the reason code, if signatures are required for a change of life cycle state, or the user's comments for a maintenance event.
Category	The name of the equipment event type, for scheduled maintenance events.
User	The full name of the user who executed the event.
Username	The username of the user in Foundation Hub.
Measurement	A reference to the measurement item that contains the collected data, if applicable.

Chapter 8: Managing Equipment Life Cycle State and Status

The equipment life cycle state and equipment status are monitored by the equipment administrator or other qualified lab personnel who manage the equipment inventory in the system.

- Life cycle state indicates whether the equipment is being defined, active and available for readings, being upgraded and not available for readings, or inactive. The life cycle state controls whether other applications such as Compose and Capture can access readings and whether you can make changes to the device definition.
- Status can be used to communicate whether the device is fit for use and the measurements from it are valid. The equipment status is independent of its life cycle state and does *not* affect whether applications can retrieve readings from the device.

Editing the Equipment Life Cycle State

1. Open **Resources > Equipment**.
2. Choose the **Life Cycle** in the upper-right corner:
   - **Draft:** The device is being defined and is not available for readings. You can change the life cycle state from Draft to Activate.
   - **Activate:** The device is operational and available for readings (and available to other BIOVIA applications such as Capture). You cannot make changes to the equipment definition unless you move the life cycle state to Upgrading. You can change the life cycle state from Activate to Upgrading or Inactivate.

     Note: If the GxP level for the device is set to GMP, the associated equipment type must be active before you can set the life cycle state to Activate.
   - **Upgrading:** The device is not available for readings. You can make changes to the equipment definition while it is in the Upgrading life cycle state. You can change the life cycle state from Upgrading to Activate or Inactivate.
   - **Inactivate:** The device is not available for readings.

   Note: Changing the life cycle state requires a signature by default. You can edit the signature policies for the different life cycle states in **Admin and Settings > Life Cycle Policies > Default Life Cycle Policy**.

Setting Equipment Status

1. Open **Resources > Equipment**.
2. Click the device name and click **Edit**.
3. Set the **Status**:
   - Pending
   - Active
   - Inactive
   - Missing
   - Salvage

- Out of Verification
- Out of Calibration
- Maintenance Needed
- Cleaning Needed

4. Click **Save**.

Part 3:
Connecting and Registering Equipment

Chapter 9: Connection Types

The connection type of the equipment determines how the data is collected and stored in Foundation Hub.

- **Not Connected:** Equipment that is not connected to a network or database and produces readings that cannot be retrieved electronically. Data is collected and entered manually. See Devices.
- **Direct-Connect:** Equipment that sends a data string via TCP/IP. You can configure how the string is parsed to store data in Foundation Hub. See Standard Direct Connected Equipment.
- **Advanced:** Direct-connected equipment that requires more advanced communication settings than standard direct-connected equipment. This type of equipment typically requires special handshaking or multiple requests to and from the device through the use of custom command scripts. See Advanced Direct Connected Equipment.
- **File-based:** Equipment that generates data files. The files must be parsed by using a Pipeline Pilot parsing protocol. See File-Based Equipment. For label printers, see Configuring Label Printers.
- **Adapter:** Equipment for which BIOVIA provides an adapter to a third-party application, such as Waters Empower™ or Thermo Scientific™ Chromeleon™ chromatography data systems (CDS). See Chromatography Data System Equipment.

Chapter 10: Standard Direct Connected Equipment

The Direct-Connect connection type is used for equipment whose data is parsed from a string that is returned by an instrument directly connected to a TCP/IP network. A standard Direct-Connected device sends a single parsing command to the device and receives a single result string back. An advanced Direct-Connected device typically requires special handshaking or multiple requests sent to and received from the device through the use of custom command scripts. See Advanced Direct Connected Equipment.

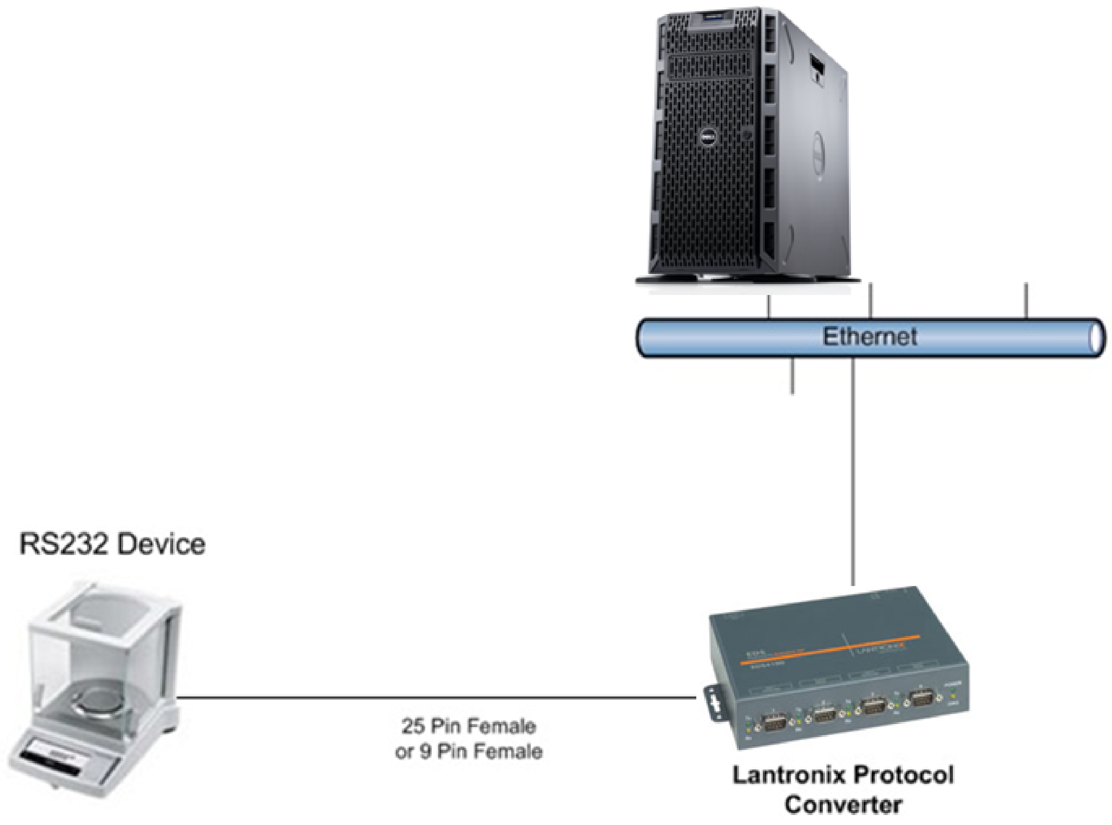
Prerequisites

- To configure a standard Direct-Connected instrument, you must have a sufficient understanding of how the instrument manages its flow of data. Contact Dassault Systèmes Customer Support for additional guidance.
- Foundation Hub 2021 HF1 must be installed and authentication must be configured. See the *BIOVIA Foundation Hub Installation and Configuration Guide*.
- Obtain and register a valid license for BIOVIA Equipment.

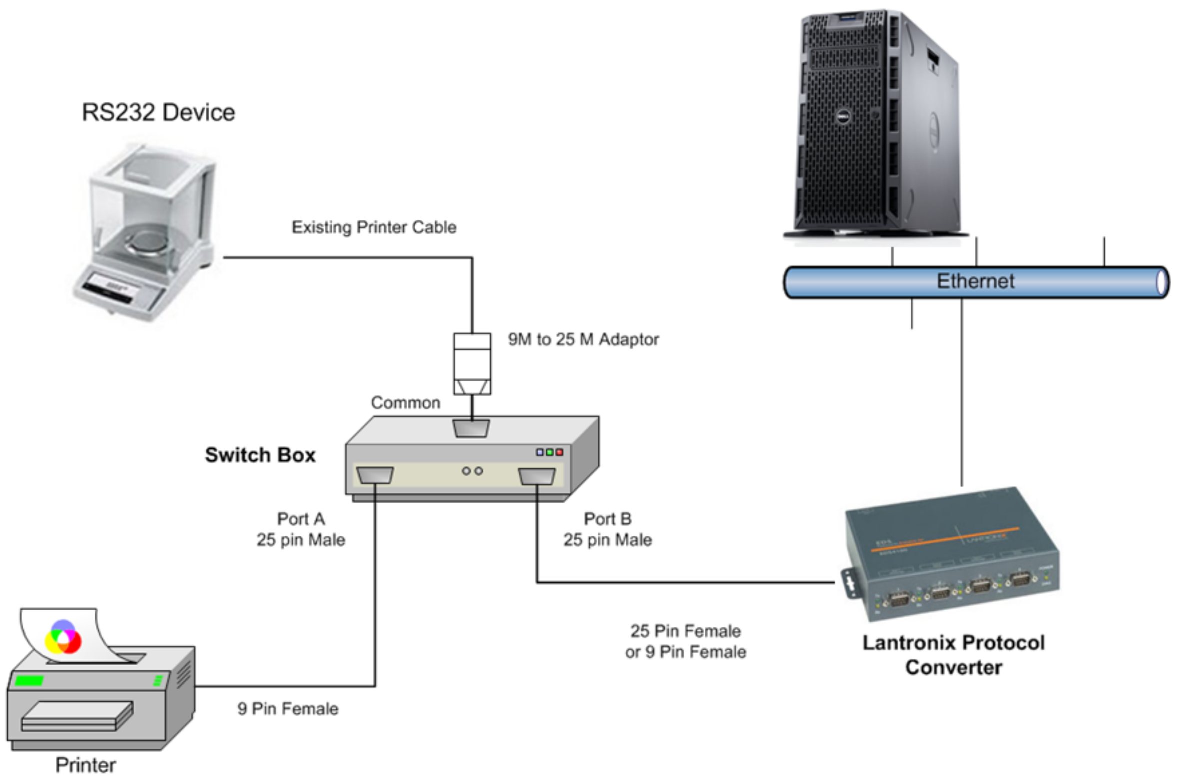
  Licenses are made available when the software is purchased. If you need assistance with licenses, contact Dassault Systèmes Customer Support.
- One top-level Location configured in Foundation Hub. If you plan to log equipment events, you must also configure your current time zone. See the *BIOVIA Foundation Hub Administration Guide*.
- Set up the Equipment Classes and Equipment Types in Foundation Hub that corresponds to the category and make/model of your equipment.
- The equipment must be connected to a TCP/IP network and accessible by Foundation Hub through a TCP/IP address and port.
- To interface the Direct-Connected equipment to the TCP/IP network, you must have a protocol converter installed on your network, and have the ability to connect to the instrument through a terminal emulator (such as PuTTY), issue commands to the instrument, and receive responses back from the instrument.

Network Architecture for Direct-Connected Equipment

- **Standalone Equipment:** This example shows an RS232 device connected to a protocol converter through a 9-pin or 25-pin serial or null modem cable. The protocol converter can access Foundation Hub over a standard TCP/IP Ethernet network connection.



- **Equipment and Printer:** This example shows the addition of a standalone printer. The device and serial printer are connected to a switch box using the appropriate adapters and 9-pin or 25-pin serial or null modem cables. The switch box is connected to the protocol converter with serial cabling. The protocol converter can access Foundation Hub over a standard TCP/IP Ethernet network connection.



Note: For information of the RS232 serial pin configurations, see Direct Connect Equipment Reference.

Configuring and Testing the Initial Communication for Direct-Connected Equipment

The equipment must be connected to a TCP/IP network and must be accessible by Foundation Hub through a TCP/IP address and port. It is important to set up and successfully test your initial communication between the device and Foundation Hub on onedevice before you proceed to set up additional devices. The recommended workflow to accomplish this is as follows:

1. Set up the initial communication between the device and your laptop. Use a serial or null modem cable to directly connect the two devices and configure the serial port properties.
2. Test the initial connectivity to the instrument using a terminal emulator (for example, HyperTerminal or PuTTY). The instrument can be used with BIOVIA Equipment if the terminal emulator can connect to the instrument, issue commands to the instrument, and receive responses from the instrument. See Using PuTTY to Test Connectivity.
3. Once the connectivity has been verified between the laptop and device, add your protocol converter. Connect the device to the protocol converter with a serial or null modem cable. Connect the laptop to the protocol converter with an Ethernet network cable.

4. Launch the protocol converter's management console:
   - For the serial connection between the device and protocol converter, configure the baud rate, data bits, parity, stop bits, and handshaking.
   - For the network connection between the laptop and protocol converter, configure the IP address, subnet mask, and default gateway.
5. Add the device in Foundation Hub, as described in Adding Direct-Connect Devices.
6. Test the reading. When the results are parsing correctly, set the instrument to "Active" to make it available for use in the system.
7. Configure and test additional instruments in the same manner.

Adding Direct-Connect Devices

Set the connection details for a Direct-Connected device as detailed here.

IMPORTANT! BIOVIA recommends that you register your standard Direct-Connected device only after you have successfully initiated the initial communication with it and parsed a data string correctly.

Adding a Direct-Connect Equipment Type

When you create a new equipment type, only basic information is captured. You must further configure the equipment type if you want to use it to register a device that you want to make active.

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Click the **Add Equipment Type** button ⊕.
3. Set the **Name**, **Manufacturer**, **Model Number**, **Equipment Class**, and optionally, **Description**. Do not include double quote characters in the name.
4. Set the **Connection Type** to **Direct**.
5. Click **OK**.
6. Open and **Edit** the equipment type you just created.
7. Click the **Commands** link on the left.
8. Complete the following fields:
   - **Name:** The purpose of the command (for example, Measure or Measure Temperature).
   - **Command String:** String sent to the instrument to solicit a reading.

     Note: If you change this after adding Readings, the readings could become invalid. See ASCII Character Replacement Tokens.
   - **Timeout (ms):** Amount of time to keep the port open for an instrument reading.
   - **Start on connection:** Start the timeout countdown on the establishment of a connection with the instrument. If false, then countdown starts when the first character is received from the device.
   - **Close on end of data:** The method that Foundation Hub uses to determine when to close the instrument's port. Select an option:
     - **No:** The port is kept open for the entire duration of the Timeout value.
     - **Any Reading:** The port is closed after any reading has been parsed.

    - **All Readings:** The port is closed after all readings have been parsed.

      IMPORTANT!
      Instrument data is read and buffered in segments of 64 bits. If all the readings are matched in a given buffered segment, the port is closed and no further data is retrieved. This means that, if the final reading uses a regular expression that is satisfied without the retrieval of further data, its content will be truncated. There are two ways of avoiding this:

      - Rewrite the final regular expression so that it is precise enough to force retrieval of the whole content.

        For example, `([^,]+)` at the end of a regular expression matches any sequence of non-comma characters up to the end of the segment. It is satisfied by the termination of the segment, and does not force retrieval of another segment. By contrast, `([^,]+),` forces retrieval of more data until a comma is encountered.

      - Add a "dummy" reading after the final actual reading, with a parse expression that will force retrieval of another segment of data.

    - **On Pattern:** The port is closed when a pattern is matched. Type the pattern in the **Pattern** box.

  - **Example Data:** Click Load to send the Command String to an instrument and return Example Data. You are prompted for a Host and Port. Hover over the Example Data to view red and green coloration, which indicate start and stop locations of the data to be parsed.

    Note: You can just type in the example reading data. Although it is not necessary, it is strongly recommended to connect to an actual instrument.

9. Click **New Reading** and complete the following fields:
  - **Name:** Name of the measurement (for example, Weight or pH).
  - **Units:** Choose from the list of standard units.
  - **Parse Method:** Choose one of the following methods to parse instrument data:
    - **Start-Stop with Key Token:** Parse a section defined by a Key Token and Start and Stop Strings.
    - **Start-Stop:** Parse a section defined by Start and Stop Strings. Highlight with Key Token: Parse a section of the string using an offset from the end of a defined Key Token string and a block length that defines how many characters to parse after an offset.
    - **Highlight with Key Token:** Parse a section of the string using an offset from the first character after the Key Token string and a block length that defines how many characters to parse after the offset.
    - **Highlight:** Parse a section of the string using an offset from the start of the string and a block length that defines how many characters to parse after the offset.
    - **Regular Expression:** Regular expression to use to locate the desired data.

      Note: The regular expression must be in the following form: ..(..).. The outside pieces are optional. For example, ([0-9]+) g and ([0-9]+}) would both return 32 for 32 g.

  - **Example Data:** Copy of the example data defined on the command.
  - **Key Token:** Enter the key string used to begin parsing. The parse begins after the Start string found after the Token Key when the Start-Stop with Key Token method is used, or after an Initial Offset value if the Highlight with Key Token method is used.

- **Start:** Enter a string that indicates the beginning of the parse. If you are using the Start-Stop with Key Token method, the parse begins after the Start string that appears after the Key Token string, which is useful if the Start string is found in multiple places throughout the Return String. This field is available for the Start-Stop parse methods.
- **Stop:** Enter a string that indicates the ending of the parse. This field is available for the Start-Stop parse methods.
- **Pattern:** The regular expression pattern to use when using that method to parse. See ASCII Character Replacement Tokens.
- **Example Output:** Value of the reading that would be returned by parsing the data based on the reading definition. For example if data is S 32.55 g and reading is Start/Stop based on S and g, the example output would be 32.55.

Note: To make results from a Direct-Connect equipment type available as measurements in Foundation Hub (for example, you can see measurements for it in the Admin and Settings > Measurements page), you must set up data packets for it. See Data Packets.

ASCII Character Replacement Tokens

Foundation Hub supports that use of replacement tokens for ASCII control characters in a command string (DirectCommand) for an Equipment Type. For example, if you want to include a carriage return as part of the command, you could use the <CR> token in the string. Foundation Hub will substitute the correct ASCII character before sending the command to the equipment.

The following methods are supported:

- **Decimal:** Use the format <ASC:#>, where # is 0-31. For example: <ASC:7> represents the "Bell" character.
- **Shorthand:** Use the format <symbol>, where symbol is one of the ASCII control character symbols. For example <CR> for carriage return or <TAB> for tab. Note that the symbol must be in upper-case.

Tip: See http://www.ascii-code.com/ for a list of the ASCII control characters that includes the decimal and the symbol.

Supported Shorthand Characters

```
<NUL>, <SOH>, <STX>, <ETX>, <EOT>, <ENQ>, <ACK>, <BEL>, <BS>, <TAB>, <LF>,
<VT>, <FF>, <CR>, <SO>, <SI>, <DLE>, <DC1>, <DC2>, <DC3>, <DC4>, <NAK>,
<SYN>, <ETB>, <CAN>, <EM>, <SUB>, <ESC>, <FS>, <GS>, <RS>, <US>
```

Adding a Direct-Connect Device and Testing

1. Add a device for the equipment type you created and configured. See Devices.
2. Open **Resources > Equipment**, click the row for the device you want to connect, and click **Edit**.
3. Click the **General** link on the left.
4. Set the **Connection Details**:
   - **Connected:** Data can be programmatically acquired from connected equipment. You must manually enter data for equipment that is not connected. You can register "not connected" equipment for traceability purposes when users manually enter measurements taken from that equipment.
   - **Host:** Host name or IP address of the Direct connected device.
   - **Port:** Port number of the Direct connected device.

5. Test the device:
   a. Choose the **Command** you configured for the equipment type.
   b. Click **Run**.
   c. Check that the Output is correct. If the reading failed or was incorrect, adjust the Connection Details and retest.
6. Click **Save**.

Chapter 11: Advanced Direct Connected Equipment

Advanced Connect equipment types require a more complex communication protocol to retrieve readings than basic instruments. In general Direct Connect instruments allow for a single command to be sent to and a single result to be retrieved from a device. This transaction pattern works for the vast majority of connected scientific equipment, and it is advisable to use this type of connection wherever possible.

Some devices require multiple requests to be sent to the device, and multiple responses from the device to be navigated by an information system (that is, Foundation Hub).

Advanced Connect functionality facilitates this by allowing the administrator to define a set of known commands and responses to send and receive. These are defined as part of an equipment type where the connection type is set to "Advanced". The actual definition of commands and responses is part of what's known as a Command Script. The Command Script, as well as some additional new parameters, comprise the new Advanced Command object. Advanced Command allows for greater flexibility in scenarios where a device needs special handshaking or multiple commands to retrieve a reading.

In Advanced Connect mode, Foundation Hub can be thought of as being reactive to what it receives from the device. The Hub still initiates the connection, as it does in Direct Connect, but once connected, the device itself can be seen as being the driver of the communication.

When Foundation Hub receives a string of text from the device that string is matched against a mapping of commands defined in the Command Script. If a matching command is found an associated response is returned from the Foundation Hub to the device. In general this process continues until timeout, reading requirements are satisfied, or the connection is closed.

Advanced Command

Advanced Connect equipment utilize an extension of Direct Command known as Advanced Command. While similar in terms content, the Advanced Command provides some additional configuration properties that facilitate the communication process.

Adding Advanced Direct-Connect Devices

Set the connection details for a Direct-Connected device as detailed here.

IMPORTANT! BIOVIA recommends that you register your advanced Direct-Connected device only after you have successfully initiated the initial communication with it (See Using PuTTY to Test Connectivity) and parsed a data string correctly.

Adding an Advanced Direct-Connect Equipment Type

When you create a new equipment type, only basic information is captured. You must further configure the equipment type if you want to use it to register a device that you want to make active.

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Click the **Add Equipment Type** button ⊕.
3. Set the **Name**, **Manufacturer**, **Model Number**, **Equipment Class**, and optionally, **Description**. Do not include double quote characters in the name.
4. Set the **Connection Type** to **Advanced**.
5. Click **OK**.

6. Open and **Edit** the equipment type you just created.
7. Click the **Commands** link on the left.
8. Complete the following fields:
   - **Name:** The purpose of the command (for example, Measure or Measure Temperature).
   - **Description:** Description of this command.
   - **Initial Command String:** Represents an optional command used to initiate a dialog with a device. Any remaining commands and responses are defined as part of the Command Script.
   - **Read Terminator:** The pattern on which each individual read from a device is terminated on.
   - **Enable Debug:** Foundation Hub will capture all data sent to or received from the device as part of the raw data returned. This also includes data stored in the match collection. This is useful when developing a Command Script, but should not be used in production.
   - **Acknowledgment:** Some devices require an acknowledgment string to be sent under certain circumstances. When the acknowledgment is sent can be configured globally at the Advanced Command level, or individually in the mappings in Command Script.
   - **Acknowledge Unknown Response:** Whether or not to automatically send the acknowledgment string for a device read that does not match a known command in the Command Script.
   - **Timeout (ms):** Represents the amount of time (in milliseconds) to keep a connection open after the last successful receipt of data from the device. Receipt of data does not mean a reading per se, but also includes commands or requests that the device may send to the Foundation Hub.

     This allows Foundation Hub to wait for additional requests from the device while keeping the connection open until the timeout is hit or the expected readings are retrieved from the device.

     Note: Timeout handling is different in Direct vs. Advanced Connect equipment types

   - **Close on end of data:** The method that Foundation Hub uses to determine when to close the instrument's port. Select an option:
     - **No:** The port is kept open for the entire duration of the Timeout value.
     - **Any Reading:** The port is closed after any reading has been parsed.
     - **All Readings:** The port is closed after all readings have been parsed.

       IMPORTANT!
       Instrument data is read and buffered in segments of 64 bits. If all the readings are matched in a given buffered segment, the port is closed and no further data is retrieved. This means that, if the final reading uses a regular expression that is satisfied without the retrieval of further data, its content will be truncated. There are two ways of avoiding this:
       - Rewrite the final regular expression so that it is precise enough to force retrieval of the whole content.

         For example, `([^,]+)` at the end of a regular expression matches any sequence of non-comma characters up to the end of the segment. It is satisfied by the termination of the segment, and does not force retrieval of another segment. By contrast, `([^,]+),` forces retrieval of more data until a comma is encountered.
       - Add a "dummy" reading after the final actual reading, with a parse expression that will force retrieval of another segment of data.
     - **On Pattern:** The port is closed when a pattern is matched. Type the pattern in the **Pattern** box.

> **Note:** The Timeout setting takes precedence over the **Close on end of data** setting.

9. **Command Script** represents a list of mappings comprised of commands, responses and control parameters. Click **New Entry** and complete the following fields:
   - **Read:** Define a string to be pattern matched against data received from the device. The string may contain regex metacharacters and groupings for capturing data. It is generally recommended to use the shortest string to successfully match/capture.
   - **Send:** Define a string to be sent to the device when device data matches the value in read. Send data supports token substitution of {MATCH} and {CHECKSUM} tags. If the send entry is omitted Foundation Hub will match the read string and do nothing.
   - **Acknowledge:** If present, and true, Foundation Hub will send the acknowledgment string defined on the Advanced Command. This happens before issuing the send value to the device.
   - **Capture:** If present, and true, Foundation Hub will include what was sent from the device as part of the raw result data against which readings are parsed. The capture option can be used on any entry.
10. For **Readings**, click **New Reading** and complete the following fields:
    - **Name:** Name of the measurement (for example, Weight or pH).
    - **Units:** Choose from the list of standard units.
    - **Parse Method:** Choose one of the following methods to parse instrument data:
      - **Start-Stop with Key Token:** Parse a section defined by a Key Token and Start and Stop Strings.
      - **Start-Stop:** Parse a section defined by Start and Stop Strings. Highlight with Key Token: Parse a section of the string using an offset from the end of a defined Key Token string and a block length that defines how many characters to parse after an offset.
      - **Highlight with Key Token:** Parse a section of the string using an offset from the first character after the Key Token string and a block length that defines how many characters to parse after the offset.
      - **Highlight:** Parse a section of the string using an offset from the start of the string and a block length that defines how many characters to parse after the offset.
      - **Regular Expression:** Regular expression to use to locate the desired data.

        > **Note:** The regular expression must be in the following form: ..(..).. The outside pieces are optional. For example, ([0-9]+) g and ([0-9]+}) would both return 32 for 32 g.
    - **Example Data:** Copy of the example data defined on the command.
    - **Key Token:** Enter the key string used to begin parsing. The parse begins after the Start string found after the Token Key when the Start-Stop with Key Token method is used, or after an Initial Offset value if the Highlight with Key Token method is used.
    - **Start:** Enter a string that indicates the beginning of the parse. If you are using the Start-Stop with Key Token method, the parse begins after the Start string that appears after the Key Token string, which is useful if the Start string is found in multiple places throughout the Return String. This field is available for the Start-Stop parse methods.
    - **Stop:** Enter a string that indicates the ending of the parse. This field is available for the Start-Stop parse methods.

- **Example Output:** Value of the reading that would be returned by parsing the data based on the reading definition. For example if data is S 32.55 g and reading is Start/Stop based on S and g, the example output would be 32.55.

11. Click **Save**.

Note: To make results from an Advanced Direct-Connect equipment type available as measurements in Foundation Hub (for example, you can see measurements for it in the Admin and Settings > Measurements page), you must set up data packets for it. See Data Packets.

Adding a Direct-Connect Device and Testing

1. Add a device for the equipment type you created and configured. See Devices.
2. Open **Resources > Equipment**, click the row for the device you want to connect, and click **Edit**.
3. Click the **General** link on the left.
4. Set the **Connection Details**:
   - **Connected:** Data can be programmatically acquired from connected equipment. You must manually enter data for equipment that is not connected. You can register "not connected" equipment for traceability purposes when users manually enter measurements taken from that equipment.
   - **Host:** Host name or IP address of the Direct connected device.
   - **Port:** Port number of the Direct connected device.
5. Test the device:
   a. Choose the **Command** you configured for the equipment type.
   b. Click **Run**.
   c. Check that the Output is correct. If the reading failed or was incorrect, adjust the Connection Details and retest.
6. Click **Save**.

Example

This example represents a command script developed to support the Siemens RAPIDLab 1200 Series blood gas analyzer. It consists of five command and response mappings.

Note: These are not in any particular order. In the case of the RAPIDLab device these read values could be sent from device at any time in the span of the connection, and in some cases (such as request for ID) unexpectedly.

~~~
```
{
   ...
  "commandScript": [
    {
      "acknowledge": true,
      "read": "SMP_NEW_AV<FS><RS>aMOD<GS>(.*?)<GS><GS><GS><FS>iIID<GS>
(.*?)<GS><GS><GS><FS>rSEQ<GS>(.*?)<GS>",
      "send": "<STX>SMP_REQ<FS><RS>aMOD<GS>{MATCH1}<GS><GS><GS><FS>iIID<GS>
{MATCH2}<GS><GS><GS><FS>rSEQ<GS>{MATCH3}<GS><GS><GS><FS><RS><ETX>
{CHECKSUM}<EOT>"
    },
    {
~~~

```
        "read": "<ACK>"
      },
      {
        "acknowledge": true,
        "read": "ID_REQ",
        "send": "<STX>ID_
DATA<FS><RS>aMOD<GS>LIS<GS><GS><GS><FS>iIID<GS>666<GS><GS><GS><FS><RS><ETX>
{CHECKSUM}<EOT>"
      },
      {
        "acknowledge": true,
        "read": "SYS_READY"
      },
      {
        "acknowledge": true,
        "capture": true,
        "read": "SMP_NEW_DATA"
      }
    ]
...
}
```

- In the first mapping Foundation Hub attempts to match the SMP_NEW_AV read from the device. In this example the Foundation Hub is using the regular expression pattern (.*?) to capture sample ID information from the SMP_NEW_AV that it needs to send as part of the response. Acknowledge has been set to true, so Foundation Hub will first acknowledge that it received the SMP_NEW_AV string. In this example the send string contains three {MATCH} tokens that represent (in order) the three parts of the SMP_NEW_AV string that were captured. These will be substituted with the captured values before the string is sent to the device. The send string also includes a {CHECKSUM} that will be calculated and inserted after the matches have been substituted and before the string is sent to the device.
- In the second mapping there is only the read value. In this case, Foundation Hub is receiving an acknowledgment from the device to which Foundation Hub should not respond.
- In the third mapping Foundation Hub is handling a request to identify itself to the device.
- In the fourth mapping Foundation Hub is receiving a SYS_READY notification. It must acknowledge that notification, but not send a follow-up response.
- Finally in the fifth mapping Foundation Hub is expecting SMP_NEW_DATA, which is known to contain the actual scientific reading data for which direct readings have been defined.

Matching and Capturing

In the previous section the concept of capturing parts of read strings from a device was introduced. For each command script mapping there is an implicit matches collection. That collection is populated by any regex capture groupings defined in the read string. For example:

```
"read": "SMP_NEW_AV<FS><RS>aMOD<GS>(.*?)<GS><GS><GS><FS>iIID<GS>
(.*?)<GS><GS><GS><FS>rSEQ<GS>(.*?)<GS>"
```

Assuming this read matches what is returned from the device the underlying match collection will contain three captured values. Foundation Hub can then use those values using token substitution in the send string:

```
"send": "<STX>SMP_REQ<FS><RS>aMOD<GS>{MATCH1}<GS><GS><GS><FS>iIID<GS>
{MATCH2}<GS><GS><GS><FS>rSEQ<GS>{MATCH3}<GS><GS><GS><FS><RS><ETX>
{CHECKSUM}<EOT>"
```

The match values are referred to via `{MATCH#}`, where # is an integer, starting at 1 and representing the order in which the capture grouping appeared in the read string.

Note: The match collection is local to an individual command string mapping. Captured data from a read can only be used in the corresponding send and not in another mapping's send. Therefore the match number resets in each mapping.

ASCII Character Escaping and Token Substitution

Similar to command strings in Direct Commands, Advanced Commands support the escaping of ASCII control characters using Foundation Hub's defined mapping of angle bracket (< >) replacements.

For Advanced Commands you should use the angle bracket representations of unprintable ASCII characters in the `Command String`, `Acknowledgement`, `Keep Alive String`, `Pattern`, `Read Terminator`, and `Command Script` mapping read and send fields.

As previously mentioned, parts of an Advanced Commands support a few special tokens that will be replaced prior to being sent to an instrument:

- `{MATCH#}` This token will be replaced with a corresponding entry from the matches collection of values captured from a "read" string. The # represents an positive integer starting at 1.
- `{CHECKSUM}` This token will be replaced by a checksum value. Currently Foundation Hub supports a checksum methodology in which a hexadecimal number is generated by performing a modulo-256 summation of all the previous characters in the frame, and then expressing the resulting 8-bit unsigned integer in hexadecimal format. Different checksum methodologies may be supported in future releases.

Advanced Equipment Type Definition Example

```
{
  "commandDetails": {
    "directCommands": [
      {
        "acknowledgeUnknownResponse": true,
        "acknowledgement": "<STX><ACK><ETX>{CHECKSUM}<EOT>",
        "autoSendAcknowledgement": false,
        "checksumType": "MOD256",
        "closeWhenReadingsComplete": "All Readings",
        "commandScript": [
          {
            "acknowledge": true,
            "read": "SMP_NEW_AV<FS><RS>aMOD<GS>(.*?)<GS><GS><GS><FS>iIID<GS>
(.*?)<GS><GS><GS><FS>rSEQ<GS>(.*?)<GS>",
            "send": "<STX>SMP_REQ<FS><RS>aMOD<GS>
{MATCH1}<GS><GS><GS><FS>iIID<GS>{MATCH2}<GS><GS><GS><FS>rSEQ<GS>
```

```
{MATCH3}<GS><GS><GS><FS><RS><ETX>{CHECKSUM}<EOT>"
            },
           {
             "read": "<ACK>"
           },
           {
             "acknowledge": true,
             "read": "ID_REQ",
             "send": "<STX>ID_
DATA<FS><RS>aMOD<GS>LIS<GS><GS><GS><FS>iIID<GS>666<GS><GS><GS><FS><RS><ETX>
{CHECKSUM}<EOT>"
            },
            {
              "acknowledge": true,
              "read": "SYS_READY"
             },
             {
               "acknowledge": true,
               "capture": true,
               "read": "SMP_NEW_DATA"
             }
           ],
           "commandString": null,"description": "Read new sample data from
device.",
           "directReadings": [
             {
               "exampleReading": "7.035",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "mpH",
               "offset": null,
               "parser": "StartStop",
               "startToken": "mpH<GS>",
               "stopToken": "<GS>",
               "unit": null
             },
             {
               "exampleReading": "119.8",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "mPCO2",
               "offset": null,
               "parser": "StartStop",
               "startToken": "mPCO2<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "488f136f-3089-4f8f-a9e3-b08104c7f85d",
                 "urn": "urn:unit:488f136f-3089-4f8f-a9e3-b08104c7f85d",
                 "displayName": "mmHg (millimeters of mercury)",
                 "externalId":
"http://qudt.org/vocab/unit#MillimeterOfMercury",
                 "name": "mmHg",
                 "symbol": "mm Hg",
                 "textName": "MMHG"
               }
             },
             {
               "exampleReading": "48.3",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "mPO2",
               "offset": null,
               "parser": "StartStop",
               "startToken": "mPO2<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "488f136f-3089-4f8f-a9e3-b08104c7f85d",
                 "urn": "urn:unit:488f136f-3089-4f8f-a9e3-b08104c7f85d",
                 "displayName": "mmHg (millimeters of mercury)",
                 "externalId":
 "http://qudt.org/vocab/unit#MillimeterOfMercury",
                 "name": "mmHg",
                 "symbol": "mm Hg",
                 "textName": "MMHG"
               }
             },
             {
               "exampleReading": "190",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "mGlucose",
               "offset": null,
               "parser": "StartStop",
               "startToken": "mGlucose<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "be097a0f-a6ec-43ca-b788-38cd3f3ff14b",
                 "urn": "urn:unit:be097a0f-a6ec-43ca-b788-38cd3f3ff14b",
                 "displayName": "mg/dL (milligrams per deciliter)",
                 "externalId": null,
                 "name": "mg/dL",
                 "symbol": "mg/dL",
                 "textName": "MGPERDL"
               }
             },
             {
               "exampleReading": "742",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "mBP",
               "offset": null,
               "parser": "StartStop",
               "startToken": "mBP<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "488f136f-3089-4f8f-a9e3-b08104c7f85d",
                 "urn": "urn:unit:488f136f-3089-4f8f-a9e3-b08104c7f85d",
                 "displayName": "mmHg (millimeters of mercury)",
                 "externalId":
 "http://qudt.org/vocab/unit#MillimeterOfMercury",
                 "name": "mmHg",
                 "symbol": "mm Hg",
                 "textName": "MMHG"
               }
             },
             {
               "exampleReading": "95.0",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "iFIO2",
               "offset": null,
               "parser": "StartStop",
               "startToken": "iFIO2<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "3440c210-7447-4930-8e3b-5777737f7828",
                 "urn": "urn:unit:3440c210-7447-4930-8e3b-5777737f7828"
                }
              },
              {
                "exampleReading": "31.3",
                "expression": null,
                "keyToken": null,
                "length": null,
                "name": "cHCO3act",
                "offset": null,
                "parser": "StartStop",
                "startToken": "cHCO3act<GS>",
                "stopToken": "<GS>",
                "unit": {
                  "id": "a8c8088e-2414-4fac-a3ac-46ec0629443f",
                  "urn": "urn:unit:a8c8088e-2414-4fac-a3ac-46ec0629443f",
                  "displayName": "mmol/L (millimoles per liter)",
                  "externalId": null,
                  "name": "mmol/L",
                  "symbol": "mmol/L",
                  "textName": "MMOLELITER"
                 }
               },
               {
                 "exampleReading": "-3.3",
                 "expression": null,
                 "keyToken": null,
               "length": null,
               "name": "cBE(vt)",
               "offset": null,
               "parser": "StartStop",
               "startToken": "cBE(vt)<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "a8c8088e-2414-4fac-a3ac-46ec0629443f",
                 "urn": "urn:unit:a8c8088e-2414-4fac-a3ac-46ec0629443f",
                 "displayName": "mmol/L (millimoles per liter)",
                 "externalId": null,
                 "name": "mmol/L",
                 "symbol": "mmol/L",
                 "textName": "MMOLELITER"
               }
             },
             {
               "exampleReading": "62.6",
               "expression": null,
               "keyToken": null,
               "length": null,
               "name": "cO2SAT",
               "offset": null,
               "parser": "StartStop",
               "startToken": "cO2SAT<GS>",
               "stopToken": "<GS>",
               "unit": {
                 "id": "3440c210-7447-4930-8e3b-5777737f7828",
                 "urn": "urn:unit:3440c210-7447-4930-8e3b-5777737f7828"
               }
             }
           ],
           "enableDebug": false,
           "exampleData": null,
           "keepAliveString": null,
           "keepAliveTimeout": null,
           "name": "READ",
           "pattern": null,
           "readTerminator": "<EOT>",
           "timeout": 10000
         }
       ]
     },
     "connectionType": "Advanced",
     "dataPacket": null,
     "description": "Siemens RAPIDLab 1200 Series Blood Gas Analyzer",
     "equipmentClass": {
       "id": "b82b5e70-6c97-41af-881b-783938fc89bd",
       "name": "Blood Gas Analyzer"
     },
     "externalId": null,
     "lastUpdated": "2017-03-29T18:32:21Z",
     "lifeCycle": {
         "id": "85217c67-9de8-4ce2-94ad-6e3a2eff40c1",
         "urn": "urn:lifecycle:85217c67-9de8-4ce2-94ad-6e3a2eff40c1",
         "description": null,
         "name": "Publish"
       },
     "lifeCycleRootId": "1ab32358-9058-4489-98ef-f597351bf5d5",
     "lifeCycleState": "draft",
     "lifeCycleVersion": "1",
     "manufacturer": "Siemens",
     "modelNumber": "1200",
     "name": "RAPIDLab 1200 Series"
  }
```

Chapter 12: File-Based Equipment

File-based equipment requires extensive setup to be integrated with Foundation Hub. In addition to defining the class and equipment type, you will need to set up the Foundation Hub **Parsing Framework** to parse reading and heading data from the equipment output files and map it to the Foundation Hub data model as described here. Foundation Hub supports the use of Pipeline Pilot subprotocols for parsing results files. Parser subprotocols are called by the *Parse* protocol that is supplied with the Parsing Framework. For more information, see The Parsing Framework.

For a full description of how to set up the Parsing Framework, see The File Crawler and the Parsing Framework.

File parsing is typically scheduled and performed automatically by the Parsing Framework. It is also performed when you upload a data file manually. For more information, see Uploading Equipment Data Files Manually.

Prerequisites

- Foundation Hub 2021 HF1 must be installed and authentication must be configured. See the *BIOVIA Foundation Hub Installation and Configuration Guide*.
- Obtain and register a valid license for BIOVIA Equipment.

  Licenses are made available when the software is purchased. If you need assistance with licenses, contact Dassault Systèmes Customer Support.
- You should have sufficient understanding of the format of the result file, and of how to create and use Pipeline Pilot protocols. Contact Dassault Systèmes Customer Support for additional guidance.
- You must have configured the following items in Foundation Hub:
  - Reference ontology vocabulary entries that map to data fields in files produced by the equipment.
  - A data packet that specifies data fields to be read from the equipment type.
  - An Equipment Class that corresponds to the category of the file-generating equipment.
  - An Equipment Type that corresponds to the make and model of your equipment.
  - At least one top-level **Location** appropriate for the equipment. If you plan to log equipment events, the Location must have the current time zone configured. See the *BIOVIA Foundation Hub Administration Guide* for details.
- There must be a parser subprotocol available in Pipeline Pilot that can correctly parse the format of data files generated by the equipment. If no suitable parser subprotocol exists, you must create your own. For more information, see Creating a Parser Subprotocol.

For more detailed guidelines on configuring the necessary items in Foundation Hub and Pipeline Pilot, see Configuring Foundation Hub for File Crawling.

Adding File-Based Equipment

To add file-based equipment, you will create a file-based equipment type and then add devices of that type.

Adding a File-based Equipment Type

1. Open **Resources > Equipment > Related Items > Equipment Types**.
2. Click the **Add Equipment Type** button ⊕.
3. Set the **Name**, **Manufacturer**, **Model Number**, **Equipment Class**, and optionally, **Description**. Do not include double quote characters in the name.
4. Set the **Connection Type** to `File`.

   Note: Devices that write data to a database, or that use an adapter, work in a similar way to file-based devices. For these data writing methods, set the **Connection Type** to `Database` or `Adapter` respectively.

5. Set the **Data Packet** to the data packet that was created for the equipment type. If no appropriate data packet yet exists, leave this field blank, and add it to the equipment type later. See Data Packets.
6. Click **OK**.
7. Open the equipment type you just created, and click the **Commands** link on the left.

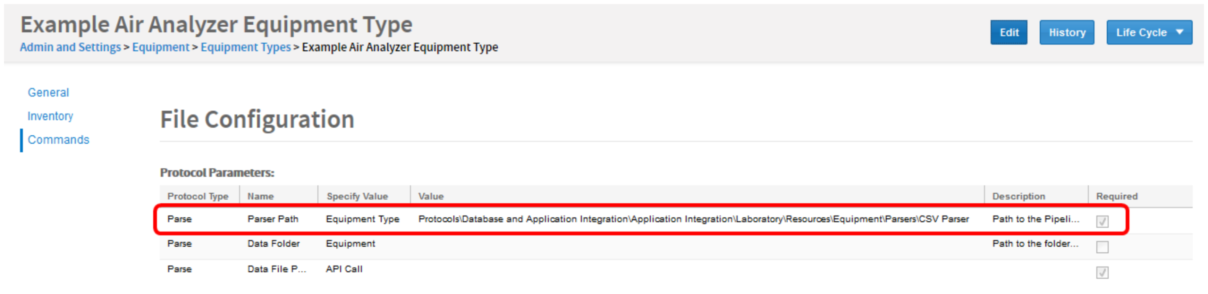
   If a dialog box opens with the message `The page contains unsaved changes`, click **Yes, discard changes**. (The message is irrelevant, since you have not made any changes.)

8. Click **Edit**.
9. Click **Add Default Parameters**.
10. A set of protocol parameters appropriate for file-based equipment is added to the equipment type.

   Edit the **Data Folder** and **Parser Protocol** parameters:

Protocol Type	Name	Specify Value	Value	Description	Required
Parse	Data Folder	Equipment	The path to the folder accessible by Pipeline Pilot that contains output files. This is normally configured in the Equipment itself, since each device will have its own data output folder. **Therefore, you should normally leave this field empty.**	Path to the folder accessible by Pipeline Pilot that contains output files and includes a file mask	No
Parse	Parser Protocol	Equipment Type	The path to the Pipeline Pilot parser subprotocol that can parse the data file. This is normally the same for all devices, and therefore is configured in the Equipment Type. An example parser path is shown below this table. **Note:** You will typically set up an equipment type before you create its parser subprotocol, so that you can test the protocol with data generated from real equipment. Therefore, you may not know the protocol path when you create the equipment type. If this is the case, you will need to add it to the equipment type later.	Path to the Pipeline Pilot Component that can parse the data file.	Yes

Example parser path:


Example Air Analyzer Equipment Type

Admin and Settings > Equipment > Equipment Types > Example Air Analyzer Equipment Type

File Configuration

Protocol Parameters:

Protocol Type	Name	Specify Value	Value	Description	Required
Parse	Parser Path	Equipment Type	Protocols\Database and Application Integration\Application Integration\Laboratory\Resources\Equipment\Parsers\CSV Parser	Path to the Pipeli...	✓
Parse	Data Folder	Equipment		Path to the folder...	☐
Parse	Data File P...	API Call			✓

Notes:
- After you create these parameters by clicking **Add Default Parameters**, the only field that you need to edit is **Value**. The other fields have appropriate default values.
- You do not need to edit the **Data File Path** parameter.

11. Click **Save**.

Adding a File-Based Device

1. Add a device for the equipment type you created and configured. See Devices.
2. Open **Resources > Equipment**, click the row for the device you want to connect, and then click **Edit**.
3. Click the **General** link on the left.
4. Set the **Location** of the equipment instance. Crawl and parse jobs will be run on the Pipeline Pilot server associated with the equipment's **Site**. If no Pipeline Pilot server is associated with the site, jobs will be run on the main Pipeline Pilot servers for the environment.

   If you want a different Pipeline Pilot server to run the crawl and parse jobs, set the **Processing Location** to specify the location of that Pipeline Pilot server. For details, see Setting the Processing Location.

   Note: Chromeleon and Empower add-ins do not require a Location. Jobs are sent to the Pipeline Pilot Data Center, or Processing Location if specified.

5. If you want files generated by the device to be discovered by the File Crawler, set **Data Discovery Method** to `Crawl`.
6. In the **Connection Details > Protocol Parameters** table, set the **Value** of the **Data Folder** parameter to the folder accessible by Pipeline Pilot that contains output files.

   Notes:
   - You can end the **Data Folder Path** with a wildcard and a file extension (for example, `G:\shared\AirAnalyzerFileDrops\*.csv`). This forces the Parsing Framework to ignore all files except for those with the specified extension.
   - If the protocol parameters are correctly configured in the equipment type, the only field that you need to edit is **Value**.
   - You do not need to edit the details of the **Data File Path** parameter if it is listed.
   - If there is no **Protocol Parameters** table, this means that the parameters are not correctly set for the equipment type. Go to the equipment's equipment type page, and configure the equipment type as described in Adding a File-based Equipment Type.

7. Click **Save**.

Setting the Processing Location

When you set up file-based equipment, you can use the **Processing Location** field to control which Pipeline Pilot server location to send crawl and parse jobs to. You can also designate a group of Pipeline Pilot servers as a data center to send jobs to.

Editing a Pipeline Pilot Location

1. Open **Settings > Applications > Installations**.
2. Click the row for the Pipeline Pilot server.
3. Click **Edit**.
4. Choose the **Location**.
5. Click **Save**.

Setting up a Data Center

You can designate a group of Pipeline Pilot servers as a data center and set the equipment to send crawl and parse jobs to that group of servers.

1. Open **Settings > Applications > Installations**.
2. Click the row for the Pipeline Pilot server.
3. Click **Edit**.
4. Set **Location** to **Data Center**.
5. Click **Save**.
6. Repeat for the remaining Pipeline Pilot servers in the data center.
7. Edit your devices and set **Processing Location** to **Data Center**.

Chapter 13: Chromatography Data System Equipment

BIOVIA provides add-ins (adapters) to Empower and Chromeleon Chromatography Data System (CDS) clients for exporting results to the Foundation Hub measurement store. The CDS client add-ins are installed separately. See the *BIOVIA Foundation Hub Installation and Configuration Guide* for details. Once the add-ins are installed and verified, you will configure equipment as discussed here.

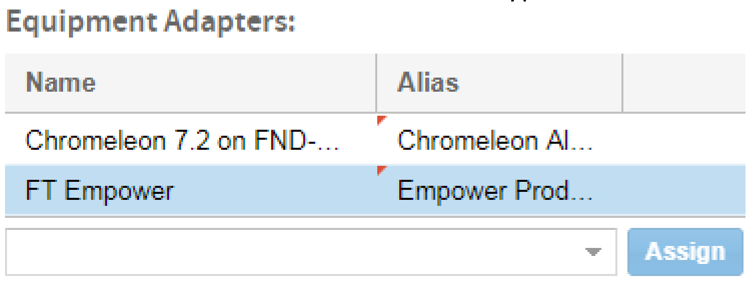
Adding an Adapter Equipment Type

1. Create a **CDS** equipment class:
   a. Open **Resources > Equipment > Related Items > Equipment Classes**.
   b. Click the **Add Equipment Class** button .
   c. Set **Name** to **CDS**.
2. Open **Resources > Equipment > Related Items > Equipment Types**.
3. Click the **Add Equipment Type** button .
4. Set the **Name**, **Manufacturer**, **Model Number**, and optionally, **Description**.
5. Set the **Connection Type** to **Adapter**.
6. Set **Equipment Class** to **CDS**.
7. Set **Data Packet** to the appropriate data packet for Empower or Chromeleon.
8. Click **OK**.
9. Open and **Edit** the equipment type you just created.
10. Click the **Commands** link on the left.
11. Click **Add Default Parameters**.
12. Click **Save**.

Adding an Adapter as a Device

1. Open **Resources > Equipment**.
2. Click the **Add Equipment** button .
3. Set **Equipment Type** to the adapter equipment type you just created. Set the remaining equipment fields as required.
4. Set the alias for the CDS. The name created when the add-in was installed will not match the actual name of the CDS. Associate the adpater name with the CDS name as follows:
   a. In **Equipment Adapters**, click the drop-down arrow and choose from the list of installed equipment adapters.
   b. Click **Assign**.

c. Click in the cell under the Alias column and type the name of the CDS:


Equipment Adapters:

Name	Alias	
Chromeleon 7.2 on FND-...	Chromeleon Al...	
FT Empower	Empower Prod...	

5. Click **Save**.

Chapter 14: Configuring Label Printers

This section details setting up a network Zebra label printer on Foundation Hub. Formatting of labels is performed by a label printer protocol in Pipeline Pilot. You can use the default label printer protocol that is supplied with Foundation Hub, or you can create your own. For more information, see Label Printer Protocols.

Create a Label Printer Equipment Class and Type

1. Open **Resources > Equipment**.
2. Click **Related Items > Equipment Classes**.
3. Click the **Add Equipment Class** button ⊕ and set the following fields:
   - **Name:** Label Printer
   - **Category:** Printer

   Note: The equipment class **Name** *must* be "Label Printer".
4. For **Equipment Types**, click **Add**.
5. Complete the Add Equipment Type information:
   - Set **Name** as desired. Do not include double quote characters in the name.
   - Set **Connection Type** to **File**.
   - Set **Equipment Class** to **Label Printer**.
   - Add an optional **Description**.
6. Click **OK** to save the equipment type and close the dialog.
7. Click **OK** to save the equipment class and close the dialog.

Add Control Protocol Parameters to the Label Printer Equipment Type

The Foundation Hub communicates with the label printer using a Pipeline Pilot protocol. In this step you will configure the parameters used by the protocol.

1. Open **Admin and Settings > Resources > Equipment**.
2. Click **Related Items > Equipment Types**.
3. Open the newly created label printer equipment type and click **Edit**.
4. Click the **Commands** link, then click **Add Parameter**.
5. Check the **Add another** checkbox and add the following parameters:
   - Add a **Number of Copies** parameter with the settings:
     - **Protocol Type:** Control
     - **Name:** Number of Copies
     - **Specify Value:** API Call
     - **Value:** 1 (this will be the default value)
     - **Required:** Yes

- Add a **Label Name** parameter with the settings:
    - **Protocol Type:** Control
    - **Name:** Label Name
    - **Specify Value:** API Call
    - **Value:** Name of the Pipeline Pilot protocol used to format the label. If you have more than one option, separate the names by pipes "|" and they will appear as options in the Capture user interface.

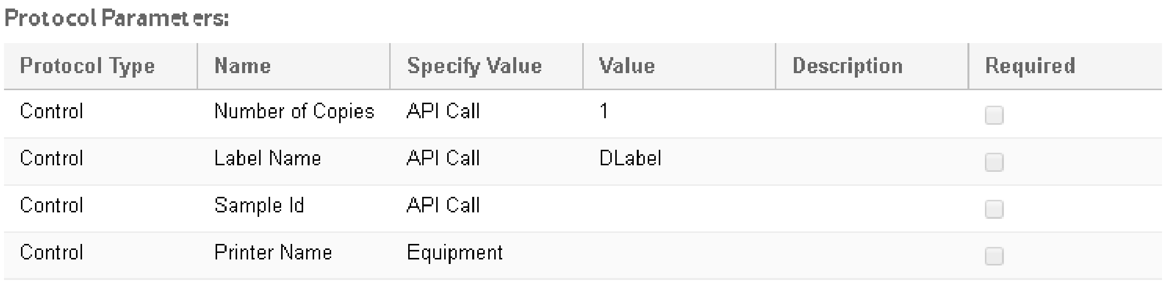
      An example protocol, **DLabel**, is available in Pipeline Pilot in **Protocols > Database and Application Integration > Application Integration > Laboratory > Resources > Equipment > Labels**.

      IMPORTANT! Any additional protocols for label printing must be stored in this folder.

      For guidelines on writing your own label printer formatting protocols, see Label Printer Protocols.
    - **Required:** Yes
- Add a **Sample Id** parameter:
    - **Protocol Type:** Control
    - **Name:** Sample Id
    - **Specify Value:** API Call
    - **Value:** none
    - **Required:** Yes

6. Depending on the label printer configuration, add either a **Printer Name** *or* **IP Address** parameter. Note that Foundation Hub will ignore IP Address if Printer Name is also defined.
    - **Printer Name** parameter: This method uses the Windows print manager/queue, which enables more jobs to be sent to the printer, and supports better error handling. For this method, install the printer as a device on the Pipeline Pilot server used for your Foundation deployment.
        - **Protocol Type:** Control
        - **Name:** Printer Name
        - **Specify Value:** Equipment
        - **Value:** none
        - **Required:** No
    - **IP Address** parameter: You can also set up to send print jobs directly via FTP. In this case, you only need to ensure that the printer is available to the Pipeline Pilot server.
        - **Protocol Type:** Control
        - **Name:** IP Address
        - **Specify Value:** Equipment
        - **Value:** none
        - **Required:** No
7. Click **OK**, then click **Save**.

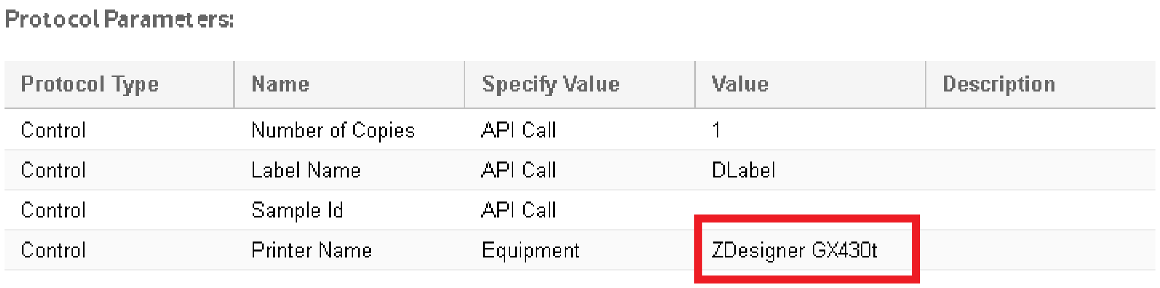
Example


Protocol Parameters:

Protocol Type	Name	Specify Value	Value	Description	Required
Control	Number of Copies	API Call	1		☐
Control	Label Name	API Call	DLabel		☐
Control	Sample Id	API Call			☐
Control	Printer Name	Equipment			☐

Add a Label Printer Device

1. Open **Admin and Settings > Resources > Equipment**.
2. Click the **Add Equipment** button ⊕.
3. Set **Equipment Type** to the label printer you have defined.
4. Set the **Barcode**, **Serial Number**, **Location**, and optionally, **Nickname**, **Primary Contact**, **Vendor**, and **Description**.
5. Set **Data Discovery Method** to **None**.
6. Click **OK**.
7. Open the newly created equipment type and click **Edit**.
8. In **Connection Details**, double-click **Printer Name** and enter the exact name of the printer (or **IP Address** depending if you configured the equipment type for that parameter):


Protocol Parameters:

Protocol Type	Name	Specify Value	Value	Description
Control	Number of Copies	API Call	1	
Control	Label Name	API Call	DLabel	
Control	Sample Id	API Call		
Control	Printer Name	Equipment	ZDesigner GX430t	

9. Click **Save**.
10. Click the **Life Cycle** button and select **Activate**.
11. Add the printer to the Pipeline Pilot server or test depending on the configuration you chose:
    - If you configured the printer for Printer Name, add the printer to the Pipeline Pilot server. See Add the Printer to the Pipeline Pilot Server.
    - If you configured a the printer for IP Address, test the printer. If the print job fails silently, see Troubleshooting a Label Printer for information about fixing the issue.

Add the Printer to the Pipeline Pilot Server

If you configured the printer for **Printer Name**, add the printer to the Pipeline Pilot. The following instructions are for Windows installations.

1. From the Pipeline Pilot server, open the **Add a printer** feature.
2. Type the printer **Hostname or IP address** and the **Port name**. Leave the option to **Query the printer and automatically select the driver to use** unselected.
3. For Additional port information required, set **Device Type** to **Standard** and choose **Generic Network Card**.
4. When prompted to Install the printer driver, choose the **Manufacturer** and **Printer**.
5. When prompted for the driver version option, choose **Use the driver that is currently installed**.
6. Type the printer name. The string must match the name you specified for the **Printer Name** parameter in step 8 in Add a Label Printer Device.
7. Test the printer:
    a. Open a task plan.
    b. Click **Print Label**.
    c. Choose the printer you configured.
    d. Click **Print**.

Label Printer Protocols

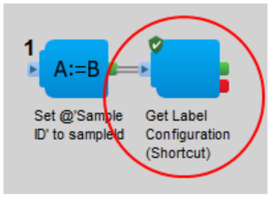
Foundation Hub uses a Pipeline Pilot protocol to generate Zebra Programming Language II (ZPL II) commands used for printing barcode labels. This section provides guidelines on writing your own label printing protocols.

The Control Protocol

The Control protocol runs when a user clicks **Print Label** in the Foundation Hub Samples widget or in the **Samples** tab of a Task Plan.

The Control protocol performs the following tasks:

- Receives information from Foundation Hub about the ID of the sample whose label is to be printed, the name of the Label Printing protocol to use, and the name or IP address of the printer to use.
- Calls the Label Printing protocol. This is done by the shortcut *Call Label Protocol > Get Label Configuration*:


1
A:=B
Set @'Sample ID' to sampleId
Get Label Configuration (Shortcut)

- Receives a string of ZPL printing commands back from the Label Printing protocol.
- Sends the ZPL commands to the label printer.

The Control protocol is configured in Foundation Hub in **Admin and Settings > Applications > Application Settings > Foundation Hub**. By default, this is set to the following protocol:

/Protocols/Database and Application Integration/Application Integration/Laboratory/Resources/Equipment/Control

This is the standard Control protocol that is supplied with the Foundation Hub component collection. Unless you have special requirements, you should always use this protocol.

Creating a Label Printer Protocol

This section provides guidelines on writing your own label printer protocols to format bar code labels.

Characteristics of a Label Printer Protocol

A label printer protocol requires input and output as described below.

Input

Name	Type	Description	Required?
Sample Id	StringType parameter	The ID of the sample whose label is being printed. This is passed by the Control protocol.	Yes

Output

Name	Type	Description	Required?
LabelData	Data record property	A string containing the ZPL II commands for printing the label.	Yes

Label Printing Components

The Pipeline Pilot core component collection includes components that you can incorporate into your label printer protocols to format labels. The label printer components are available from the Pipeline Pilot here: */Components/Data Access and Manipulation/Utilities/Prototypes/Label Printing*.

For details of the components' behavior and parameters, see the Label Printing component Help in the Pipeline Pilot client or the Pipeline Pilot Help Center.

Converters

Component	Behavior
Label to ZPL II	Creates a string of ZPL II processing instructions, and writes it to the property specified in the *Output to Property* parameter.

Elements

These components create objects that appear on labels. Each component takes input parameters that format the object, and creates a data record object. The label printer protocol should pass the data record objects to the *Label to ZPL II* component for conversion to ZPL II.

For details of the parameters taken by the components, see the component Help in the Pipeline Pilot client or the Pipeline Pilot Help Center.

Component	Behavior
1D Barcode (Label)	Creates a 1D bar code.
2D Barcode (Label)	Creates a 2D bar code.
Circle (Label)	Adds a circle or ellipse to a label.
Image from File (Label)	Reads an image file and adds it to a label.
Label	Aggregates all incoming label items into a single record for conversion to a set of printer commands.

Component	Behavior
Line (Label)	Adds a line to a label.
Rectangle (Label)	Adds a rectangle to a label.
Text (Label)	Adds text to a label.

Writers

Component	Behavior
Print Label	Sends label printing commands to a named label printer on the network.
Print Label (FTP)	Sends label printing commands to a label printer via FTP.

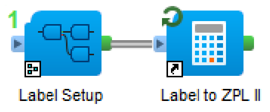
Label Printer Protocol: Example

An example label printer protocol is at the following location in Pipeline Pilot:

/Protocols/Database and Application Integration/Application Integration/Laboratory/Resources/Equipment/Labels/DLabel

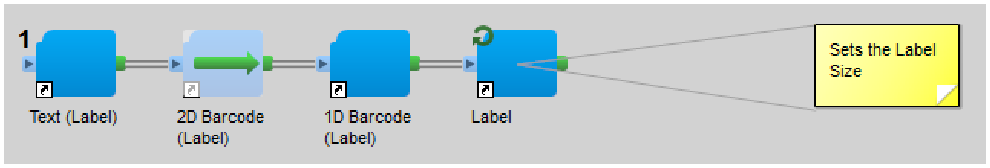
It is recommended that you use this protocol as the basis of your own label printer protocols.

The top level of this protocol has two components:



- *Label Setup:* A subprotocol that sets up formatting with parameters that are easy for a human user to understand. It uses Element components to do this — see Elements above.

  The example subprotocol formats a simple label that includes text and a 1D bar code:



  When you create your own protocol from the example, you can modify the subprotocol so that it creates a 2D bar code, by enabling *2D Barcode (Label)* and disabling *1D Barcode (Label)*. To add other objects to the label, insert appropriate Element components.

- *Label to ZPL II:* A copy of the *Label to ZPL II* component that converts the formatting specifications from *Label Setup* to ZLP II commands, and writes them to the *LabelData* output property parameter.

Part 4:
Managing Data Packets and Reference Ontologies

Chapter 15: Reference Ontologies

A **reference ontology** is a standard for a data format that is independent of equipment-specific data fields.

Reference ontologies are held in Foundation Hub vocabularies. For example, the vocabulary **BIOVIA Measurement Ontology** might contain entries for **Melting Point**, **Boiling Point**, **Length**, **Volume**, and so on. These represent standard measurement types that can be compared between data from different equipment types.

Individual fields in data packets are associated with reference ontologies. When fields in different data packets point to the same reference ontology, it can safely be assumed that they are based on the same scales, units, and so on. For more information, see Data Packets.

External IDs

Each entry in an ontology vocabulary must have an **External ID** defined. This is typically a URI that uniquely identifies the measurement type with a recognized standard — for example, `http://purl.alldatadefinitions.org/ontologies/property#XYZ_0000231`. Note that the URI does not need to point to a valid web address: the use of HTTP URIs is merely a convenience to ensure global uniqueness.

The **External ID** is used by the *Parse* protocol to identify the measurement type when it is called by the Equipment File Crawler. For more information about the File Crawler and the *Parse* protocol, see Using the Equipment File Crawler.

Creating a Reference Ontology

To create a reference ontology:

1. Open **Resources > Vocabularies**.
2. Click the **Add Vocabulary** button near the top of the grid. Give the vocabulary a **Name**, a **Category**, and if desired a **Description**.

   Note: It is recommended to include "Ontology" in the vocabulary name, so that it is easily identifiable in the list of vocabularies (for example, **BIOVIA Measurement Ontology**).

3. Create a vocabulary entry for each ontology measurement type — for example, **humidity**, **temperature**, **comment**, and so on:
   a. Under **Entries**, click **Add**.
   b. Specify a **Name** for the vocabulary entry. This is the text of the entry as it will appear in lists.
   c. If desired, provide a **Description** of the vocabulary entry.
   d. Click **OK**.

   Tip: You can select **Add another** to keep the Add Vocabulary Entry window open, which enables you to enter a another entry immediately. When you have finished, click **Cancel** to close the Add Vocabulary Entry window.

4. In the Add Vocabulary window, click **OK**.

   The ontology vocabulary is added to the **Vocabularies** list.

5. Add external IDs to the ontology entries:
   a. Click the new ontology vocabulary in the **Vocabularies** list.
   b. For each vocabulary entry under **Entries**:
      i. Click the entry.
      ii. Click **Edit**.
      iii. Type the external ID string in the **External ID** field.
      iv. Click **Save**.
      v. Press your browser's "back" button, to return to the page for the vocabulary, and then repeat Substeps i to iv for the next entry.

Note: Step 5 is required because you can only add an external ID to a vocabulary entry that already exists. You cannot specify an external ID when you create a vocabulary entry.

Chapter 16: Data Packets

A **data packet** specifies a set of **data fields** whose values are read from the output of a particular equipment type. The equipment type must output measurement data to a file or a database. For more information, see File-Based Equipment.

The data specified by a data packet is read and parsed by the *Parse* protocol that runs in Pipeline Pilot. The parsed data is stored in the Foundation Hub database.

Applications can retrieve parsed data through the Measurement REST API. The Foundation Hub Measurements page uses this API to retrieve and display measurement data. For more information, see Measurements.

There is usually a one-to-one relationship between an equipment type and a data packet. The name of each field in the data packet must be identical to the name of a field in the source equipment data. For example, if a field is called **Air Humidity** in the equipment data file, there must be a field in the data packet whose **Name** is `Air Humidity`.

The data packet workflow is as follows:

1. Create the data packet, including its data fields. See Creating a Data Packet.
2. If the equipment type associated with the data packet does not exist, create it. See Equipment Types.
3. Edit the equipment type and set the **Data Packet** field to the data packet that you created.

   Note: This field is not visible when you create a new equipment type.

4. Associate the equipment type with the appropriate parser subprotocol. See Equipment Types. If the parser subprotocol has not yet been created, you may need to leave this field blank until you know the subprotocol's path and name.

You can clone a data packet, and manage the data fields associated with data packets.

Creating a Data Packet

1. Open **Admin and Settings > Resources > Data Packets**.
2. Click the **Add** button ⊕.
3. Provide a **Name** and optional **Description**, then click **OK**.
4. Click the new data packet to open it, then click **Edit**.
5. Click the **Add** button under **Data Fields**.
6. Define **Data Fields**. For explanations of the settings for data fields, see Adding a Data Field.
7. In the **Data** box, provide any extra information that is required by the parser subprotocol.
8. Click **OK**. Add additional fields.
9. Click an existing field to open and click **Edit** to make changes.
10. Hover over a field and click the **Remove...** button ⊖ to delete it.

Cloning a Data Packet

1. From the **Admin and Settings** page, click **Data Packets**.
2. Select the check box for the data packet that you want to clone.
3. Click the **Clone Selected** icon .
4. In the **Confirm Clone** dialog box, click **Yes**.

   A clone of the data packet is created, and an editor for the new data packet opens.

   By default, the name of the new data packet is that of the source data packet, with a numeric suffix to make it unique. For example, if you clone the data packet pH Meter, the first cloned data packet is called pH Meter(2), the second pH Meter(3), and so on.

   When you clone a data packet, related data fields are cloned with it.
5. Edit the details of the cloned data packet.
6. Click **Save**.

Data Fields

You can manage data fields either from their parent data packets, or from the Data Fields page, which lists all data fields.

Adding a Data Field

1. Open **Admin and Settings > Resources > Data Packets > Related Items > Data Fields**.
2. Click the **Add** button .
3. Provide details of the data field:
   - **Name:** Name of the field class.
   - **Data Packet:** The data field's parent data packet.
   - **Is Sample Id:** Whether the field functions as the sample ID of data records. Parser subprotocols typically require one field (and one field only) to be the sample ID.
   - **Is Group By:** Whether the field is a grouping field. All rows with the same value in this field are grouped together in the Foundation Hub user interface, and can only have one sample assigned to them.
   - **Series:** Series of the data field class (for example, `Header` or `Table`). This gives a hint to the parser subprotocol about how to process the field. For example, the series `Header` can be used to indicate that the field should be output as header information that is not specific to individual readings.
   - **Type:** Data type of the field.
   - **Reference Ontology:** Standard definition for the data field's value.
   - **Unit:** Units for the data field.
   - **Required:** Whether the field is required.
   - **Data:** Any extra information that is required by the parser subprotocol.
4. Click **OK**. Add additional fields.
5. Click an existing field to open and click **Edit** to make changes.
6. Hover over a field and click the **Remove...** button to delete it.

Viewing and Editing a Data Field

1. From the **Admin and Settings** page, click **Groups**.
2. Open **Admin and Settings > Resources > Data Packets > Related Items > Data Fields**.
3. Click the data field.
4. To edit the data field:
   a. Click **Edit**.
   b. Edit the details of the data field. See Adding a Data Field for explanations of the details.
   c. Click **Save**.

Deleting a Data Field

1. Open **Admin and Settings > Resources > Data Packets > Related Items > Data Fields**.
2. Select the check box for the data field that you want to delete.
3. Click **Delete Selected** ⛔.
4. In the **Confirm Delete** dialog box, click **Yes**.

Cloning a Data Field

1. From the **Admin and Settings** page, click **Data Packets > Related Items > Data FieldsData Fields**.
2. Select the check box for the data field that you want to clone.
3. Click the **Clone Selected** icon ⧉ .
4. In the **Confirm Clone** dialog box, click **Yes**.

   A clone of the data field is created, and an editor for the new data field opens.

   By default, the name of the new data field is that of the source data field, with a numeric suffix to make it unique. For example, if you clone the data field Temperature, the first cloned data field is called Temperature(2), the second Temperature(3), and so on.
5. Edit the details of the cloned data field.
6. Click **Save**.

Part 5:
The File Crawler and the Parsing Framework

Chapter 17: Using the Equipment File Crawler

The Equipment **File Crawler** is a Pipeline Pilot protocol that roams the network searching for result files generated by instruments. Instruments typically deposit these files onto the file systems of computers that are connected to Pipeline Pilot Server through network shares.

The Parsing Framework

The File Crawler is part of the BIOVIA Foundation **Parsing Framework**. "Parsing Framework" is a general term for the Foundation Hub software components and Pipeline Pilot protocols that find, queue, and parse instrument-generated data files.

The Parsing Framework performs the following functions:

- **Data discovery.** The File Crawler monitors network shares for files that have been generated by instruments. If the File Crawler finds files, it adds them to the Parsing Framework's parsing queue.
- **File queuing.** Foundation Hub places data files in a queue as **work items**, waiting to be processed by the *Parse* protocol on Pipeline Pilot Server. When Foundation Hub finds a queued work item that is ready to be parsed, it initiates a job on Pipeline Pilot Server for the *Parse* protocol to process the data file.
- **Parsing.** The *Parse* protocol reads the data file, looks up the type of the equipment that generated the file, and forwards the file's content to the appropriate **parsing subprotocol**.

  Output files from parsing jobs are placed in the **Measurement Store**. Data in the Measurement Store is accessible to analysts through the Foundation Hub Measurements page . For details, see Measurements.

  If you are using equipment in Compose and Capture workflows, the Measurements page provides views of pre-parsed data during the execution of a recipe or test method in BIOVIA Capture.

Data File Names

Data file names should not contain characters that are invalid in a Windows file path or a URL.

Chapter 18: Configuring Foundation Hub for File Crawling

In order for the File Crawler to pick up equipment data files, resources must be appropriately configured in Foundation Hub.

For each instrument whose data will be processed by the Foundation Hub Parsing Framework, you must create the items in the table below. It is recommended that you create the items in the order listed.

Step	Item(s) to create	Details	Where to create the item(s)
1	Reference ontology (vocabularies)	Each field of data that a parser extracts must be associated with a **reference ontology** vocabulary entry. See Reference Ontologies.	**Foundation Hub Admin and Settings > Resources > Vocabularies**
2	Data packet and data fields	A **data packet** specifies a set of **data fields** to be read from a particular equipment type. See Data Packets and Data Fields.	**Foundation Hub Admin and Settings > Resources > Data Packets**
3	Equipment class	The **equipment class** is the highest level of the equipment hierarchy. Every equipment type must belong to an equipment class. See Equipment Classes.	**Foundation Hub Admin and Settings > Resources > Equipment > Equipment Classes**
4	Equipment type	Every device is an instance of an **equipment type**. An instrument used with the Parsing Framework must belong to an equipment type with a **Parser Path** parameter, specifying the parser subprotocol that you will create in Step 7. (If you have not decided on a name and path for the parser subprotocol, you will need to add it to the equipment type later.) For more about creating equipment types for the File Crawler, see Equipment Types.	**Foundation Hub Admin and Settings > Resources > Equipment > Equipment Types**

Step	Item(s) to create	Details	Where to create the item(s)
5	Location	Every instrument must have a physical **location**. See Locations.	**Foundation Hub Admin and Settings > Resources > Locations**
6	Equipment	This is the individual instrument that generates data to be parsed. For more about creating equipment for the File Crawler, see Equipment.	**Foundation Hub Admin and Settings > Resources > Equipment**
7	Parser subprotocol	This is the subprotocol that is called by the *Parse* protocol to convert equipment data into standard formats. For instructions on writing parser subprotocols, see Creating a Parser Subprotocol.	Pipeline Pilot client, logged in to the Pipeline Pilot Server connected to Foundation Hub

After you have the created the Foundation Hub resources, you must configure and enable the File Crawler. For instructions, see Administering the File Crawler.

Reference Ontologies

Each field of data that a parser extracts must be associated with a **reference ontology** vocabulary entry. This specifies a standard for a data format that is independent of equipment-specific data fields. For example, a data field **melt_pt** might be associated with a reference ontology entry **Melting Point**, which in turn is a member of a vocabulary **BIOVIA Measurement Ontology**.

The data fields in a data packet can be associated with different reference ontology vocabularies. For example, a data packet could have some fields linked to entries in a **BIOVIA Measurement Ontology**, and other fields linked to entries in a **Universal Measurement Ontology**.

For instructions on creating reference ontologies, see Reference Ontologies.

Data Packets and Data Fields

A **data packet** specifies a set of fields to be read from a particular equipment type.

One field in each data packet must be configured to be the sample ID of the data packet. The sample ID identifies each data reading. To make a field the sample ID, select **Is Sample Id** when you create the field.

Ensure that you specify an appropriate **Series** for each data field. This should be **Table** or **Header**:

- **Table** data is usually the data generated from individual readings. Some instruments output multiple readings to the same file, or append new readings to an existing file. In these cases, the file contains multiple table rows of data, with each row representing a reading.

  Table data is shown in the Table view of a measurement.

- **Header** data is not specific to any individual data reading, but is common to all readings, and therefore only occurs once in the file, typically above the table containing the reading data. Example Header fields are **Equipment Operator** or **Comments**.

  Header data is shown in the Header view of a measurement.

Equipment Classes

There are no special requirements for an equipment class that is used with the Parsing Framework. For instructions on creating equipment classes, see the chapter Equipment Classes.

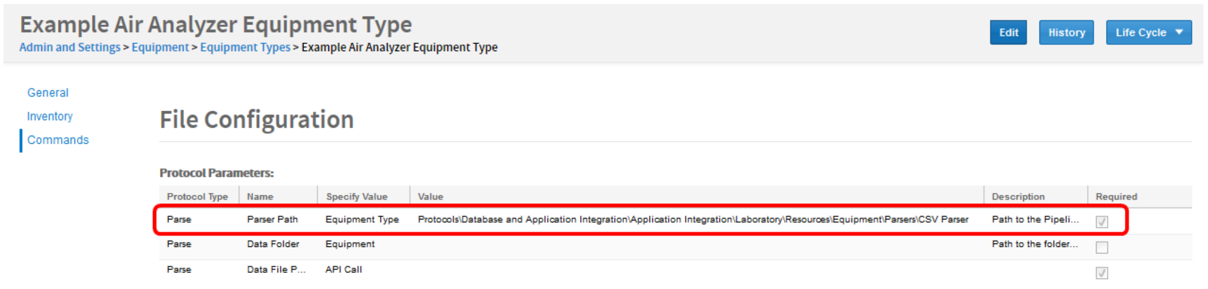
Equipment Types

To enable the File Crawler to pick up equipment data files, you must create at least one equipment type with the following configuration:

- Its **Connection Type** must be `File`, `Database`, or `Adapter`.
- In its **Commands** settings, the following protocol parameters must be configured:

Protocol Type	Name	Specify Value	Value	Description	Required
Parse	Data Folder	Equipment	The path to the folder accessible by Pipeline Pilot that contains output files. This is normally configured in the Equipment itself, since each device will have its own data output folder. **Therefore, you should normally leave this field empty.**	Path to the folder accessible by Pipeline Pilot that contains output files and includes a file mask	No
Parse	Parser Protocol	Equipment Type	The path to the Pipeline Pilot parser subprotocol that can parse the data file. This is normally the same for all devices, and therefore is configured in the Equipment Type. An example parser path is shown below this table. **Note:** You will typically set up an equipment type before you create its parser subprotocol, so that you can test the protocol with data generated from real equipment. Therefore, you may not know the protocol path when you create the equipment type. If this is the case, you will need to add it to the equipment type later.	Path to the Pipeline Pilot Component that can parse the data file.	Yes

Example parser path:


Example Air Analyzer Equipment Type
Admin and Settings > Equipment > Equipment Types > Example Air Analyzer Equipment Type
Edit
History
Life Cycle
General
Inventory
Commands
File Configuration
Protocol Parameters:

Protocol Type	Name	Specify Value	Value	Description	Required
Parse	Parser Path	Equipment Type	Protocols\Database and Application Integration\Application Integration\Laboratory\Resources\Equipment\Parsers\CSV Parser	Path to the Pipeli...	✓
Parse	Data Folder	Equipment		Path to the folder...	☐
Parse	Data File P...	API Call			✓

For complete instructions on configuring a file-based equipment type, see Equipment Types and Adding a File-based Equipment Type.

After you configure the equipment type, you must set up equipment that uses it. See Equipment.

Locations

Every device must be assigned to a location. No special configuration is required for equipment locations.

You create locations in **Foundation Hub Admin and Settings > Resources > Locations**. For instructions on creating locations, see the *BIOVIA Foundation Hub Administration Guide*.

Equipment

Each device whose data files are to be picked up by the File Crawler must be configured as follows:

- It must belong to an equipment type that is configured as described in Equipment Types.
- Its **Data Discovery Method** must be `Crawl`.
- Its **Data Folder Path** must be set to the folder where its data files are created. You set this in the **General** settings, under **Connection Details**, where there should be a **Protocol Parameters** table.

Tips:

- You can end the **Data Folder Path** with a wildcard and a file extension (for example, `G:\shared\AirAnalyzerFileDrops\*.csv`). This forces the Parsing Framework to ignore all files except for those with the specified extension.
- If there is no **Protocol Parameters** table, this means that the parameters are not correctly set for the Equipment Type. Go to the equipment's Equipment Type page, and configure the Equipment Type as described in Equipment Types.

For full instructions on configuring equipment, see Devices.

Chapter 19: Parser Subprotocols

The Parsing Framework runs the *Parse* protocol over files in the work items queue. The *Parse* protocol, in turn, calls a parser subprotocol for the type of the equipment that generated each file.

Note: If you develop a parser protocol that reads Allotrope files, it will only run on instances of Pipeline Pilot that have been installed on Windows Server 2012+ or Red Hat Linux 7+.

The Parse Protocol

The *Parse* protocol runs when there is a work item in the queue that needs to be parsed, and a parsing slot is available. The Parsing Framework passes the work item (which includes the instrument ID and the path of the file to parse) to the *Parse* protocol. The *Parse* protocol extracts measurement data from the file, calls the appropriate parser subprotocol to parse the data, and writes the parsed data to the Foundation Hub Measurement Store. Data from the Measurement Store is viewable in the Measurements page.

If you need to upload a data file that cannot be discovered by the File Crawler, you can upload it manually to the Data Preview page as described in Uploading Equipment Data Files Manually.

For development and testing purposes, you can also run the *Parse* protocol in the Pipeline Pilot client, as described below.

Testing the Parsing Framework

IMPORTANT! These instructions are NOT appropriate for inserting live data into the Measurement Store. If you need to upload live data that is not discoverable by the File Crawler, follow the instructions in Uploading Equipment Data Files Manually.

To test protocol parsing, or to learn how parsing is performed, you can run the *Parse* protocol manually in the Pipeline Pilot client:

1. Create appropriate Foundation Hub resources as described in Configuring Foundation Hub for File Crawling.
2. Create a parser subprotocol, as described in Creating a Parser Subprotocol.
3. Open the Pipeline Pilot client, and log in to the Pipeline Pilot Server that is connected to Foundation Hub.
4. Open the following protocol:

   ```
   Protocols\Database and Application Integration\Application
   Integration\Laboratory\Resources\Equipment\Parse
   ```
5. Click a blank area of the protocol's canvas.
6. For the value of the *Instrument ID* parameter, provide the ID of the instrument whose data you want to process. To find this:
   a. Log in to Foundation Hub as an administrator.
   b. Go to the **Admin and Settings > Equipment** page.
   c. In the list, click the entry for the instrument whose data you want to process.

      A page opens showing details of the equipment.

d. In the browser's address bar, copy the instrument ID from the URL. The instrument ID is a sequence of hexadecimal character blocks separated by dashes, for example `56341974-956b-4298-b9e6-82a46077fde9`.

7. For the value of the *Data File Path* parameter, provide the path of the data file created by the instrument. You can use Pipeline Pilot variables.

   Example:

   ```
   $(/pilot_data/scitegic/hub)/data/Equipment/Parsers/sampleParserFiles/
   ExampleAirAnalyzer.csv
   ```

   Note: You must specify a particular file. Unlike in the instrument's Connection Details page in Foundation Hub, you cannot use the * wildcard.

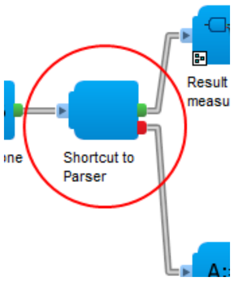
Creating a Parser Subprotocol

For each equipment type whose data is processed by the Parsing Framework, you must create a parser subprotocol.

This section provides guidelines on creating a parser subprotocol. Following it is a description of the *CSV Parser* that is supplied with the Foundation Hub Component Collection.

Where Parser Subprotocols are Called From

Parser subprotocols are called by the *Shortcut to Parser* component of the *Parse* protocol:



This component chooses a subprotocol based on the equipment type of the instrument that created the data file.

Characteristics of a Parser Subprotocol

A parser subprotocol must have a *Source Path* input parameter, and must generate appropriate output parameters and files.

Input

A parser subprotocol requires the following input parameter:

Name	Type	Description
Source Path	URLType parameter	Path of the file to be parsed. This parameter is passed by the *Parse* protocol.

Output

The Foundation Hub Parsing Framework requires the following properties and files to be output by a parser subprotocol:

Name	Type	Description
Manifest	Data record property	A list of the paths of files added to the job folder by the parser subprotocol. This can be an array, or a CSV string. The **Manifest** property *must* include the path of the `parsed_xy_data_0` output file. If the equipment generates header data, the Manifest property must also include the path of the `parsed_header_data` output file. **Note:** After the parser subprotocol has run, and has returned control to the *Parse* protocol, the *Parse* protocol appends a file called `equipment_metadata.json` to the **Manifest** property. This file, along with the parsed output files, is downloadable from the Measurement page for the parsing job. **Required.**
@Samples	Global property	A list of the IDs of the samples from the data file. This can be an array, or a CSV string. **Required.**

`parsed_header_data`	File	JSON string containing header fields parsed from the data file.
		Example:
		`{`
		`  "filename": "parsed_header_data.json",`
		`  "data": [`
		`    {`
		`      "number": 1,`
		`      "id": "Instrument Operator",`
		`      "name": "example instrument operator",`
		`      "prefLabel": "example instrument operator",`
		`      "value": "Janet Smith",`
		`      "dataType": "String"`
		`    },`
		`    {`
		`      "number": 2,`
		`      "id": "Comment",`
		`      "name": "example comment",`
		`      "prefLabel": "example comment",`
		`      "value": "Note sharp drop in humidity since yesterday due to wind veering south",`
		`      "dataType": "String"`
		`    }`
		`  ]`
		`}`

Name	Type	Description
		The file must be written to the folder of the Pipeline Pilot parsing job. For example, use a *Text Writer* component with the *Destination* parameter set to `$(JobDir)/parsed_header_data`. If the parser subprotocol does not create a `parsed_header_data` property, the measurement will include no header fields. **Required** if the equipment type's data packet specifies header fields.

`parsed_xy_data_0`	File	CSV-formatted data, containing table fields parsed from the data file.
		Example:
		`example instrument sample id,example date+time, example temperature,example humidity`
		`1,3/19/2017 13:30,23.64340634,46.73440332`
		`2,3/19/2017 14:30,23.28412678,46.77543443`
		`3,3/19/2017 15:30,23.21111095,46.77895568`
		The file must be written to the folder of the Pipeline Pilot parsing job. For example, use a *Text Writer* component with the *Destination* parameter set to `$(JobDir)/parsed_xy_data_0`.
		The file name can have an extension. For example, `parsed_xy_data_0.txt` and `parsed_xy_data_0.csv` are valid file names.
		There must not be more than one file whose root name (before the extension) is `parsed_xy_data_0`.
		Required.

Notes on Header Fields and Table Fields

- **Header fields** contain general information that is not specific to any individual data reading. They usually occupy their own section of the data file — for example, above the readings.

  Header data is shown in the Header view of Measurements pages. Note that not all equipment file formats include header data.

  By convention, header fields are designated in Foundation Hub by the value `Header` in their **Series** property. Note that the **Series** value of a field does not have an intrinsic meaning: it just gives a hint to the parser subprotocol about how the field should be processed. For example, a data packet might have fields whose **Series** is `Metadata`. A parser subprotocol could choose to include `Metadata` fields in its header output. It would therefore include these fields in the **parsed_header_data** property that it generates.

- **Table fields** contain the individual readings from the instrument. A data file might contain several data readings, or just one reading.

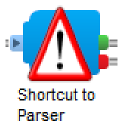
  Table data is shown in the Table view of Measurements pages.

  By convention, table fields are designated in Foundation Hub by the value `Table` in their **Series** property. As with header fields, the designation `Table` is simply a hint to the parser subprotocol, and fields from other series could be interpreted as table fields if appropriate. The subprotocol would include such fields in the comma-delimited **parsed_xy_data_0** property that it generates.

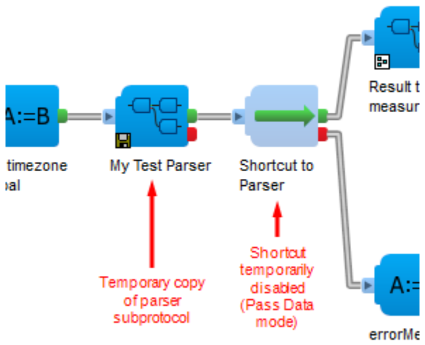
Development Tips

Here are some tips for creating a parser subprotocol:

- **Base your subprotocol on an existing parser subprotocol**, such as the *CSV Parser* that is supplied with the Foundation Hub Component Collection.
- **Test the subprotocol in the Pipeline Pilot client**, by running the *Parse* protocol with appropriate parameters:
    - *Instrument ID:* The ID of an instrument of a type that creates data files to be processed by the parser subprotocol.
    - *Data File Path:* The path of a data file generated by the instrument. If you know the format of the data, you may find it convenient to test using a dummy file, varying its content as appropriate.
- **Use reader components to extract data from the source file.** Pipeline Pilot has an extensive library of reader components to facilitate data extraction. Components that you might find helpful include *Delimited Text Reader*, *JSON Reader*, *Excel Reader*, *Text Reader*, *XML Reader*, and *PDF Reader*. To find a reader suitable for your purpose, search on `Reader` in the **Components** tab in the Pipeline Pilot client.
- **Debug using a copy of the parser subprotocol rather than the shortcut.** The Pipeline Pilot client only locates errors in the protocol that you are running, and not from subprotocols accessed through shortcuts. Therefore, if an error occurs in the parser subprotocol, it will simply be flagged in the *Shortcut to Parser* component:

  This is not very useful for debugging, since it does not indicate where in the subprotocol the error occurred. To locate errors in subprotocol components, you can temporarily disable *Shortcut to Parser*, and copy the parser subprotocol directly into the *Parse* protocol:

  
  timezone
My Test Parser
Shortcut to Parser
Result t measur
errorMe
Temporary copy of parser subprotocol
Shortcut temporarily disabled (Pass Data mode)

  IMPORTANT!
  When development is complete, ensure that:
    - You transfer any changes that you made in the copy of the subprotocol to the original subprotocol.
    - You revert all temporary changes that you made to the *Parse* protocol. (You may prefer to work with a copy of the *Parse* protocol, to avoid accidental changes to the original one.)
- **Ensure that the parser subprotocol deletes temporary files that it creates.** If Foundation Hub finds

unexpected files in parsing job folders, it might not process measurements correctly.

- **Use debugging components.** Use *Data Record Tree Viewer* and *Global Data Tree Viewer* to track how data changes through pipelines.

Performance Considerations

These are several ways that you can optimize performance of your parser subprotocol:

- **Avoid unnecessary creation of temporary files, and reading from temporary files.** Where possible, store temporary data in properties instead of files.
- **Prune cloned subprotocols of irrelevant components and code.** If you are basing your subprotocol on an existing subprotocol, there may be many components that you can safely remove without affecting your output. This can improve performance dramatically. The same applies to lines of PilotScript (or Perl, Java, Python, and so on) that are not relevant to your design.
- **Remove components and code that you used for debugging.** Debug components and code typically write to the Pipeline Pilot client user interface, properties, or files. These actions slow down performance, as well as cluttering your subprotocol. Once development is complete, remove content that you used for debugging.

Example: CSV Parser

This section describes the *CSV Parser* subprotocol that is provided with the Foundation Hub Component Collection. It is a relatively simple parser, and therefore provides a good illustration of the main features required of a parser subprotocol and its associated Foundation Hub resources.

The *CSV Parser* is at the following location:

Protocols\Database and Application Integration\Application Integration\Laboratory\Resources\Equipment\Parsers\CSV Parser

The *CSV Parser* analyzes a CSV file for fields whose names match the names in an instrument's data packet.

The *CSV Parser* is designed to parse source files with either of the following structures:

- With header data:
  - Line 1: Names of header fields.
  - Line 2: Values of header fields.
  - Line 3: Empty. (The *CSV Parser* ignores this line, so strictly speaking it can contain anything.)
  - Line 4: Names of table fields.
  - Lines 5 onward: Values of table fields. Each line represents a separate record.
- Without header data:
  - Line 1: Names of table fields.
  - Lines 2 onward: Values of table fields. Each line represents a separate record.

Notes:

- The *CSV Parser* does not require that all lines have the same number of commas. It only requires that the numbers of columns within the header and table sections are consistent. In thus far, it does not enforce compliance with the CSV standard recommendation RFC 4180.
- If you save a CSV file from Microsoft Excel, it is padded with empty cells, so that all lines contain the same number of commas. This format is accepted by the *CSV Parser*.

If you want to create a parser that processes CSV files with a different structure, you may be able to base it on the *CSV Parser* with relatively small modifications.

Example CSV File

An example source file for the *CSV Parser* is included with the Foundation Hub Component Collection. It is installed at the following location:

```
<pps_install>\apps\scitegic\hub\dataroot\data\Equipment\Parsers\
sampleParserFiles\ExampleAirAnalyzer.csv
```

This file simulates the output of an instrument that performs simple air analysis. The content of the file is as follows:

```
Instrument Operator,Comment
Janet Smith,Note sharp drop in humidity since yesterday due to wind veering south

Air Sample ID,Date+Time,Not for Display,Air Temperature,Air Humidity
1,3/19/2017 13:30,2544.51687,23.64340634,46.73440332
2,3/19/2017 14:30,4564.44346,23.28412678,48.77543443
3,3/19/2017 15:30,111.16553,23.21111095,46.77895568
```

Note the following points:

- There are two header data fields: **Instrument Operator** and **Comment**.
- Line 3 is empty, because the *CSV Parser* requires table data to start on Line 4, if there is header data.
- There are five columns of table data, named on Line 4: **Air Sample ID**, **Date+Time**, **Not for Display**, **Air Temperature**, and **Air Humidity**.
- There are three records of table data, starting on Line 5.

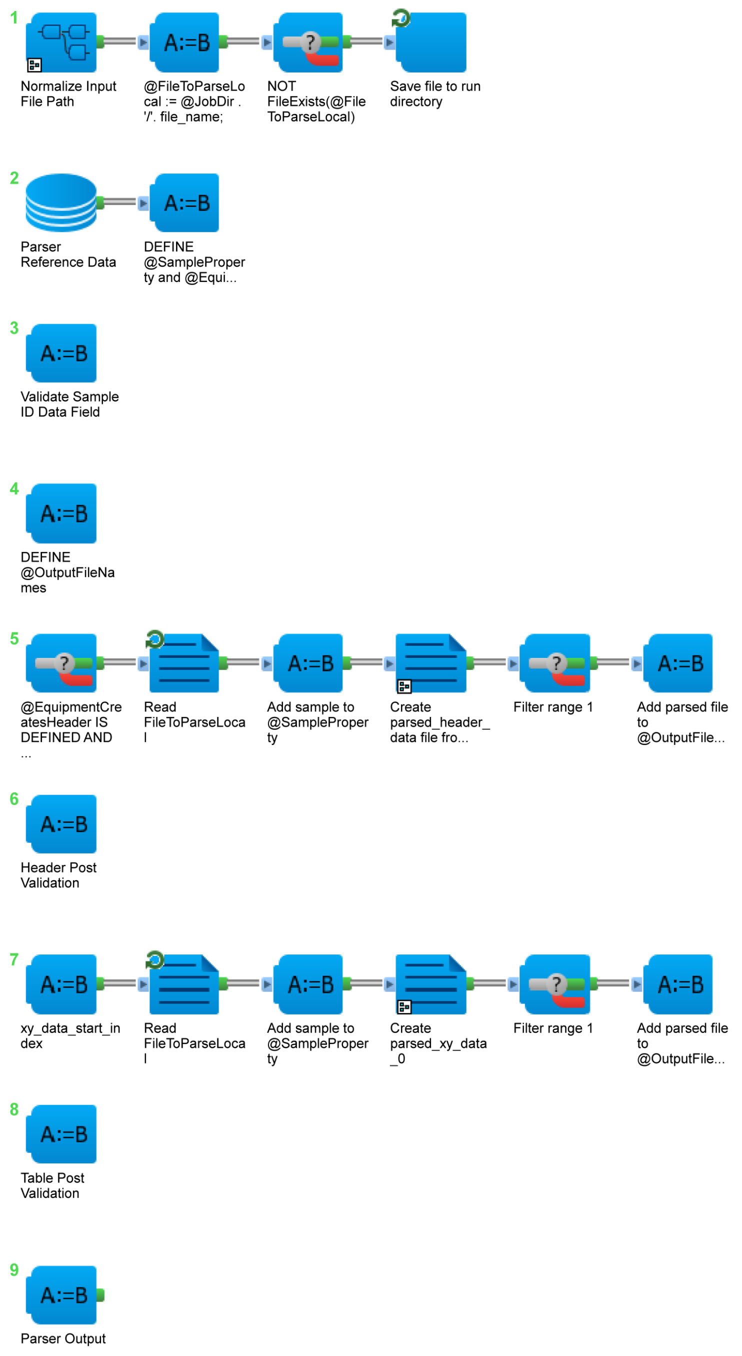
Foundation Hub Resources Example

The Foundation Hub Component Collection includes a set of Foundation Hub resources that are required to parse the example CSV file. For a detailed description, see Foundation Hub Resources Example.

How the CSV Parser Processes Data

This section provides an overview of the operation of the *CSV Parser* subprotocol. To understand in detail how it works, study its components in the Pipeline Pilot client, paying attention to the Sticky Notes and PilotScript code comments.

The top level of the *CSV Parser* subprotocol is shown on the next page as it appears in the Pipeline Pilot client. Following this is a summary of the protocol's pipes.


1
Normalize Input File Path
@FileToParseLocal := @JobDir . '/'. file_name;
NOT FileExists(@FileToParseLocal)
Save file to run directory
2
Parser Reference Data
DEFINE @SampleProperty and @Equi...
3
Validate Sample ID Data Field
4
DEFINE @OutputFileNames
5
@EquipmentCreatesHeader IS DEFINED AND ...
Read FileToParseLocal
Add sample to @SampleProperty
Create parsed_header_data file fro...
Filter range 1
Add parsed file to @OutputFile...
6
Header Post Validation
7
xy_data_start_index
Read FileToParseLocal
Add sample to @SampleProperty
Create parsed_xy_data_0
Filter range 1
Add parsed file to @OutputFile...
8
Table Post Validation
9
Parser Output

Top-Level Pipes: Summary

Pipe 1

- Specifies a file extension filter for the source data file. This is the value of the parameter *ParsedDataFileTypeExtension* of the component *Normalize Input File Path*. If the file name supplied by the *Parse* protocol does not have the specified extension, or if the file does not exist, an error is thrown.

The default value of *ParsedDataFileTypeExtension* is `csv`. If you create a parser subprotocol based on the *CSV Parser*, ensure that *ParsedDataFileTypeExtension* is set to the extension of files created by the instrument.

- Finds the source data file.
- Trims white space from the end of the file path.
- Saves a copy of the file to the parser job folder.

Pipe 2

- Reads the data packet's fields from the *reference data* cache. This cache is created by the *Parse* protocol before it calls the parser.
- Records which field in the data packet contains the sample ID.
- Records whether the data packet includes header fields. If a field's **Series** (when converted to lowercase) has the value `header`, *@EquipmentCreatesHeader* is set to `true`.

Pipe 3

- Verifies that there is a Sample ID field in the data packet.

Pipe 4

- Creates global parameters that will list the output files and sample names.

Pipe 5

- Processes the header fields from the data file, if there are any.
- Checks if the sample ID is in a header field, and if it is, adds it to the list of samples.
- Writes header data in JSON format to the file `parsed_header_data` in the job folder.
- Appends `parsed_header_data` to the list of output files.

Pipe 6

- Verifies that header data was generated if the data packet specified it.

Pipe 7

- Sets the starting line of table data in the data file. This is Line 1 if the file contains only table data, and Line 4 if there is header data.
- Processes the table fields from the data file.
- Checks if the sample ID is in a table field, and if it is, adds it to the list of samples.
- Writes table data to the file `parsed_xy_data_0`.
- Appends `parsed_xy_data_0` to the list of output files.

Pipe 8

- Verifies that table records contain at least one table field that is defined in the data packet.

Pipe 9

- Creates the data record property **Manifest**. This property lists the files that should have been output to the job folder. These must include, at a minimum, `parsed_xy_data_0`.

  The **Manifest** property is used subsequently by *Parse* protocol to upload the files to the Measurement Store.
- Creates the global property **@Samples**, which lists the sample IDs.

Chapter 20: Equipment Work Items, Parser Jobs, and Discovery Events

Each time a new equipment data file is found during a run of the File Crawler, an **equipment work item** is created. Equipment Work items are listed in the Equipment Work Items page.

From the Equipment Work Items page, you can also open pages for parser jobs and discovery events.

Equipment Work Items

When the File Crawler finds an equipment data file, it creates an equipment work item for it. The association between the equipment and the work item is maintained for as long as the file exists.

Equipment work items are listed in the table at the following page:

Admin and Settings > Equipment > Equipment Work Items

Each row in the table contains the following details of the work item:

- **Equipment:** The device that provided the reading in the file.
- **Work Item Identifier:** The name of the file that was generated for the work item. This acts as a unique identifier for the work item.
- **State:** The state of the work item. Possible states are as follows:
  - NEW: The File Crawler has found a new file, but the file has not yet been parsed. (If a free parsing slot is available, the File Crawler starts a parsing job for the work item, and sets the Work Item's state to PARSING.)
  - UPDATED: The file has been updated since it was last parsed. This state is used with equipment that appends new readings to an existing file, rather than creating a new file for each reading.
  - PARSING: The File Crawler has sent the file to a parsing protocol for processing.
  - COMPLETED: The file was parsed successfully. The result of parsing is shown in the Last Parse Result column. A corresponding measurement is created and viewable from the Measurements page.
  - IGNORED: The file has been imported into the Equipment Work Items list, but creation of parser jobs for the file is disabled.
  - PARSER_ERROR: The File Crawler called the parsing protocol, but received an error message back.

    The maximum number of calls that the File Crawler makes to the parsing protocol is configured in the setting **Attempts on Equipment File Parsing**. For more information, see Configuring the File Crawler.

    If parsing fails on the first attempt, the File Crawler waits 2 minutes before trying again. The waiting period doubles after each failed attempt. Thus, the waiting periods are 2, 4, 8, and 16 minutes, and so on. After the maximum number of failed attempts, the File Crawler moves the work item to the FAILED state.

    The default maximum number of parsing attempts is 5.
  - FAILED: The File Crawler made the maximum number of requests to the parsing protocol, but all attempts to parse the file failed.
- **Work Item Last Updated:** The date and time of the last update of the data file.

- **Last Parse Date:** The data and time of the last parse of the file by its parsing protocol.
- **Last Parse Result:** The result of the last parse of the file.

  Note: If the parser generated an error, the error message is supplied as the parse result.

Viewing Full Details of a Work Item

To view full details of a work item:

- Click anywhere in the row of the work item, except for the first two cells.

  A page opens that shows the following information for the work item:

  - **Equipment:** The device that provided the reading.
  - **Work Item Identifier:** The name of the file that was generated for the work item. This acts as a unique identifier for the work item.
  - **Work Item Size:** The size of the file that was generated for the work item.
  - **Work Item Last Updated:** The date and time of the last update of the data file.
  - **Additional Context:** Additional context to be forwarded to the parser.

  Parse Result:

  - **State:** The state of the work item. For descriptions of states, see Equipment Work Items.
  - **Last Parse Result:** The result of the last parse of the file. The information is presented in an easily readable format that includes line breaks.

    Note: If the parser generated an error, the error message is supplied as the parse result.

  - **Last Parse Duration:** The duration of the last run of the parsing protocol, in milliseconds.
  - **Last Parse Date:** The date and time of the last parse of the file by its parsing protocol.

Other Actions

In the Equipment Work Items page, you can also perform these actions:

- To view details of the equipment associated with a work item, click the hyperlink in the **Equipment** column.
- To open the Parser Jobs and Work Item Discovery Events pages, click the hyperlinks under **Related Items**.
- To delete a work item, select the box in its left cell, and then click ⛔.

  IMPORTANT! Be VERY CAREFUL when deleting work items. If you delete a work item for a file, the File Crawler will not add it back. The file will never be parsed again. Only delete a work item if you know that its data file is badly formatted and can never be parsed. You should not delete a work item for any other reason.

Parser Jobs

The Parser Jobs page provides a view of the queue of parser jobs waiting to be run on equipment work items. Every time the File Crawler finds a new file for parsing, or an existing file that has been updated, the file's work item is added to the Parser Jobs queue.

To view the Parser Jobs page, go to the Equipment Work Items page , and click **Parser Jobs**.

Each row in the table contains the following details of the parser job:

- **Score:** A statistical score that is used to determine the job's priority in the queue. For details of how the score is calculated, see Parser Job Scoring Algorithm.
- **Equipment Work Item:** The name of the file that was generated for the work item. This acts as a unique identifier for the work item.
- **Work Item State:** The state of the work item. Possible states are as follows:
  - NEW: The File Crawler has found a new file, but the file has not yet been parsed. (If a free parsing slot is available, the File Crawler starts a parsing job for the work item, and sets the Work Item's state to PARSING.)
  - UPDATED: The file has been updated since it was last parsed. This state is used with equipment that appends new readings to an existing file, rather than creating a new file for each reading.
  - PARSING: The File Crawler has sent the file to a parsing protocol for processing.
  - COMPLETED: The file was parsed successfully. The result of parsing is shown in the Last Parse Result column. A corresponding measurement is created and viewable from the Measurements page.
  - IGNORED: The file has been imported into the Equipment Work Items list, but creation of parser jobs for the file is disabled.
  - PARSER_ERROR: The File Crawler called the parsing protocol, but received an error message back.

    The maximum number of calls that the File Crawler makes to the parsing protocol is configured in the setting **Attempts on Equipment File Parsing**. For more information, see Configuring the File Crawler.

    If parsing fails on the first attempt, the File Crawler waits 2 minutes before trying again. The waiting period doubles after each failed attempt. Thus, the waiting periods are 2, 4, 8, and 16 minutes, and so on. After the maximum number of failed attempts, the File Crawler moves the work item to the FAILED state.

    The default maximum number of parsing attempts is 5.
  - FAILED: The File Crawler made the maximum number of requests to the parsing protocol, but all attempts to parse the file failed.
- **Work Item Last Updated:** The date and time of the last update of the data file.
- **Priority:** The priority of the job, inherited from the equipment. For more information, see Parser Job Scoring Algorithm.
- **Assigned Pipeline Pilot:** The instance of Pipeline Pilot Server that runs the parsing protocol. Clicking the hyperlink takes you to the Pipeline Pilot Server home page.
- **Schedule Start:** The date and time when the job's file was last updated, or parsing of the file failed.
- **Start Date:** The date when the job was started.
- **Average Duration:** The average duration of previous parsing jobs.
- **Attempt:** The index of the parsing attempt. (1 is the first parsing attempt, 2 the second, and so on.)
- **Last Parse Date:** The data and time of the last parse of the file by its parsing protocol.
- **Last Parse Result:** The result of the last parse of the file.

  Note: If the parser generated an error, the error message is supplied as the parse result.

Parser Job Scoring Algorithm

The score of a parser job is calculated from the following properties:

Property	Impact	Description
Age	1 second = 1 point	This is the baseline. Older files have a higher priority. This prevents files from "starving" in the jobs queue. If an equipment file's date is earlier than the date that the equipment was added to Foundation Hub, a one-year penalty is applied, rather than a positive value. This is because the File Crawler assumes that the file is an obsolete one from an older instantiation of the equipment, and should not be allowed to block other jobs.
Average executions per day	**100 executions per day:** 3600 points **10 executions per day:** 1200 points **1 execution per day:** 600 points **< 1 execution per day:** 0 points	If a device is used more frequently, its jobs' priority is slightly increased.
Average job duration	1 second = -2 points	The longer a parsing job normally takes, the greater the penalty.
Priority	**Very Low:** -31536000 points (-365 days) **Low:** -2592000 points (-30 days) **Medium:** 0 points **High:** 2592000 points (30 days) **Critical:** 15552000 points (180 days) **Immediate:** 31536000 points (365 days)	The priority is inherited from the equipment. Of all criteria, the priority has the biggest impact on a job's score. A priority of Very Low is only assigned internally if the equipment work item is older then the equipment. A priority of Immediate is assigned if immediate job parsing is requested. This priority cannot be inherited from equipment.

Work Item Discovery Events

The Work Item Discovery Events page lists events associated with the File Crawler's activity. It can be useful for troubleshooting equipment file crawling problems, because it includes error messages from the File Crawler. For example, if you configured a device for file crawling, but no work item is visible for it in the Equipment Work Items list after the next scheduled run of the crawler, you can look in the Work Item Discovery Events page for an `Error` event that explains why the work item could not be created. (Perhaps the equipment's **Data Folder Path** was invalid.)

To view the Work Item Discovery Events page, go to the Equipment Work Items page, and click **Work Item Discovery Events**.

Each row in the table contains the following details of a discovery event:

- **Timestamp:** The date and time of the event.
- **Event:** The event type, for example `Success` or `Error`.

- **Target Object:** The object affected by the event. This is the ID of the equipment.
- **Description:** Details of the event.

Viewing Full Details of a Work Item Discovery Event

To view full details of a work item discovery event:

- Click anywhere in the row of the event.

  A page opens that shows the following information for the event, where available:

  - **Timestamp:** The date and time of the event.
  - **Target Object:** The object affected by the event. This is the ID of the equipment.
  - **Reason Code:** An internal code for the discovery event. This field may be empty.
  - **Description:** Details of the event.
  - **Performed By:** The user account of the File Crawler.
  - **Event:** The event type, for example `Success` or `Error`.
  - **Category:** The category of the event, for example `EquipmentFileDiscovery`.

  Client Information:

  This section contains information about the client device that triggered the event.

Chapter 21: Measurements

A measurement is a package of data that Foundation Hub has generated from an equipment reading. The **Measurement Store** is collection of measurements from devices that are configured with direct-connect, advanced, file-based, or adapter connection types in Foundation Hub.

Measurements can be generated in several different ways, depending on how the device is connected. For example:

- Automatic execution of the *Parse* protocol on a file discovered by the File Crawler.
- Manual execution of the *Parse* protocol in the Pipeline Pilot client. See Testing the Parsing Framework.
- Automatic execution of the *Parse* protocol after manual upload of a data file. See Uploading Equipment Data Files Manually.

The Measurements Page

The **Measurements** page shows summary information about each measurement package in the Measurement Store. To view the Measurements page, open **Admin and Settings > Resources > Measurements**.

From the Measurements page, you can open a page showing full details of individual measurements. You can also view the details of the equipment that generated each measurement. For more information, see Viewing Full Measurement Details and Viewing Details of a Measurement's Equipment.

Fields in the Measurements Table

For each measurement package, the following information is shown:

- **Display Name:** The automatically-generated display name of the measurement package. This begins with the name of the output file of parsed data, and also includes the Barcode and Nickname of the equipment that generated the data.
- **Record Date:** The date and time when the raw data was last updated in the file system.
- **Equipment:** The Barcode of the equipment that produced the measurement.
- **Date Created:** The date and time when the measurement was uploaded to Foundation Hub. This is approximately when the raw data was parsed.
- **Last Updated:** The date and time when the data file was last modified in Foundation Hub.
- **Context:** A reference to the context of the measurement. For example, a Capture recipe execution or another task.

Viewing Full Measurement Details

To view full details of a measurement:

1. Open **Admin and Settings > Resources > Measurements**.
2. Click the row of the measurement in the list of measurements.

   Note: Clicking the hyperlink in the **Equipment** column opens the Equipment page, and not the Measurement Details page. See Viewing Details of a Measurement's Equipment.

The **Measurement Details** page opens. There, you can view full information about the measurement, and open its attachments.

Viewing Details of a Measurement's Equipment

To view the Equipment page for the instrument that generated a measurement:

1. Open **Admin and Settings > Resources > Measurements**.
2. Click the hyperlinked equipment name in the **Equipment** column of the measurement.

Measurement Details

The **Measurement Details** page shows the data generated from an equipment reading. It can also show general header data about the measurement and attachment files associated with the measurement, if applicable.

Viewing Measurement Details

By default, the Measurement Details page opens in Grid View, which lists the readings in the measurement file. You can toggle between Grid View and Header View, which shows general information about the measurement, if available.

To view details of a measurement:

1. Open **Admin and Settings > Resources > Measurements**.
2. Click the measurement you want to view.

   The main panel shows the raw data string, if applicable, and parsed data in Grid View.
3. To toggle between views, click **Grid View** or **Header View**.

Raw Data

For direct-connected equipment, the raw data string that the device sent to Foundation Hub appears above any parsed data in Grid View or Header View.

Grid View

The data listed in Grid View varies depending on the device configuration and how the data is parsed. However, these fields are always present:

- **Mapped Sample Id:** The sample ID to which the measurement reading is mapped in Foundation Hub. If the sample was mapped automatically when the measurement was taken, this is the same as the sample ID in the measurement reading. It can be different, however, if a user mapped the ID manually during execution of a data acquisition task. For more information, see "Executing Data Acquisition Tasks" in the *BIOVIA Foundation Hub Help*.
- **Instrument Sample Id:** The sample ID to which the reading applies. The **Is Sample Id** flag in the data packet determines which field this is in the source data.
- **Row:** The record's row in the parsed data table. Rows are numbered from the first data record. For example, if the parsed data file is an Excel spreadsheet, and table data starts on Row 6 (with Rows 1 to 5 containing metadata and column headings), then Row 1 in the Grid View table corresponds to Row 6 in the Excel file.

Header View

The Header View shows information from the parsed file. Header information is typically metadata that is not specific to any individual results (for example, the instrument operator or comments).

The fields shown in the Header View are specific to the format of the parsed data. This format is defined in the data packet details of the instrument's type.

If the measurement does not have header data, the Header View is empty, with just the message, "No header data is available".

Viewing Attachments

The **Attachments** panel contains links to files associated with the measurement. By default, it shows just the source file or files of the measurement. You can expand the list to include system files.

Opening an Attachment

To open an attachment:

- In the **Attachments** panel, click the entry for the attachment that you want to open.

  The attachment opens in the associated application (for example, Microsoft Excel if the file is in XLSX or CSV format).

Showing System File Attachments

By default, the **Attachments** panel shows only the final parsed data files. It does not show the intermediary files (the "system" files) that are generated during parsing. To show the system files as well:

- Select the **Display System Files** box.

The system files can be in various formats, such as CSV or JSON.

Expanding and Collapsing the Attachments Panel

The Attachments panel is expanded by default.

- To collapse the panel, click the icon ‹ in the title bar.
- To expand a collapsed panel, click the icon › in the title bar.

Chapter 22: Administering the File Crawler

The Equipment File Crawler is a Pipeline Pilot protocol that Foundation Hub runs at regular intervals. You configure the File Crawler in Foundation Hub on the following page:

Admin and Settings > Applications Application Settings > Foundation Hub

Configuring the File Crawler

To configure the Equipment File Crawler:

1. In Foundation Hub, go to **Admin and Settings > Applications > Application Settings > Foundation Hub**.
2. Click **Edit**.
3. Under **Equipment**, configure the following settings (note that settings not related to the File Crawler are not listed here):
   - **Parse Instrument File Protocol:** The path of the *Parse* protocol that parses equipment data files. See The Parse Protocol.
   - **Enable Equipment File Crawler:** When set to Yes, causes the File Crawler to run at regular intervals, as specified by **Equipment File Crawler Schedule**. If not selected, the File Crawler is disabled. See Enabling the File Crawler.
   - **Equipment File Crawler Schedule:** The days of the week, and the time of day, at which the File Crawler runs.

     In the **Hours** and **Minutes Past Hour** boxes, specify the hours of the day, and the minutes of the hour, when the File Crawler runs. You can use CRON formats, for example:

Hours	Minutes Past Hour	Meaning
0/1	0/1	Every minute of the day.
2	30	At 02:30.
0,6,12,18	0	At 00:00, 06:00, 12:00, and 18:00.
0,6,12,18	0,30	At 00:00, 00:30, 06:00, 06:30, 12:00, 12:30, 18:00, and 18:30.
0/1	0	Every hour, on the hour.
0/1	0,30	Every half hour.
6-23	0/1	Every minute between 06:00 and 23:00.
6-23	0-14	Every minute of the first quarter-hour of every hour between 06:00 and 23:00.

   - **Equipment File Crawler Protocol:** The path of the File Crawler protocol.
   - **Maximum Active Crawlers:** The maximum number of File Crawler instances that can run simultaneously.

This setting prevents unchecked generation of new File Crawler instances when the File Crawler is running slowly. A reason for slow running of the File Crawler might be the presence of a large number of old data files in the crawled folders.

- **Maximum File Age to be sent for Parsing:** The maximum age, in days, that a data file can be if it is to be parsed by the Parsing Framework. Work items for files older than this value enter the IGNORED state, and do not trigger parse jobs. (Foundation Hub administrators can still parse these files manually from the files' work item pages.)
- **Automatically Parse Equipment Work Items:** When set to Yes, causes the File Crawler to create parse jobs automatically for new and updated files that it finds. This is the appropriate value for normal running of the File Crawler.

  When set to No, causes automatic creation of parse jobs to be suspended. You might want to use this option if you are setting up your system, and you want to check that file discovery is working correctly, but are not ready to start parsing files.
- **Number of Equipment File Parse Jobs to run concurrently:** The maximum number of parse jobs that can run concurrently in Pipeline Pilot. The default is 10. You can increase or decrease this to suit the processing and memory capacities of your system. Some parse slots can be reserved for short-running jobs — see below.
- **Number of slots reserved for short Equipment File Parse Jobs:** Parse jobs for large files can take a long time to run (up to several hours). To avoid short parse jobs being held up by long ones, you can use this setting to dedicate a number of parse slots to short jobs. The default setting is 2. A short parse job is one whose duration (estimated from previous parse jobs of its equipment work item) is less than or equal to the value of **Maximum Duration of Short Equipment File Parse Jobs**.
- **Maximum Duration of Short Equipment File Parse Jobs:** The maximum duration, in seconds, of a parse job for it to be considered short, and therefore eligible to be sent to a parse slot reserved for short-running jobs.
- **Attempts on Equipment Work Item Parsing:** The maximum number of attempts made to parse a file before the work item's state is set to FAILED. The default value is 5.

  If parsing fails on the first attempt, the File Crawler waits 2 minutes before trying again. The waiting period doubles after each failed attempt. Thus, the waiting periods are 2, 4, 8, and 16 minutes, and so on. After the maximum number of failed attempts, the File Crawler moves the work item to the FAILED state.

  For more information, see Equipment Work Items.
- **Pipeline Pilot Job Expiration:** The amount of time, in seconds, that Pipeline Pilot keeps a job's results after its completion. This setting provides a mechanism for Foundation Hub to retrieve job results after a loss of connectivity between Foundation Hub and Pipeline Pilot Server. The default value is 600 (that is, 10 minutes).

4. Click **Save**.

Enabling the File Crawler

After you install Foundation Hub, the File Crawler is disabled by default.

To enable the Equipment File Crawler:

1. In Foundation Hub, go to **Admin and Settings > Applications > Application Settings > Foundation Hub**.
2. Click **Edit**.

3. Under **Enable Equipment File Crawler**, select **Yes**.
4. Click **Save**.

Chapter 23: Getting Parser Statistics for File Crawler and Parsing Reports

The Foundation Hub REST API includes endpoints that you can use to retrieve statistics about the performance of parser subprotocols. You can call these endpoints from your own client code to generate formatted reports of File Crawler and Parsing Framework activity.

The root URI of all parser statistic endpoints is the same as for other Foundation Hub endpoints; that is:

```
http://<server>:<port>/foundation/hub/
```

The endpoints are summarized in the table below. More detailed descriptions, with examples, are shown in Endpoint Details.

For full descriptions of these endpoints, and others, see the *BIOVIA Foundation Hub API Reference Guide*. (Click the **?** icon in the top toolbar of Foundation Hub, and select **API Reference Guide**.)

HTTP method and REST endpoint	Description
`GET /api/v1/parserstats`	Retrieves statistics for all parsers
`GET /api/v1/parserstats/<parser ID>`	Retrieves statistics for a single parser
`DELETE /api/v1/parserstats/<parser ID>`	Resets the statistics for a parser
`GET /api/v1/parserstats?$show=deleted`	Retrieves deleted statistics for all parsers
`GET /api/v1/equipment/<equipment ID>/equipmentworkitems`	Retrieves work items associated with a device

Endpoint Details

GET /api/v1/parserstats

Retrieves statistics for all parsers. Statistics are provided for individual instruments, equipment types, and the File Crawler. The abridged example below shows one instance of each of these sets of statistics.

Example:

```
[
    {
        "id": "6e93c3e3-74d1-4e7f-b14f-83da9766226d",
        "averageDuration": 10507,
        "averageRunsPerDay": 1,
        "description": "Parser stats for single equipment [5087]. Parser
path
[Protocols\\Database and Application Integration\\Application
Integration\\Laboratory\\Resources\\Equipment\\Parsers\\CSV Parser].",
        "equipment": {
            "id": "56341974-956b-4298-b9e6-82a46077fde9",
            "barcode": "5087",
            "displayName": "5087",
            "equipmentType": {
                "id": "f0549e5f-2a6d-4766-8202-5bb675aa3ec9"
            },
            "gxpLevel": {
                "id": "3be4df3c-82c1-40b7-b1ea-51ce189b3375"
            }
        },
        "equipmentScore": -20,
        "equipmentType": null,
        "errorCount": 2,
        "errorStreakCount": 0,
        "lastDuration": 12708,
        "lastError": "2018-04-03T21:15:02Z",
        "lastReset": "2018-04-03T21:12:02Z",
        "lastSuccess": "2018-04-06T13:36:20Z",
        "lastUpdated": "2018-04-06T13:37:00Z",
        "successCount": 3,
        "successStreakCount": 3
    },

    ...

    {
        "id": "6f9a83b5-c6c4-49a0-b152-acf7204d0e23",
        "averageDuration": 10507,
        "averageRunsPerDay": 4,
        "description": "Parser stats for every equipment of equipment type
[E
xample Air Analyzer Equipment Type]. Parser path [Protocols\\Database and
Application Integration\\Application
Integration\\Laboratory\\Resources\\Equipment\\Parsers\\CSV Parser].",
        "equipment": null,
        "equipmentScore": -20,
        "equipmentType": {
            "id": "f0549e5f-2a6d-4766-8202-5bb675aa3ec9",
            "connectionType": "File",
            "dataPacket": {
                "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
                "name": "Example Air Analysis"
            },
            "equipmentClass": {
                "id": "71c3040a-64bc-4d01-9772-c98529885c5b",
                "name": "Example Air Analysis Equipment Class"
            },
            "lifeCycleVersion": "1",
            "name": "Example Air Analyzer Equipment Type"
        },
        "errorCount": 2,
        "errorStreakCount": 0,
        "lastDuration": 12708,
        "lastError": "2018-04-03T21:15:02Z",
        "lastReset": "2018-04-03T21:13:00Z",
        "lastSuccess": "2018-04-06T13:36:20Z",
        "lastUpdated": "2018-04-06T13:37:00Z",
        "successCount": 3,
        "successStreakCount": 3
    },

    ...

    {
        "id": "9419a11b-682d-4221-a05d-b62862be134d",
        "averageDuration": 250,
        "averageRunsPerDay": 0,
        "description": "Crawler protocol stats.",
        "equipment": null,
        "equipmentScore": -1920,
        "equipmentType": null,
        "errorCount": 21318,
        "errorStreakCount": 10578,
        "lastDuration": 0,
        "lastError": "2018-04-11T19:55:02Z",
        "lastReset": "2018-03-28T00:22:04Z",
        "lastSuccess": "2018-04-04T11:32:10Z",
        "lastUpdated": "2018-04-11T19:55:02Z",
        "successCount": 7,
        "successStreakCount": 0
    }
]
```

GET /api/v1/parserstats/<parser ID>

Retrieves statistics for a single parser. Results are collated from all devices that use the parser.

Example:

```
{
    "id": "6f9a83b5-c6c4-49a0-b152-acf7204d0e23",
    "averageDuration": 10507,
    "averageRunsPerDay": 0,
    "description": "Parser stats for every equipment of equipment type
[Example
Air Analyzer Equipment Type]. Parser path [Protocols\\Database and
Application Integration\\Application
Integration\\Laboratory\\Resources\\Equipment\\Parsers\\CSV Parser].",
    "equipment": null,
    "equipmentScore": -20,
    "equipmentType": {
        "id": "f0549e5f-2a6d-4766-8202-5bb675aa3ec9",
        "connectionType": "File",
        "dataPacket": {
            "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
            "name": "Example Air Analysis"
        },
        "equipmentClass": {
            "id": "71c3040a-64bc-4d01-9772-c98529885c5b",
            "name": "Example Air Analysis Equipment Class"
        },
        "lifeCycleVersion": "1",
        "name": "Example Air Analyzer Equipment Type"
    },
    "errorCount": 2,
    "errorStreakCount": 0,
    "lastDuration": 12708,
    "lastError": "2018-04-03T21:15:02Z",
    "lastReset": "2018-04-03T21:13:00Z",
    "lastSuccess": "2018-04-06T13:36:20Z",
    "lastUpdated": "2018-04-06T13:37:00Z",
    "successCount": 3,
    "successStreakCount": 3
}
```

DELETE /api/v1/parserstats/<parser ID>

Resets the statistics for a parser, as follows:

```
lastReset = now
errorCount = 0
successCount = 0
=> averageRunsPerDay = 0
=> equipmentScore = 0
```

The response to this request is a 204 "No Content" status message.

GET /api/v1/parserstats?$show=deleted

Retrieves statistics for all parsers, but with deleted statistics included. See GET /api/v1/parserstats.

GET /api/v1/equipment/<equipment ID>/equipmentworkitems

Retrieves work items associated with a device.

Example:

```
[
    {
        "id": "e79bee9c-1a35-4fbe-974c-e658779ca2fc",
        "absoluteFilePath": "$(/pilot_
data/scitegic/hub)/data/Equipment/Parsers/
sampleParserFiles/ExampleAirAnalyzer.csv",
        "additionalContext": null,
        "equipment": {
            "id": "56341974-956b-4298-b9e6-82a46077fde9",
            "barcode": "5087",
            "displayName": "5087",
            "equipmentType": {
                "id": "f0549e5f-2a6d-4766-8202-5bb675aa3ec9"
            },
            "gxpLevel": {
                "id": "3be4df3c-82c1-40b7-b1ea-51ce189b3375"
            }
        },
        "fileReferences": null,
        "lastParseDate": "2018-04-06T13:36:07Z",
        "lastParseDuration": 12708,
        "lastParseResult": "{\"data\":{\"id\":\"8743e9b9-8b25-4ab1-bfbc-
921f16b37
0fd\",\"urn\":\"urn:measurement:8743e9b9-8b25-4ab1-bfbc-
921f16b370fd\"},\"logbook
Id\":\"d2413c7c-3fab-40b2-990b-2bc2a588d413\",\"results\":null}",
        "lastUpdated": "2018-04-06T13:36:20Z",
        "state": "COMPLETED",
        "workItemCreated": "2018-04-05T21:46:42Z",
        "workItemIdentifier": "ExampleAirAnalyzer.csv",
        "workItemLastUpdated": "2018-04-05T21:46:42Z",
        "workItemSize": 351
    }
]
```

Chapter 24: Uploading Equipment Data Files Manually

Occasionally, a file that is output by an instrument might not be discoverable by the File Crawler because of issues such as network outages, or temporary disconnection of the instrument. In such events, you can manually upload the file to Foundation Hub's Data Preview page. Foundation Hub immediately parses the file, and writes its data to the Measurement Store if parsing is successful.

You can also use the manual file upload feature to import data from instruments that are permanently disconnected from Foundation Hub's network.

Note: You can upload files in CSV format, even if the instrument normally generates files in a different format. See Uploading CSV Files.

To upload a data file manually:

1. In Foundation Hub, create a data acquisition task:
   a. In **Admin and Settings**, select **Resources > Activities >** , and create an activity with an activity type of **data acquisition**.
   b. In the Foundation Hub Landing Page, click in the My Task Plans widget to open the My Task Plans page.
   c. If there is not already a suitable task plan to which you can add the data acquisition task, click **Create** to create a new task plan.
   d. In the **Samples** tab, add one or more samples.
   e. In the **Tasks** tab, click to create a new activity. In the **Activities** list, select the data acquisition activity that you created in Substep a., and click **OK**.

      A data acquisition task is created and added to the **Tasks** list.
   f. Click to edit the data acquisition task. Add a sample or samples to the task, and click **OK**.
2. Execute the data acquisition task:
   a. Still in the Task Planner, select the data acquisition task, and click **Execute**.

      The Data Preview page opens in a new browser tab or window.
   b. In the **Search Equipment** list, select the instrument for which you want to upload data, and click **Add Equipment**.

      A panel opens that lists the existing work items for the equipment, if there are any.
   c. Click **Upload Instrument Data**.
   d. Click **Browse**, and choose the data file the you want to upload.
   e. Click **Upload Files**.

      A work item is created for the uploaded file, and added to the equipment's work items list.

      Note: Work items for manually uploaded files have the prefix **manual_upload**, followed by a time stamp. This enables them to be easily distinguished in the Equipment Work Items administration page. For details, see Equipment Work Items.

      Foundation Hub starts a parser job for the file. The parser job uses the appropriate parser subprotocol for the equipment's type. For more information, see Parser Subprotocols.

The status of the parser job is indicated to the right of the work item. Initially, this is QUEUED or PARSING.

f. Wait a few seconds, and then click 🔄 to refresh the work items list.

   If parsing is successful, the status of the work item changes to COMPLETED. This means that the file's data has been successfully added to the Measurement Store.

g. To view the uploaded file's parsed measurement data, click the file's entry in the list. This information is also available in the page **Admin and Settings > Resources > Measurements**. See Measurement Details.

Uploading CSV Files

You can upload data files in CSV format for any instrument, regardless of the usual format of the instrument's files. This can be useful if you need to create data files manually, since you do not need to know the usual file format.

When you upload a CSV file, the Parsing Framework uses the *CSV Parser* subprotocol to parse the file. The *CSV Parser* looks for columns in the file data that match data field names in the equipment type's data packet.

Note that if CSV is the normal file format defined for the equipment type, the *CSV Parser* is not necessarily used. Instead, the Parsing Framework uses whichever parser is specified for the equipment type. This might be the *CSV Parser*, or it might be a custom CSV-processing subprotocol.

CSV File Format

Column names in a CSV data file must exactly match field names in the data packet associated with the instrument. Column names that do not match data field names are ignored. For more information, see Data Packets.

The *CSV Parser* can process files with or without header data. The formats are as follows:

- With header data:
  - Line 1: Names of header fields.
  - Line 2: Values of header fields.
  - Line 3: Empty. (The *CSV Parser* ignores this line, so strictly speaking it can contain anything.)
  - Line 4: Names of table fields.
  - Lines 5 onward: Values of table fields. Each line represents a separate record.
- Without header data:
  - Line 1: Names of table fields.
  - Lines 2 onward: Values of table fields. Each line represents a separate record.

Notes:

- The *CSV Parser* does not require that all lines have the same number of commas. It only requires that the numbers of columns within the header and table sections are consistent. In thus far, it does not enforce compliance with the CSV standard recommendation RFC 4180.
- If you save a CSV file from Microsoft Excel, it is padded with empty cells, so that all lines contain the same number of commas. This format is accepted by the *CSV Parser*.

The *CSV Parser* automatically determines which format to use, based on whether the column names in the first row are for header fields or table fields. See Notes on Header Fields and Table Fields.

Part 6: Appendices

Appendix A: Troubleshooting Equipment

The Foundation Hub generates standard log files that you can use to troubleshoot issues with the Foundation Hub. See Foundation Hub Logging to learn how to adjust the level of reporting.

Equipment and instrument features are not enabled as expected

Check that you have a license for equipment and instrument functionality in **Settings > License Files**. If you add a license file, you *do not* need to stop or restart the Foundation Hub service.

Licenses are made available when the software is purchased. If you need assistance with licenses, contact Dassault Systèmes Customer Support.

Foundation Hub Logging

Note: Each node in a load-balanced deployment has its own log file. To figure out if an error occurred you must check all log files.

Viewing Log Files

1. Open **Settings > Logging**.
2. Click **Download Current Log File** or **Download All Log Files**.
3. See `<hub_install>/logs/hub.log`.

Editing Logging Settings

To manage Foundation Hub application logging, navigate to **Admin and Settings > Settings > Logging** and click **Edit**:

- **Application Logging Level:** Choose from **Debug**, **Info**, and **Warn** to set the level of verbosity. It is recommended that you leave the level set to **Warn**.
- **Log SQL Statements:** Logs a line for each SQL statement executed against the database. This option is verbose. It is not recommended for normal operation.
- **Log SQL Statement Parameters:** Logs a line for each parameter of each SQL statement executed against the database. This option is extremely verbose. It is not recommended for normal operation.
- **Download Current Log File:** Retrieves the active `hub.log` file from the server.
- **Download All Log Files:** Retrieves the current `hub.log` file, all archived log files (up to a maximum of ten) as well as the most recent `stderr` and `stdout` log files.
- **Output Elapsed Times:** Includes the time elapsed for various operations such as GET, PUT, and POST. This provides profiling information.
- **Include System Information:** Includes the system information (for example, host operating system and Java system information).

Tomcat Logging

Apache Tomcat logs additional information in its own log files:

- `stdout` (standard output)
- `stderr` (standard error)

Tomcat Http Access Logging

You can turn on Tomcat Http Access Logging in **Settings > Hub Configuration > Enable Tomcat Http Access Logging**.

Foundation Hub will record all requests processed by the server. With this enabled the following information is written to an access_log file in the log folder:

- Remote host name (or IP address if enableLookups for the connector is false)
- Date and time, in Common Log Format
- First line of the request (method and request URI)
- HTTP status code of the response
- User session ID
- Time taken to process the request, in milliseconds
- Time taken to commit the response, in milliseconds
- Current request thread name

Note: These log files are not cleaned up automatically. If you plan to use this feature, it is recommended that you include regular cleanup as part of your business process.

Java Garbage Collection Logs

Timestamped logs for Java garbage collection are created in `<hub_install>/logs`. These log files are not cleaned up automatically. It is recommended that you include regular cleanup as part of your business process.

Using PuTTY to Test Connectivity

Installations of Foundation Hub include the PuTTY terminal emulator, which you can use to test network connectivity of equipment and devices.

Running PuTTY

Access PuTTY here: `<hub_install>\util\putty\win64`.

Accessing PuTTY Documentation

See the PuTTY documentation: https://www.chiark.greenend.org.uk/~sgtatham/putty/docs.html.

Troubleshooting a Label Printer

If you submit a label to be printed and it does not print, perform the following checks to troubleshoot the printer setup:

1. From **Admin and Settings**, click **Equipment**, click the row for the label printer and verify that the **Equipment Type** is set to an **Equipment Class** named *Label Printer*. The class must be set to **Label Printer** to work.
2. Check if the list of **Protocol Parameters** includes a *Printer Name* value. This determines the method of data transfer from the Pipeline Pilot protocol to your printer. Every print job executes a *Control* protocol specified by the **Applications > Application Settings > Foundation Hub > Control Instrument Protocol** setting.

- If *Printer Name* is included as a parameter: Foundation Hub sends data to a Windows queue for an installed printer on the Pipeline Pilot server. Verify that your printer is installed and enabled on your Pipeline Pilot server:
  a. Open the Pipeline Pilot client, and then locate and open the *Control* protocol.
  b. Set the following parameters manually:
     - *Label Name:* Test
     - *Sample Id:* Test
     - *Printer Name:* Value of the *Printer Name* parameter that is set in Foundation Hub
  c. Run the protocol to send a test page to the label printer.
  d. If the test page does not print, verify if the printer installation is functional as stand-alone, external from your BIOVIA system. If so then set the Pipeline Pilot client to debug mode and run the protocol again. The Debug Messages pane will indicate if the protocol finds the label printer. It might tell you that your printer is seen but its driver cannot accept the required type of data ('DocFlavor'). If this is the case then you need to re-create your printer on the Pipeline Pilot server with a text/generic type of driver.
- If *Printer Name* is *not* included as a parameter: The Pipeline Pilot server sends data via FTP directly to a printer located anywhere with network access using the IP Address value in the Equipment list of Protocol Parameters. Remote into the Pipeline Pilot server and ping the label printer to verify that the Pipeline Pilot server has network access to your label printer. After this is successful then send a print job to this printer externally from your local system to verify it is functional as a stand-alone printer.

Appendix B: Troubleshooting the File Crawler and the Parsing Framework

This appendix provides hints to resolve issues that you might encounter when you use the Equipment File Crawler and the Parsing Framework.

No measurement results, equipment work items, or work item discovery events are being generated

Description: None of the pages associated with the File Crawler in Foundation Hub are updating according to the File Crawler's schedule.

Cause and solution: A likely cause is that the File Crawler is not enabled. To enable it, follow the instructions in Enabling the File Crawler.

An instrument was configured for File Crawler discovery, but no entry for it appears in the Equipment Work Items list

Description: You configured an instrument to have its output files discovered by the File Crawler by setting its **Data Discovery Event** to `Crawl`, but no entry for its output file appears in the Equipment Work Items list after the next scheduled run of the File Crawler.

Cause and solution: The File Crawler failed to add the file to Equipment Work Items list. Possible reasons for this are that the **Data Folder** path of the equipment is inaccessible or incorrect, or that more than one file matches the **Data Folder** path and file pattern of the equipment.

To investigate the source of the problem, look in the Work Item Discovery Events page (**Admin and Settings > Equipment > Equipment Work Items > Work Item Discovery Events**) for entries with `Error` in the **Event** column. One of these entries should describe the failed attempt to create a work item.

Execution of the Parse protocol fails because SampleProperty is undefined

Description: The *Parse* protocol generates an error like this:

```
The expression was trying to access the value of a GLOBAL property
'SampleProperty' but it is undefined. One solution is to edit the
PilotScript to test whether the property is defined before using it.
This is done with the expression: @'SampleProperty' IS DEFINED.
```

Cause and solution: The *Parse* protocol must be able to identify a field to use as the sample ID for each record. If no field is configured in the data packet as the sample ID, or if the sample ID field is not present in the instrument data, parsing is aborted.

To rectify this issue:

1. In Foundation Hub, go to **Admin and Settings > Data Packets**.
2. Click the data packet that is used by the equipment type of the instrument whose data could not be processed.
3. Under **Data Fields**, click the field that you want to use as the sample ID (for example, **Air Sample ID**).

If there is no appropriate sample ID field:

a. Click **Edit**
b. Under **Data Fields**, click **Add**.
c. Create the new data field. Ensure that you select **Is Sample Id**.
d. Click **OK**.
e. Click **Save**, and ignore Steps 4 to 6.

4. Click **Edit**.
5. Select **Is Sample Id**.
6. Click **Save**.

Note: If a sample ID field is configured in the data packet, but the error still occurs, verify that the sample ID field is present in files that are output from the instrument. If it is not present, configure a different sample ID field in the data packet.

No measurements are shown in the Measurement page for a measurement

Description: A measurement is reported in the Measurements page, but when you click it to view its details, the parsed result data is not shown. The **Instrument Sample Id** field may be populated, but the only other columns in the table are **Mapped Sample Id** and **Row**, and these are empty.

Cause and solution: The parser subprotocol may not be generating the correct output properties. These properties must be named **parsed_header_data** (containing header data) and **parsed_xy_data_0** (containing table data).

Note that the Parsing Framework performs wildcard searches on some property and file names. Where two or more properties match the search, a conflict can occur, resulting in neither property being read for its data. For example, if a parser subprotocol creates properties **parsed_header_data** and **parsed_header_data_0**, header data might not be properly parsed. In this case, you would modify the parser subprotocol so that only **parsed_header_data** is generated (in addition to **parsed_xy_data_0**).

Appendix C: Foundation Hub Resources Example

The Foundation Hub Component Collection includes a set of Foundation Hub resources that are required to parse the example CSV file. These resources center on an imaginary instrument called **Example Air Analyzer Equipment**.

Importing the Example Resources

To import the example resources into Foundation Hub:

- Using a REST client application, such as Postman, upload the resources into Foundation Hub as described in "Importing Resources" in the *BIOVIA Foundation Hub Administration Guide*. The ZIP file that you need to upload is here:

  ```
  <pps_install>\apps\scitegic\hub\dataroot\data\Equipment\Parsers\
  ExampleAirAnalyzer.zip
  ```

The resources are as shown below. They are shown in their JSON formats from the ZIP file. If they are installed correctly, you can also view them in the Foundation Hub **Admin and Settings** pages.

Reference Ontology Vocabulary

This vocabulary provides simulated generic references for the fields in the `ExampleAirAnalyzer.csv` file. Note that in a real reference ontology vocabulary, the `externalId` values would probably be URIs (that is, strings beginning with `http://`). For more information, see Reference Ontologies.

```
[
  {
    "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3",
    "category": "System",
    "description": "Example Air Analyzer Ontology",
    "name": "Example Air Analyzer Ontology",
    "vocabularyEntries": [
      {
        "externalId": "exampleTemperature",
        "name": "example temperature",
        "vocabulary": {
          "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3"
        }
      },
      {
        "externalId": "exampleHumidity",
        "name": "example humidity",
        "vocabulary": {
          "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3"
        }
      },
      {
        "externalId": "exampleInstrumentSampleId",
        "name": "example instrument sample id",
        "vocabulary": {
           "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3"
        }
```

```
        },
        {
          "externalId": "exampleComment",
          "name": "example comment",
          "vocabulary": {
            "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3"
          }
        },
        {
          "externalId": "exampleDateTime",
          "name": "example date+time",
          "vocabulary": {
            "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3"
          }
        },
        {
          "externalId": "exampleInstrumentOperator",
          "name": "example instrument operator",
          "vocabulary": {
            "id": "29cd9643-d7c5-4499-ac6c-3205b1699fb3"
          }
        }
      ],
    }
]
```

Data Packet and Data Fields

The data packet for the Example Air Analyzer Equipment is shown below, with its data fields included. Points to note are:

- The field names — **Air Humidity**, **Air Temperature**, and so on — are deliberately specific, to distinguish them from fields that might already be configured in Foundation Hub.
- The two header fields, **Instrument Operator** and **Comment**, are members of the `Header` series (`"series": "Header"`).
- The table fields **Air Sample ID**, **Date+Time**, **Air Temperature**, and **Air Humidity** are members of the `Table` series (`"series": "Table"`).
- There is no data field corresponding to the **Not for Display** field in the source file. Therefore, the Parsing Framework will ignore this field, and it will not be included in the parsed output.
- **isSampleId** is set to `True` for the **Air Sample ID** field. This field will be used by the *Parse* protocol to identify records.
- Every data field is associated with a reference ontology.

```
[
  {
    "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
    "description": "Basic air analysis",
    "name": "Example Air Analysis",
    "dataFields": [
      {
        "dataPacket": {
          "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
          "name": "Example Air Analysis"
        },
        "isGroupBy": false,
        "isSampleId": false,
        "name": "Air Humidity",
        "referenceOntology": {
          "externalId": "exampleHumidity",
        },
        "required": false,
        "series": "Table",
        "type": "Float",
        "unit": {
          "id": "131d4d80-baf8-407b-befa-1b143c28176e",
          "displayName": "% (percent)",
          "externalId": "http:\/\/qudt.org\/vocab\/unit#Percent",
          "name": "%",
          "symbol": "%",
          "textName": "PERCENT",
          "unitType": {
            "id": "3440c210-7447-4930-8e3b-5777737f7828",
            "name": "Ratio"
          }
        }
      },
      {
         "dataPacket": {
          "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
          "name": "Example Air Analysis"
        },
        "isGroupBy": false,
        "isSampleId": true,
        "name": "Air Sample ID",
        "referenceOntology": {
          "externalId": "exampleInstrumentSampleId",
         },
        "required": false,
        "series": "Table",
        "type": "Integer",
        "unit": null
      },
      {
        "dataPacket": {
          "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
          "name": "Example Air Analysis"
        },
        "isGroupBy": false,
        "isSampleId": false,
        "name": "Air Temperature",
        "referenceOntology": {
          "externalId": "exampleTemperature",
        },
        "required": false,
        "series": "Table",
        "type": "Float",
        "unit": {
          "id": "8e0fd1ff-1d0a-406e-aa0f-1f8207297b30",
          "displayName": "degC (degrees celsius)",
          "externalId": "http:\/\/qudt.org\/vocab\/unit#DegreeCelsius",
          "name": "degC",
          "symbol": "\u00b0C",
          "textName": "DEGC",
          "unitType": {
            "id": "c1986783-56c2-42a4-ba57-59fabce0b861",
            "name": "Temperature"
          }
        }
      },
      {
        "dataPacket": {
          "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
          "name": "Example Air Analysis"
        },
        "isGroupBy": false,
        "isSampleId": false,
        "name": "Comment",
        "referenceOntology": {
          "externalId": "exampleComment"
        },
        "required": false,
        "series": "Header",
        "type": "String",
        "unit": null
      },
      {
        "dataPacket": {
          "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
          "name": "Example Air Analysis"
        },
        "isGroupBy": false,
        "isSampleId": false,
        "name": "Date+Time",
        "referenceOntology": {
          "externalId": "exampleDateTime"
        },
        "required": false,
        "series": "Table",
        "type": "Date",
        "unit": null
      },
      {
        "dataPacket": {
          "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
          "name": "Example Air Analysis"
        },
        "isGroupBy": false,
        "isSampleId": false,
        "name": "Instrument Operator",
          "referenceOntology": {
            "externalId": "exampleInstrumentOperator"
          },
          "required": false,
          "series": "Header",
          "type": "String",
          "unit": null
        }
      ]
    }
]
```

Equipment Class

The equipment class used for the Example Air Analyzer Equipment is shown below. Note that only its name is supplied. When it is imported into Foundation Hub, it is also given an ID. This ID is specific to your system.

```
[
  {
    "name": "Example Air Analysis Equipment Class"
  }
]
```

Equipment Type

The equipment type of the *Example Air Analyzer Equipment* protocol is `Example Air Analyzer Equipment Type`. Note the following points:

- Its ***connectionType*** is `File`, indicating that it outputs data to a file.
- Its *Parser Path* command parameter specifies the *CSV Parser* subprotocol. This information is visible in Foundation Hub **Admin and Settings > Equipment Type > Equipment Types > Example Air Analyzer Equipment Type > Commands**.

  Because the *Parser Path* is defined at the equipment type level (`"level": "Equipment Type"`), all instances of the equipment type will use the same parser subprotocol. This is the standard configuration when instances of an equipment type are functionally identical.
- The *Data Folder* command parameter is delegated to individual devices (`"level": "Equipment"`). Each device writes its files to a different location.

```
[
  {
    "name": "Example Air Analyzer Equipment Type",
    "description": "Example Air Analyzer Equipment Type",
    "dataPacket": {
      "id": "34484eb5-576f-470e-a771-ecfa5c8f3465",
    },
    "connectionType": "File",
    "equipmentClass": {
      "name": "Example Air Analysis Equipment Class"
    },
    "lifeCycleState": "draft",
    "lifeCycleVersion": "1",
    "manufacturer": "AerTools",
    "modelNumber": "024500",
    "commandDetails": {
      "parameters": [
        {
          "description": "Path to the Pipeline Pilot Component that can
parse the data file.",
          "level": "Equipment Type",
          "name": "Parser Path",
          "protocolType": "Parse",
          "required": true,
          "value": "Protocols\\Database and Application
Integration\\Application
Integration\\Laboratory\\Resources\\Equipment\\Parsers\\CSV Parser"
        },
        {
          "description": "Path to the folder accessible by Pipeline Pilot
that contains output files and includes a file mask",
          "level": "Equipment",
          "name": "Data Folder",
          "protocolType": "Parse",
          "required": false,
          "value": null
        },
        {
          "description": null,
          "level": "API Call",
          "name": "Data File Path",
          "protocolType": "Parse",
          "required": true,
          "value": null
        }
      ]
    }
  }
]
```

Location

Every device must be associated with a location. A location is created for the Example Air Analyzer Equipment, called `Example Location`:

```
[
  {
    "name": "Example Location",
    "description": "Example Location",
    "locationType": {
      "name": "Site"
    },
    "path": "Example Location"
  }
]
```

Equipment

The example equipment details are shown below. Observe the following points:

- The equipment is an instance of the Example Air Analyzer Equipment Type. The Parsing Framework uses the equipment type to determine which parser subprotocol to use.
- The instrument's *Location* is `Example Location`.
- The *Data Folder* parameter specifies the location of files generated by the equipment. The `*.csv` filter tells the Parsing Framework to ignore all files except for CSV files.

  Note that if the Parsing Framework finds more than one file matching the Data Folder path and file pattern, it will generate an error, and not parse any files in the folder, because it cannot determine which file is the correct one.
- The *dataDiscoveryMethod* setting has its *id* set to `d48cac7e-de4d-4a8c-be34-9e381778b297`. This is the ID of the vocabulary entry for `Crawl`. This setting causes the File Crawler to find the equipment's files when it scans data folders.

  Note: The ID of the `Crawl` vocabulary entry is always `d48cac7e-de4d-4a8c-be34-9e381778b297`. Whenever you define equipment in JSON format, use this ID if you want its output files to be found by the File Crawler.
- The *Data Folder* has a *protocolType* of `Parse`.

```
[
  {
    "id": "56341974-956b-4298-b9e6-82a46077fde9",
    "barcode": "5087",
    "serialNumber": "5087",
    "description": "Example Air Analyzer Equipment",
    "equipmentType": {
      "name": "Example Air Analyzer Equipment Type",
    },
    "location": {
      "path": "Example Location"
    },
    "dataDiscoveryMethod": {
      "id": "d48cac7e-de4d-4a8c-be34-9e381778b297"
    },
    "connectionDetails": {
      "parameters": [
        {
          "name": "Data Folder",
          "value": "$(/pilot_
data/scitegic/hub)/data/Equipment/Parsers/sampleParserFiles/*.csv",
          "protocolType": "Parse"
        }
      ]
    }
  }
]
```

Appendix D: Direct Connect Equipment Reference

- BIOVIA Equipment Library
- BIOVIA Standard Pin Configuration for Equipment Integration

BIOVIA Equipment Library

BIOVIA provides an equipment library of definitions for commonly-used lab equipment and instruments that are connected to a TCP/IP network, typically through RS232 serial connections and a protocol converter. The equipment class, equipment type, and a configured parsing method are all pre-configured per instrument, so you can import the entire library all at once, or delete any unused equipment definitions from the file first before you import it in order to keep your equipment more manageable. See Importing an Equipment Type Library.

Class	Equipment Type
Balance	AND HR-200
Balance	Mettler AG104
Balance	Mettler AG135
Balance	Mettler AG204
Balance	Mettler AG245
Balance	Mettler AT200
Balance	Mettler AT201
Balance	Mettler AT250
Balance	Mettler AT261
Balance	Mettler AX105
Balance	Mettler AX105DR
Balance	Mettler AX205
Balance	Mettler AX205
Balance	Mettler AX26
Balance	Mettler AX304
Balance	Mettler AX304
Balance	Mettler MT5
Balance	Mettler MX5
Balance	Mettler PB1502-S
Balance	Mettler PB3001-S

Class	Equipment Type
Balance	Mettler PB3002
Balance	Mettler PB3002-S
Balance	Mettler PB302
Balance	Mettler PB403-S
Balance	Mettler PB602-S
Balance	Mettler PG1003-S
Balance	Mettler PG2002-S
Balance	Mettler PG3001-S
Balance	Mettler PG4002-S
Balance	Mettler PG403-S
Balance	Mettler PG6002-S
Balance	Mettler PM100
Balance	Mettler PM1200
Balance	Mettler PM2000
Balance	Mettler PM2500
Balance	Mettler PR2002
Balance	Mettler PR2003
Balance	Mettler PR5002
Balance	Mettler PR503
Balance	Mettler PR8002
Balance	Mettler PR803
Balance	Mettler SB16001
Balance	Mettler SR16001
Balance	Mettler UMX2
Balance	Mettler WXSS205DU
Balance	Mettler XP1203-S
Balance	Mettler XP1203-S
Balance	Mettler XP204
Balance	Mettler XP205
Balance	Mettler XP205

Class	Equipment Type
Balance	Mettler XP205DR
Balance	Mettler XP26
Balance	Mettler XP26
Balance	Mettler XP56
Balance	Mettler XS1003-S
Balance	Mettler XS104
Balance	Mettler XS204
Balance	Mettler XS205
Balance	Mettler XS205DU
Balance	Mettler XS3DU
Balance	Mettler XS603-S
Balance	Mettler XS802-S
Balance	Ohaus EP-413C
Balance	OHAUS EP612C
Balance	OHAUS Explorer
Balance	Ohaus V12140
Balance	Sartorius A200
Balance	Sartorius A200-S
Balance	Sartorius AC210S
Balance	Sartorius AC211S
Balance	Sartorius BP210D
Balance	Sartorius BP211D
Balance	Sartorius BP221S
Balance	Sartorius BP3100S
Balance	Sartorius BP4100S
Balance	Sartorius CP124S
Balance	Sartorius CP2201
Balance	Sartorius CP2202S
Balance	Sartorius CP225D
Balance	Sartorius CP2P-F

Class	Equipment Type
Balance	Sartorius ED6202 S
Balance	Sartorius ED6202S
Balance	Sartorius Genius
Balance	Sartorius L 420 S
Balance	Sartorius LA2200S
Balance	Sartorius LA230S
Balance	Sartorius LA310S
Balance	Sartorius LC12001S
Balance	Sartorius LC2201
Balance	Sartorius LC220S
Balance	Sartorius LC4200S
Balance	Sartorius LE2202S
Balance	Sartorius LE225D
Balance	Sartorius LE26P
Balance	Sartorius LE4202S
Balance	Sartorius LP1200
Balance	Sartorius LP2200S
Balance	Sartorius LP3200D
Balance	Sartorius LP4200S
Balance	Sartorius LP5200P
Balance	Sartorius LP620S
Balance	Sartorius M-power
Balance	Sartorius M5P
Balance	Sartorius MC210P
Balance	Sartorius MC210S
Balance	Sartorius MC5
Balance	Sartorius ME215P
Balance	Sartorius ME215S
Balance	Sartorius ME235
Balance	Sartorius ME235S

Appendix D: Direct Connect Equipment Reference

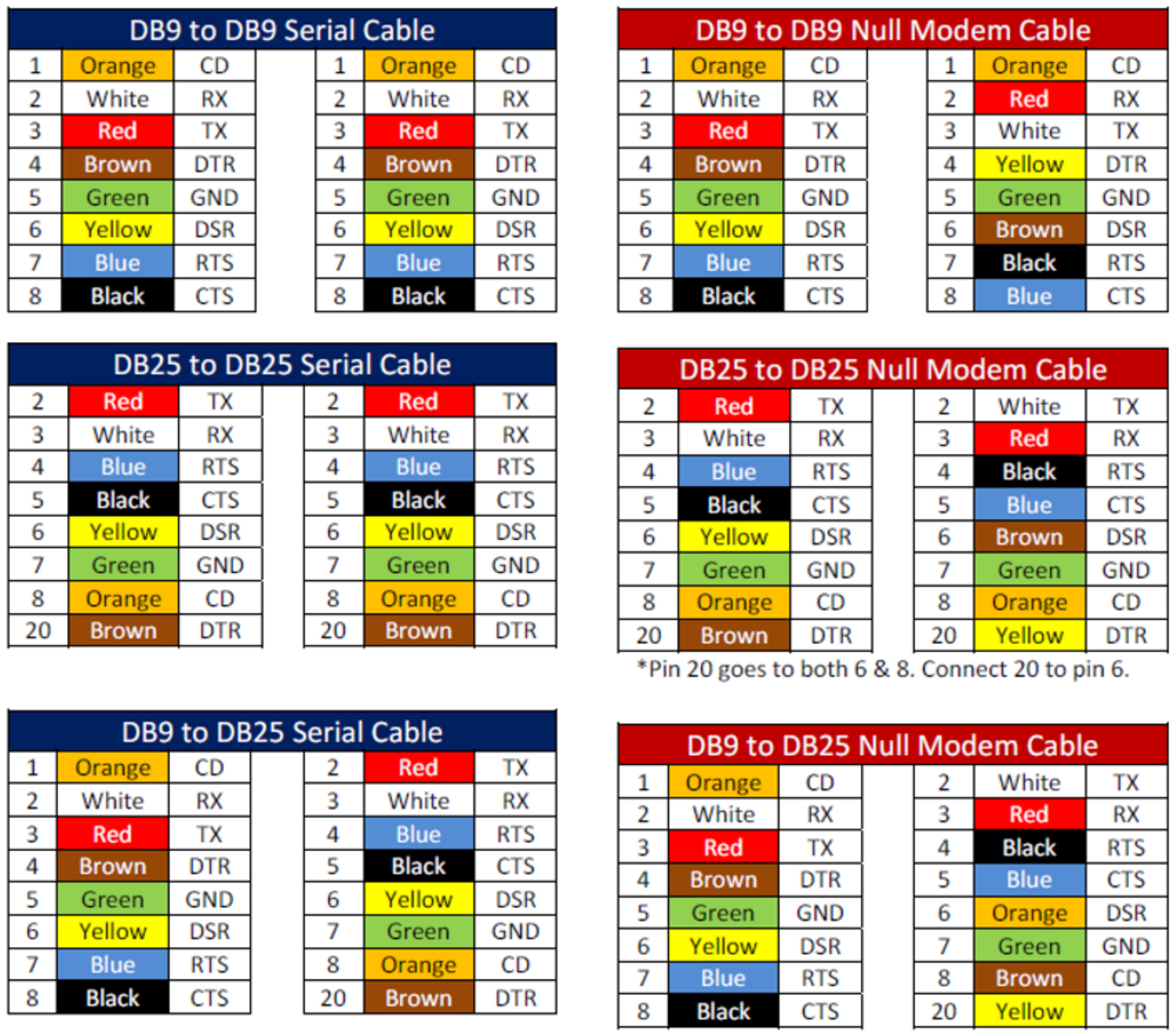
Class	Equipment Type
Balance	Sartorius ME36S
Balance	Sartorius ME5
Balance	Sartorius MSA6.6S
Balance	Sartorius R160P
Balance	Sartorius R200-D
Balance	Sartorius SE2
Balance	Sartorius TE612
Balance	Shimadzu AUW1200
Calorimeter	Zeiss MCS311
Conductivity Meter	Mettler MPC227
Conductivity Meter	Mettler SevenCompact
Conductivity Meter	Orion 150
Conductivity Meter	Orion 150A
Conductivity Meter	Orion 162A
Conductivity Meter	Radiometer CDM230
Conductivity Meter	Thermo Orion 150 A+
Conductivity Meter	YSI 3200
Conductometer	Metrohm 712
Coulometer	Metrohm 737 KF Coulometer
Coulometer	Metrohm 756 KF Coulometer
Densitometer	Mettler DE40
Densitometer	Paar DMA48
Density Meter	Anton Paar DMA 4500
Density Meter	Anton Paar/Citizen DMA 5000
Dissolution Bath	Distek 2100B
Dissolution Bath	Hanson Research SR8 Plus
Dissolution Bath	VanKel 25-1000
Dissolution Bath	Vankel VK7000
DissoPrep	GTB Dissoprep MX8
Flow Meter	TSI 4043

Class	Equipment Type
Fluorometer	Turner Quantech FM109535
Hardness Tester	Dr. Schleuniger 8M
Hardness Tester	Holland C50
Hardness Tester	Kraemer Elektronik HC 97
Hardness Tester	Sotax HT1
Hardness Tester	Vankel VK 200
Karl Fischer	Cosa Instruments CAVA-100
Karl Fischer	Metrohm 831 Coulometric
Karl Fischer	Mettler DL38
Melting Point Apparatus	Buchi B-540
Melting Point Apparatus	Stanford Research OptiMelt
Moisture Analyzer	Mettler HB43
Moisture Analyzer	Mitsubishi CA-100
Moisture Analyzer	Photovolt Aquatest - 10
Moisture Analyzer	Photovolt Aquatest-2010
Moisture Analyzer	Sartorius MA40
Moisture Analyzer	Sartorius MA51
Osmometer	Advanced Instruments 3250
Osmometer	Advanced Instruments 3300
Osmometer	Advanced Instruments 3D3
Osmometer	Fiske 210
Osmometer	Wescor 5520
Particle Counter	Hach Ultra Met One 3400
pH Meter	Accumet AR15
pH Meter	Accumet AR20
pH Meter	Beckman 360
pH Meter	Beckman 45 pH Meter
pH Meter	Brinkmann Metrohm 713
pH Meter	Corning 350
pH Meter	Corning 540

Class	Equipment Type
pH Meter	Fisher Scientific AR25
pH Meter	Jenway 3020
pH Meter	Jenway 3320
pH Meter	Metrohm 780
pH Meter	Mettler SevenCompact
pH Meter	Mettler SevenEasy
pH Meter	Mettler SevenMulti
pH Meter	Orion 250A
pH Meter	Orion 350
pH Meter	Orion 370
pH Meter	Orion 420A
pH Meter	Orion 420A
pH Meter	Orion 420A
pH Meter	Orion 720A
pH Meter	Orion 920A
pH Meter	Radiometer PHM210
pH Meter	Radiometer PHM220
pH Meter	Radiometer PHM93
pH Meter	Thermo Orion 3 Star
pH Meter	Thermo Orion 370
pH Meter	Thermo Orion 4 Star
pH Meter	Thermo Orion 420A+
pH Meter	Thermo Orion 520A+
pH Meter	Thermo Orion 720A Plus
pH Meter	VWR sympHony SB70P
pH Meter	VWR sympHony SB80PC
pH Meter	VWR sympHony SP21
pH Meter	VWR sympHony SP70P
pH Meter	VWR sympHony SR301
pH Meter	VWR sympHony SR30I

Class	Equipment Type
pH Meter	VWR sympHony SR40C
Polarimeter	Perkin Elmer 341
Refractometer	Leica Auto ABBE
Refractometer	Mettler RE40
Thermometer	Fluke 1523
Thermostatic Bath	Julabo F26MV
Titrator	Brinkmann 756 KF Coulometer
Titrator	Metrohm 702SM Titrino
Titrator	Metrohm 736 GP Titrino
Titrator	Metrohm 787 KF Titrino
Titrator	Metrohm 795 Titrino
Titrator	Metrohm 831 Coulometric
Titrator	Metrohm 841 KF Titrando
Titrator	Metrohm 841 Titrando
Titrator	Mettler DL31
Titrator	Mettler DL36
Titrator	Mitsubishi CA-200
Titrator	Mitsubishi CAVA-100
Turbidimeter	Hach 2100AN
Vacuum Pump	Millipore Milliflex PLUS
Viscometer	Brookfield DVIII-LV
Viscometer	Brookfield DVIII-RV
Waterbath	Julabo MD

BIOVIA Standard Pin Configuration for Equipment Integration



DB9 to DB9 Serial Cable

1	Orange	CD	1	Orange	CD
2	White	RX	2	White	RX
3	Red	TX	3	Red	TX
4	Brown	DTR	4	Brown	DTR
5	Green	GND	5	Green	GND
6	Yellow	DSR	6	Yellow	DSR
7	Blue	RTS	7	Blue	RTS
8	Black	CTS	8	Black	CTS

DB9 to DB9 Null Modem Cable

1	Orange	CD	1	Orange	CD
2	White	RX	2	Red	RX
3	Red	TX	3	White	TX
4	Brown	DTR	4	Yellow	DTR
5	Green	GND	5	Green	GND
6	Yellow	DSR	6	Brown	DSR
7	Blue	RTS	7	Black	RTS
8	Black	CTS	8	Blue	CTS

DB25 to DB25 Serial Cable

2	Red	TX	2	Red	TX
3	White	RX	3	White	RX
4	Blue	RTS	4	Blue	RTS
5	Black	CTS	5	Black	CTS
6	Yellow	DSR	6	Yellow	DSR
7	Green	GND	7	Green	GND
8	Orange	CD	8	Orange	CD
20	Brown	DTR	20	Brown	DTR

DB25 to DB25 Null Modem Cable

2	Red	TX	2	White	TX
3	White	RX	3	Red	RX
4	Blue	RTS	4	Black	RTS
5	Black	CTS	5	Blue	CTS
6	Yellow	DSR	6	Brown	DSR
7	Green	GND	7	Green	GND
8	Orange	CD	8	Orange	CD
20	Brown	DTR	20	Yellow	DTR

*Pin 20 goes to both 6 & 8. Connect 20 to pin 6.

DB9 to DB25 Serial Cable

1	Orange	CD	2	Red	TX
2	White	RX	3	White	RX
3	Red	TX	4	Blue	RTS
4	Brown	DTR	5	Black	CTS
5	Green	GND	6	Yellow	DSR
6	Yellow	DSR	7	Green	GND
7	Blue	RTS	8	Orange	CD
8	Black	CTS	20	Brown	DTR

DB9 to DB25 Null Modem Cable

1	Orange	CD	2	White	TX
2	White	RX	3	Red	RX
3	Red	TX	4	Black	RTS
4	Brown	DTR	5	Blue	CTS
5	Green	GND	6	Orange	DSR
6	Yellow	DSR	7	Green	GND
7	Blue	RTS	8	Brown	CD
8	Black	CTS	20	Yellow	DTR

Dassault Systèmes Support Resources

For additional resources or to contact Dassault Systèmes Customer Support, visit the Support portal:

https://www.3ds.com/support/

From this portal, you can:

- Call or email Dassault Systèmes Customer Support
- Submit a request
- Download installers
- Access hardware and software requirements
- Access Knowledge Base
- Access Communities and Twitter feeds