ADVANTEST®

ADVANTEST CORPORATION

ACS Real-Time Data Infrastructure (RTDI) User Guide

v2.3.0

This version includes:

ACS Nexus v3.0.0
ACS Unified Server v2.2.0
ACS Edge Server v3.3.0
ACS Container Hub v3.0.0

v2.3.0, July 2025
Printed in U.S.A

Legal Notices

Trademarks and Registered Trademarks

- ADVANTEST is a trademark of Advantest Corporation.
- All other marks referenced herein are trademarks or registered trademarks of their respective owners.

Revision History

Rev.	Date	Notes
v1.0.0	August 2023	First Release
v1.1.0	October 2023	Added 3 new advanced control items to section 2.2.2 "Advanced Control"
		Added the following sections: • 2.2.2.2 Variable Control • 2.2.2.3 Test Item Control • 2.2.2.4 Activity Control • 2.5.2 Auto Deploy Mode • 3.15 Update Client Credentials • 3.17 Create Application Descriptor • 3.18 Delete Application Descriptor • Appendix 2. Python Logger, Result, and Event
		Added new items to Table 2-3, "Basic and Advanced Control Commands"
		Updated section 2.3.3.5 "class oneapi::Command" to include new advanced control details.
		Updated section 3.14 "Create Client Credentials" to add expiry date feature.
		Updated the description for setRegHost().
		Appended description of volume_attach option for ContainerConfig class
v1.1.1	October 2023	Updated section 2.1 with additional descriptive details, as follows: • Acronym descriptions for the real time test cell data types • Description for EDL • Reference to Advantest TDC for more info on Application Model and Recipe file • Updated userdefined.ini content example
		Updated Appendix 2, "Python Logger, Result, and Event"
		Added Appendix 3, "Application Descriptor Command Line Tool"
v1.2.0 RC	December 2023	Added section 3.1 "ACS Container Hub Registration"
		Added section 3.2 "Connecting ACS Container Hub to an ACS Unified Server"
		Added section 3.18 "Update Application Descriptor"
		Added example code for all the functions in section 6.2.1 for the containerconfig public functions. Also added new setOption function.

v1.2.0	February 2024	Added sections 4.1, 4.2, 4.3, 5.1, 5.2, and 5.3
		Updated steps 6, 7, and 8 for "Using Variable Control in SmarTest 8"
		Added a SetNewBin setup example (search for "SetNewBin Example")
		Added section 3.3.17, "Reset Client Credential"
		Added section 4.2.4, "Redundancy"
		Added section 5.1.3, "Dynamic Certificates for MTLS"
		Added section 8, "ACS RTDI Troubleshooting"
v1.3.0	April 2024	Updated the OneAPI C++ SDK package contents in section 2.3.1, "General Information" and OneAPI Python SDK package contents in section 2.4.1.
		Added MEASURED_SCAN data type and updated DEVICE data type in Table 2-4 and Table 2-12, "OneAPI Data Types."
		Added new data types to section 2.3.3.3, "enum oneapi::DataType" and section 2.4.3.3, "enum DataType"
		Modified the following ACS Nexus data type enums: • DATA_TYP_MEASURED_FUNCTIONAL • DATA_TYP_MEASURED_MULTI_PARAM
		Added the following ACS Nexus data type enums: • DATA_TYP_MEASURED_SCAN • DATA_TYP_DEVICE_PIN • DATA_TYP_DEVICE_PATTERN (DATA_TYP_PIN_DATA is replaced by DATA_TYP_DEVICE_PIN)
		Added Python 3.10 and 3.11 support for Nexus Python SDK

		Added the following Nexus functions:
		get_PatternCount()
		get_PinInfoCount()
		query_OpSequence()
		query_PatternLables()
		query_TotalCycleCount()
		query_FailCycleCount()
		query_PatternResults()
		query_PatternName()
		query_PinResults()
		query_PinName()
		query_FailCycles()
		get_PatternCount()
		query_PatternLabels()
		get_PinInfoCount()
		query_ChannelType()
		query_ChannelName()
		query_PhysicalName()
		query_LogicalName()
		query_HeadNumber()
		query_SiteNumber()

		Extensive updates to section 2.5.3, "Non-Auto Deploy Mode"
		Modified section 2.5.4, "Container Configuration File"

v1.3.0	April 2024	Added section 2.5.5 and section 2.6.3, "Workflow"
		Added Certificate Expiration Reminder to Table 2-19, "System Status Indicators"
		Added section 2.10, "FAST-API Support on ACS Edge"
		Added section 2.11, "ACS Edge Redundancy Support"
		Updated descriptions for Container attributes in section 3.3.18, "Create Application Descriptor"
		Added section 5.1.4, "Multi-Tester Application Support
		Modified description for *publish-ports* and *environment* options in section 6.2, "Container Config"
		Modified Step 1 example data in section 6.5, "Container Deployment Example"
v1.4.0	July 2024	Added the following sections: • 2.3.3.7 class oneapi::QueryResponse • 2.3.3.8 class oneapi::DFFData • 2.4.3.7 class QueryResponse • 2.4.3.8 class DFFData • 2.12 Bi-Directional Communication between TP and RTDI Application • 2.13 Data Feed Forward • 5.1.5 Data Feed Forward • 5.4.1.2 Container Hub Method
		Updated section 2.3.1, including Table 2-2
		Modified the following under section 2.3.3 C++ API and under section 2.4.3 Python API: • connect() • getConnectionState()
		Modified section 2.3.4, including Table 2-9
		Removed "conf" from package contents in section 2.4.1
		Added the following under section 2.4.3 Python API: • consumeTPSend() • consumeTPRequest()
		Modified section 2.4.4, including Table 2-14
		Added edge.containers.volume_attachments to Table 2-17 "images.json Content Descriptions"
		Added Container Status Information and License Status to Table 2-19 "System Status Indicators"

v2.0.0	October 2024	Added measurement data to the data type list in section 2.1 and to Table 2-4 “OneAPI Data Types”
		Added measurement data functions to the NexusData class

		Added the following ACS Nexus functions:
		query_TestSuite()
		query_MeasurementName()
		get_MeasurementName()
		get_GroupCount()
		query_GroupName()
		query_GroupBypassed()
		query_GroupSites()
		get_SequenceCount()
		query_SequenceName()
		query_SequenceBypassed
		query_SequenceSites()
		get_SequenceGroupResults()
		query_SequenceGroupName()
		query_SequenceGroupBypassed()
		query_SequenceGroupSites()

		Replaced “Test Floor Server” with “ACS Unified Server” in Table 2-19 “System Status Indicators”
		Modified Container Hub sections on creating, updating, and deleting Application Descriptors.
		Added the following sections: **ACS Container Hub** 3.3.21 "Support for 1:N Applications" 3.3.22 "Viewing 1:N Application Status" **ACS Edge Server** 4.2.5 "Remote Service Upgrade" **ACS Unified Server** 5.4.3 "1:N Application Descriptors" 5.6 "File Synchronization" 5.7 "Memory Store SDK"
		Revised section 8.2 "ACS Unified Server Troubleshooting"
v2.1.0	December 2024	Added the following sections: **ACS Nexus** 2.4.3.8 "class DFFData" 2.12.1.2 "NexusTPI for SMT8" 2.12.1.3 "NexusTPI for SMT7" 2.13.1 "Data Writing – Nexus Data" 2.13.2 "Data Writing – Customer Data from the Application" 2.13.3 "Data Reading – By Application" 2.13.4 "Data Writing – Customer Data from Test Program" 2.13.5 "Data Reading – by Test Program" 2.14 "Configurable Nexus Services" **ACS Unified Server** 5.9 "Data Feed Forward – Flex" **Appendix 3. "Azure Hosted Container Registry"**
		New ACS Nexus function: createRawQueryRequest()

		Updated values in Table 2-9 and Table 2-14 "OneAPI Environment Variable Description"
		Added jsonschema to Python SDK software
		Revised all of section 2.4.3.9 "Class DFF"
v2.2.0	March 2025	Added -3 (unknown) and -4 (not supported) as return values for Nexus API `connect() and sendCommand()` (C++ and Python)
		Added "init" interface to Nexus TPI for SMT 8 (Table 2-21) and SMT7 (Table 2-23)
		Added the following sections: 2.3.3.9 class oneapi::TestCell 2.4.3.10 class TestCell 2.15 Nexus License Check Modes 3.3.21 Multi-User Permissions for Managing App Descriptors or Organization Project 3.3.22 Using Multi-User Permissions 5.9.4 DFF UI
		Added instruction on accessing the Unified Server license server in section 5.2 Licensing
		New Data Streaming field (STREAM_ENABLED) added to the Application Descriptor `unified` field in section 5.5.3 1:N Application Descriptors
		A new `restart policy` option is added to ContainerConfig class in section 6.2 ContainerConfig
		`setOption()` function is updated to include the new restart policy in section 6.2.1 Public Function
v2.3.0	June 2025	Added the following sections: 3.3.25 Viewing, Creating, and Editing 1:N Application Descriptors 3.3.26 Test Applications for 1:N Application Descriptors 3.3.27 Monitoring – Alert Subscriptions 3.3.28 Replication 5.6 Unified Server Application Testing 5.8 Redis Memory Store SDK for Cross-Cluster Replication
		Added a NOTE regarding Advanced Control support for ACS Nexus v3.0.0 to section 2.2.2 Advanced Control
		New functions added to section 2.3.3.2 class oneapi::Monitor
		Brokers option added to TestFloor_Server in Table 2-13. acs_nexus.ini Content Descriptions
		"Data Structure in JSON Format" and corresponding tables are added to 2.13.3 Data Reading – By Application
		Upgrade procedure modified in section 4.2.5 Remote Service Upgrade

Table of Contents

1. Introduction

The ACS Real-time Data Infrastructure is a comprehensive high-value platform that collectively delivers improved yield, quality, and time to market. At its core, the platform acts as a communication backplane for the test floor, facilitating rapid and accurate information exchange. With support for online edge computing/analytics, the platform enables real-time adaptive decision making within and between touchdown operations, allowing for optimized performance.

The ACS RTDI platform consists of the following products:

ACS Nexus
A software solution integrated into the host workstation to provide a standard interface for real time test cell data streaming and inline equipment control between equipment and external clients across Advantest platforms.

ACS Container Hub
The artifact repository and distribution hub for Advantest cloud workloads, used for storage and distribution of container images.

ACS Edge Server
A high-performance edge compute and analytics server that, when integrated into a test cell, enables ultra-fast algorithmic (AI, machine learning, and statistical) decision making during test execution.

ACS Unified Server
A multi-purpose server that supports application service, database storage, and true Zero Trust Security for the test floor.

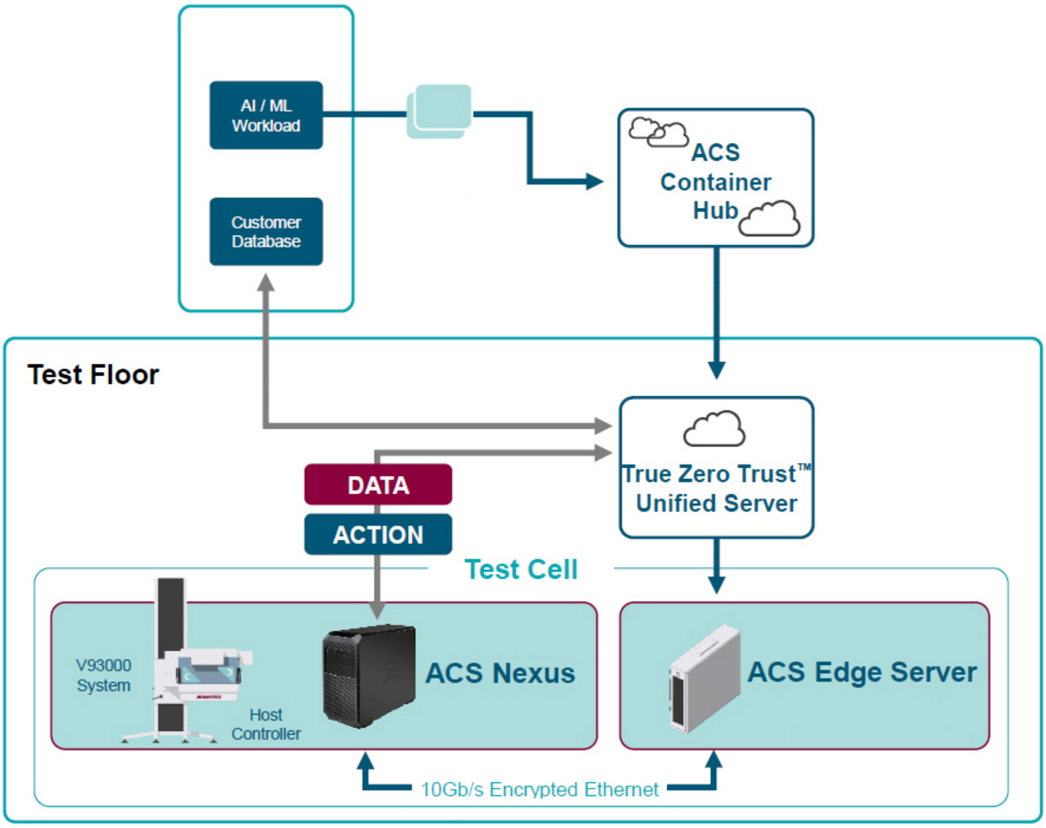


Figure 1-1. ACS RTDI Platform Overview

2. ACS Nexus

The ACS Nexus is the software solution that provides a standard interface for real-time test cell data streaming and inline equipment control between equipment and external clients across all Advantest platforms.

NOTE 1: ACS Nexus automatically starts and runs perpetually on the Host Workstation after installation. If an ACS Nexus restart is needed, see Restarting ACS Nexus.

NOTE 2: Only one user at a time can run ACS Nexus on the same Host Workstation.

2.1 Collecting Real Time Test Cell Data

Comprehensive real time test cell data collection enables applications to do data analysis in real time. ACS Nexus provides real time data to the ACS Edge Server per test item level. Real time data is provided through OneAPI after test start of the first device in a lot.

NOTE: For SmarTest 7, there are multiple ways to start production. Nexus support for production start is as follows:

Supported: Application Model File

Unsupported: TCCT

Not Verified: Workorder and BLC client

Real time data includes the following:

Measured Value Data	Parametric test data Multiple parametric test data Functional test data Scan test data
Production Event Data	Lot Start Data – Master Information Record (MIR) / Site Description Record (SDR) Wafer Start Data – Wafer Information Record (WIR) / Wafer Configuration Record (WCR) Test Start Data – Part Information Record (PIR) Test Flow Start Data Test Flow End Data Test End Data – Part Results Record (PRR) Wafer End Data – Wafer Result Record (WRR) Lot End Data – Master Results Record (MRR)
Device Data	Device Configuration – Basic information / Pin configuration file name / Channel attribute file name Bin list – both soft bin and hard bin Pin data Pattern data
User Defined Data	Only support user defined variables in Recipe file (SmarTest 8) or Application Model file (SmarTest 7) Only support user defined variables outside the device level in SmarTest 8 or SmarTest 7 Max support variable count is 100 Can be configurable to select needed data
Measurement Data	Parallel group (only applies to SMT version 8.7.0 and greater (refer to Parallel Group Data) Operating sequence call (only defined for SmartBurst)

To automatically collect Measured Value data, Production Event data, and Device data, the only requirement is to ensure that EDL (Event Data Logging - the default real time data stream provided by 93K Smartest) is fully enabled during production. Collecting User Defined data requires setting up variables. Follow the steps below to setup variables for collecting User Defined data.

1. Set variables in the Recipe (SmarTest 8) or in the Application Model file (SmarTest 7). For more information on the Recipe file and Application Model file, refer to the Advantest Technical Documentation Center (TDC).

   NOTE: In the user defined data examples below, on the left side is the variable name which ACS Nexus provided to the application. On the right side is the variable name defined by the user.

Example Recipe File for SmarTest 8:

```
<TestProgram action="ACTIVATE" name="inco4_v255_online/Inco4.prog" />
<TestProgram action="LOAD" />
<Assignment><Set name="ROOT_PATH" value="/home/Elijah/smt/inco4_v255_online" />
</Assignment>
```

User Defined Data

```
<Assignment>
   <Set name="DATALOG_RATE" value = "10" />
   <Set name="DIFFUSION_CENTER" value = "10" />
   <Set name="PROCESS_CODE" value = "10" />
   <Set name="LOT_SIZE" value = "10" />
</Assignment>
```

```
<!-- Create and connect PH session with specified driver -->
<EquipmentControl functions="START_PHCONTROL">
   <!-- Currently, the first input must GENERIC_93K_DRIVER -->
   <In>GENERIC_93_DRIVER</In>
   <In>{$ROOT_PATH}/GenericProber/Tel/</In>
   <In>{$ROOT_PATH}/GenericProber/Tel/config/P12-GPIB-Prober-4-die-256-bin.cfg</In>
   <!-- the wafer description file can be specified in the fourth parameter -->
   <In>{$ROOT_PATH}/wafer_15x15</In>
</EquipmentControl>

<!--Active, Load, Bind and Start test program -->
<TestProgram action="RUN" name="inco4_v255_online/Inco4.prog" lotType="WAFER_TEST/>

<!-- To setup some detail lot information in a sub-recipe file -->
<SubRecipe path="sub.xml" />
```

Example Application Model File for SmarTest 7:

```
{ WAFER [(wafermap),({wafer_id}] wafer:

       ACTIONS
               dummy = PROB_HND_CALL (get wafer);
                   -- * wafer_id      = ELEMENT_OF_PARAM_LIST ({cassette_list}) / REGEXP (int);
                   * wid            = AUTOINCREMENT;
                   * wafer_id       = CONST_INPUT({Lot_id}.{wid});
                STDF_FILE = CONST_INPUT(/tmp/AA/aa_{Lot_id}_{wafer_id}_{Start_time}.stdf);
```

User Defined Data

```
       * DATALOG_RATE          = CONST_INPUT (vws01);
       * DIFFUSION_CENTER      = CONST_INPUT (vws02);
       * PROCESS_CODE          = CONST_INPUT (vws01);
       * LOT_SIZE              = CONST_INPUT (vws02);
```

```
   LOG_SPECIFICATION
               RESULT_LOG      = ON;
               NEW_RESULT_FILE = ON; (/tmp/{Lot_id}.wfrSum{wafer_id}.gdf);
               POST_PROCESSING = D_BRIDGE_CALL (send_to_dbase /tmp/(Lot_id}.wfrSum{wafer_id}.gdf);

   { DEVICE die:
      ACTIONS
                           -- SET_TEMPERATURE({Temperature});
                    PROB_HND_CALL (get_die);
```

2. Configure the User Defined data to collect in the Nexus configuration file. The path for the user defined data is `/opt/acs/nexus/conf/collect/userdefined.ini`.

userdefined.ini Content Example:

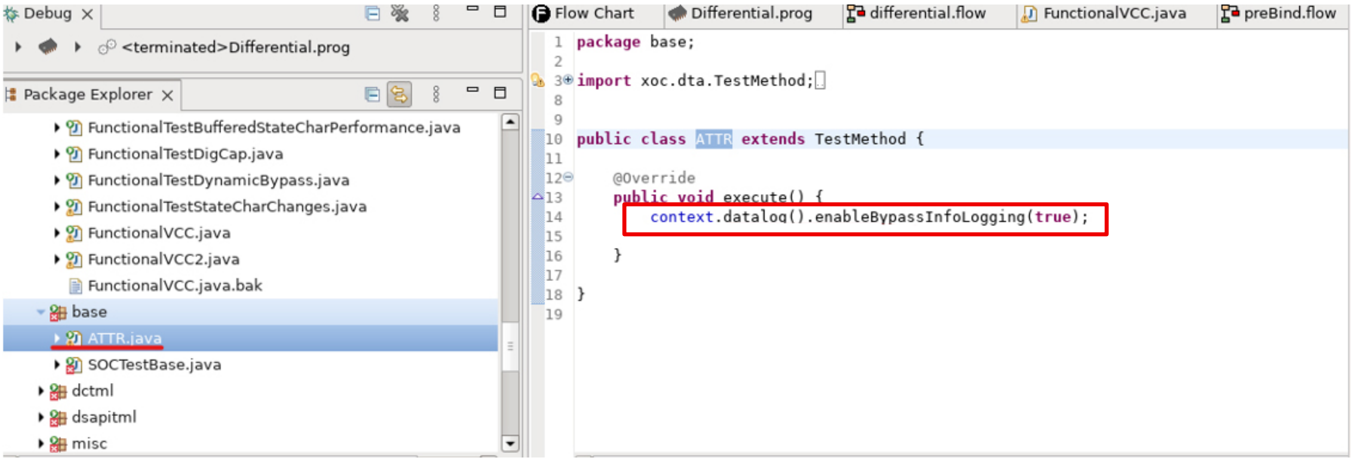
```
[SMT7]
datalog_rate = DATALOG_RATE
diffusion_center = DIFFUSION_CENTER
process_code = PROCESS_CODE
lot_size = LOT_SIZE

[SMT8]
datalog_rate = DATALOG_RATE
diffusion_center = DIFFUSION_CENTER
process_code = PROCESS_CODE
lot_size = LOT_SIZE
##############################################
```

3. Use OneAPI to collect the User Defined data from ACS Nexus. For C++ interface, refer to page 40 and 46. For Python interface, refer to page 112 and 120.

Parallel Group Data (additional information)

Ensure bypass logging is enabled in the test program:

```
context.datalog().enableBypassInfoLogging(true);
```



```
package base;

import xoc.dta.TestMethod;


public class ATTR extends TestMethod {

    @Override
    public void execute() {
        context.datalog().enableBypassInfoLogging(true);

    }

}

```

2.2 Controlling the Test Cell

ACS Nexus provides basic and advanced actions during production, enabling real time controls from the client software (the user-developed application based on OneAPI).

2.2.1 Basic Control

NOTE: To enable basic control, production should be executed by Recipe (SmarTest 8) or by Application Model (SmarTest 7). Refer to the Advantest Technical Documentation Center (TDC) for additional information.

The basic control actions include Pause and Stop. Both these actions can be sent at any time, but they can only be executed after test flow execution and device binning.

- PAUSE

  Executes a real-time pause during wafer testing or final testing when production issues are identified by an organization's applications. Follow-up actions after the pause are handled by the organization (for example, automatic action by other tools or manual action by an operator).

- STOP

  Executes a real-time stop of wafer testing or final testing when production issues are identified by an organization's applications

2.2.2 Advanced Control

NOTE: ACS Nexus v3.0.0 does not support Advanced Control features on the V93K RHEL 9 platform.

The following Advanced Control features provide additional control within the test cell environment:

- Bin Control
- Site Activity Control

These Advanced Control functions are supported in SmarTest 7 and SmarTest 8 as indicated in the table below.

	SmarTest 7		SmarTest 8	
	Final Test	Wafer Sort	Final Test	Wafer Sort
BinControl	✓	✓	✓	✓
SiteActivityControl	✓	-	✓	-

2.2.2.1 Bin Control

Bin Control is an advanced control action which overrides binning information to be sent to a prober or handler with the altered bin, which is directed by binning controller applications such as DPAT or other outlier detection applications. All binning operations are logged into a dedicated log file for future reference. The log file follows the format indicated below.

NOTE: The default path for the below log file is `/binning_hist_files/ binning_hist_<YYYYmmdd_HHMMSS.csv`.

```
Line format: <TIMESTAMP> "," <DEVICE_ID> "," <ORIG_BIN> "," <ALT_BIN>
      <TIMESTAMP>:      <YYYYmmddHHMMSS>
      <DEVICE_ID> :     <DEVICE_ID_WS> || <DEVICE_ID_FT>
      <DEVICE_ID_WS> :  "W" <X> "," <Y>
      <DEVICE_ID_FT> :  "P" <INDEX_COUNT> "." <SITE>
```

The Bin Control feature is available only for using a prober/handler driver which is dedicated for ACS Nexus. Other restrictions include:

- The EDL stream will not be changed by this feature.
- BIN Control request is valid after testflow start.
- BIN ID "-1" cannot be set through BinControl.

For OneAPI Bin Control details, see C++ class Command or Python class Command.

2.2.2.1.1 Test Program Configuration (Bin Control)

This feature works with the equipment driver (located in /opt/acs/nexus/ph_libs). The equipment driver should be installed before using this feature. For instructions on how to install this driver, refer to section 2.4 of the *ACS RTDI Installation Guide*.

To use this feature:

1. Specify the drivers in a test program if using “GenericHandler.”
2. Check that the driver objects (which will be used in production) are located in /opt/acs/nexus/ph_libs/
3. Modify the description for the line in "prod" as follows: **Hp_lib:** /opt/acs/nexus/ph_libs/GenericHandler.

2.2.2.1.2 Bin Control Configuration

The default bin control functions can be configured using bincontrol.cfg. This path for this file is /opt/acs/nexus/conf/bincontrol.cfg.

Advanced bin control configurations listed in Table 2-1 can be specified through OneAPI command. Refer to class oneapi::Command for C++, or class Command for Java.

Table 2-1. Advanced Bin Control Configurations

Parameters	Type and [Default Value]	Description
`Enabled`	`Integer [0]`	0 or 1 can be specified (1 is enabled for this feature).
`Timeout`	`Sec [1]`	The number of seconds to detect a timeout.
`TimoutBin`	`Integer [9]`	The bin number if timeout is detected. **NOTE:** TimeoutAction should be TimeoutBin.
`TimeoutAction`	`Fixed String [TimeoutBin]`	The timeout action can be Abort, OriginalBin, or TimeoutBin. • Abort: Aborts the production. • OriginalBin: Use an original bin if timeout is detected. • TimeoutBin: Use binnumber specified with "TimeoutBin" if timeout is detected.
`BinningHistoryDir`	`String for directory path []`	The log file path for binning history.
`IllBinAction`	`Fixed String [Abort]`	Defines the behavior if an illegal bin is specified. The action can be Abort or AltBin. • Abort: Abort the production. • AltBin: Use binnumber specified with "IllAltBin" if the dedicated bin is out of the specified range. The range is specified by MinValidAltBin and MaxValidAltBin
`IllAltBin*`	`Integer [-1]`	The bin number if the specified bin is out of range.
`MinValidAltBin*`	`Integer [-1]`	The minimum bin number.
`MaxValidAltBin*`	`Integer [-1]`	The Maximum bin number.

* Even if IllAltBin is inside the range specified by MinValidAltBin and MaxValidAltBin, the driver may detect out of range based on the driver’s configuration rules.

2.2.2.2 Site Activity Control

Site Activity Control is an advanced control action that activates/deactivates test sites in real time during lot test execution. This feature operates in one of two modes:

- **Synchronous mode**
  Driver waits for the “SiteActivityControl Set” command to execute during the timeout period. If a timeout occurs, the driver takes a predefined action.
- **Asynchronous mode**
  Driver does not wait for the “SiteActivityControl Set” command to run. If the command is not executed before the test starts, lot execution will continue.

2.3 OneAPI C++ SDK

OneAPI is the standard bi-directional communication interface that enable containerized/non-containerized applications to consume real time data from (and send control command to) ACS Nexus.

An application developer can use OneAPI to develop an application to consume the real time data and send control instructions during production testing. ACS Nexus invokes a callback function of the containerized application per event.

In this version of OneAPI, C++ SDK and Python (3.9, 3.10 and 3.11) SDK are supported. This section describes the usage of OneAPI C++ SDK.

2.3.1 General Information

Package Contents

OneAPI C++ SDK provides programming interfaces in C++ / C++11 standard for the user to develop applications. All contents (see below) are packaged in a tar.gz file.

```
oneAPI_cpp

|-- Dockerfile.example
|-- build_base_image.sh
|    |-- oneAPI_conf.ini
|-- examples
|    |-- main.cpp
|    |-- makefile
|    |-- sampleMonitor.cpp
|    |-- sampleMonitor.h
|-- include
|    |-- rapidjson
|    |-- OneAPI.h
|    |-- OneAPI_AppInfo.h
|    |-- OneAPI_Command.h
|    |-- OneAPI_NexusData.h
|    |-- OneAPI_ServerInfo.h
|    |-- OneAPI_TestCell.h
|    |-- OneAPI DFFData.h
|-- lib
|    |-- liboneAPI.so
```

Table 2-2. OneAPI C++ Package Content Descriptions

Content	Description
`Dockerfile.example`	This file provides a reference for users on how to build a OneAPI Application image based on the base image. Users need to rewrite Dockerfile according to their actual situation. Refer to the comments in this file for specific rewriting methods.
`build_base_image.sh`	This script provides a reference for users on how to build an image with Dockerfile.
`examples`	This folder includes example code that demonstrates the use of OneAPI C++ for developers. This folder also contains the makefile used for compilation, which developers can use as a reference.
`include`	This folder contains all the header files provided by OneAPI C++ and describes the declarations of all interfaces and classes. When developers use OneAPI, all files in this folder must be included. This folder also provides the third-party library "Rapidjson" to help customers parse the DFF data Json string (for customers who use the DFF function).
`lib`	This folder contains all library files provided by OneAPI C++. Developers need to link these files for compilation and include them in the application release package or image.

Environment Requirements

Users can develop containerized applications running on the ACS Edge Server or containerized/non-containerized applications running on a test floor server that is based on OneAPI C++ SDK. For both scenarios, specific environment conditions are required, as noted below.

- **Application Development Environment**

  Supported OS: Red Hat 7 and CentOS 7

  Required Software: C++ compiler

- **Application Operating Environment**

  A successful connection at least needs to ensure that the network between Application Operating environment and ACS Nexus operation environment is enabled, and other dependent guarantees, such as ports availability, firewall permission, etc.

2.3.2 Usage Scenario

Basic and Advanced Test Cell Control

Table 2-3. Basic and Advanced Control Commands

Command	Description
`PAUSE`	PAUSE is a basic control command that provides real-time pause of wafer testing or final testing when user applications identify production issues. Actions taken after a pause are determined by the user (for example, automatic actions performed by other tools or manual action by the operator).
`STOP`	STOP is a basic control command that provides real-time stop of wafer testing or final testing when user applications identify production issues.
`Bin Control`	Bin Control is an advanced control command: • I/F 1: SetNewBin parameter: New bin of each site • I/F 2: SetNewBinConfig parameter: Enabled flag, Timeout, etc. **NOTE:** SetNewBinConfig should be executed sometime between Lot Start and Start of the first touchdown.
`Site Activity Control`	Site Activity Control is an advanced control command: • I/F 1: SiteActivityControl Set parameter: Site name and value • I/F 2: SiteActivityControl Config parameter: Activated flag, Timeout, TimoutAction, Sync, StatusQuery

Real Time Test Cell Data Collection

Comprehensive real time test cell data collection enables applications to perform data analysis in real time. Below is a list of OneAPI data types and a brief description of information that can be collected for each data type.

NOTE: The collected information provided for each data type in the list below is not comprehensive. For a complete list of information that is collected, refer to class oneapi::NexusData.

Table 2-4. OneAPI Data Types

Data Type	Information Collected
`PRODUCTION_LOTSTART`	Lot Start event includes the following information: • Lot start time • Lot ID, Sublot ID • Test type • Site list • Prober/Handler, LB . . .
`PRODUCTION_LOTEND`	Lot End event includes the following information: • Lot complete time • Lot ID . . .
`PRODUCTION_WAFERSTART`	Wafer Start event includes the following information: • Wafer start time • Wafer ID • Wafer layout information . . .
`PRODUCTION_WAFEREND`	Wafer End event includes the following information: • Wafer complete time • Wafer ID . . .
`PRODUCTION_TESTSTART`	Test Start event includes the following information: • Test start time • Site list
`PRODUCTION_TESTEND`	Test End event includes the following information: • Test complete time • Bin result of each site • Part ID of each site • X/Y Coordinates of each site • Test time . . .
`PRODUCTION_TESTFLOWSTART`	Test Flow Start event includes the following information: • Test flow start time • Test flow name

Data Type	Information Collected
`PRODUCTION_TESTFLOWEND`	Test Flow End event includes the following information: • Test flow end time • Test flow name
`MEASURED_PARAMETRIC`	Parametric Test Result event includes: • Test method information: number, name, lo/hi limit, unit • Result of each site: pass/fail, measured value, etc. . . .
`MEASURED_FUNCTIONAL`	Functional Test Result event includes: • Test method information: number, name • Result of each site: pass/fail, cycle count, fail-pins etc.
`MEASURED_MULTI_PARAM`	Multi-Parametric Test Result event includes: • Test method information: number, name, lo/hi limit, unit • Multiple measured values of each site . . .
`MEASURED_SCAN`	Scan Test Result event includes: • Test method info: number • Result of each Site: total cycle count, fail cycle count • Result of each Pattern: name, pins information • Result of each Pin: name, failing cycles …
`DEVICE`	Device data includes: • Test Program info: name, path, pin config file etc. • Bin table: soft-bin, hard-bin • Pin data: index, type, name, number etc. • Pattern data: index, name, pattern files etc.
`USERDEFINED`	User Defined data includes: • User specified variable name and its value
`DATALOGTEXT`	Datalog Text data includes: • Datalog text time stamp • Datalog text expression
`MEASUREMENT`	Measurement Data includes: • Measurement name • The execution information of parallel groups • The execution information of operating sequence calls

2.3.3 C++ API

2.3.3.1 class oneapi::Interface

This is a set of static functions which are used for communication between the user application and ACS Nexus. During the lifecycle of the application, no object of this class needs to be generated.

```
static int connect(const AppInfo&, bool NexusDataEnabled = true, bool TPServiceEnabled = false);
```

Description	Initiate the request to connect to ACS Nexus. Typically, the application only needs to input AppInfo. Connection must be established before sending control commands or receiving data.
Parameter	Input: • Information of the application • Whether to enable Nexus Data Streaming and Control • Whether to enable TPService for communication wwith NexusTPI
Return	0 – initial connection succeeded -1 – failed to enable Nexus Data Streaming and Control -2 – failed to enable TPService -3 – unknown location -4 – not support **NOTES:** The return value 0 does not mean a connection with ACS Nexus has been established. Another interface get`ConnectionState()` can provide the current connection state of the command channel. For a return value of -1, a possible reason could be failed to connect to nexus-broker. For a return value of -2, a possible reason could be TPService port is occupied. (the default TPService port is 21122). For a return value of -3, a possible reason could be OneAPI runs in an unknown location. For a return value -4, a possible reason could be OneAPI does not support the corresponding feature in the current location. For any exception case, all enabled connections will be disconnected.

```
static int disconnect();
```

Description	Disconnects from ACS Nexus.
Return	0 disconnect succeeded -1 disconnect failed

```
static void registerMonitor(Monitor* );
```

Description	Registers the event monitor to receive real-time data through a callback function. For additional details, refer to Class oneapi::Monitor.
Return	Input: Pointer of the event monitor object

```
static void getConnectionState(int &cmdChannel);
```

Description	Get the connection status of the command channel. For each channel, the status value is 0 only when the current status is established, and communication is available. Otherwise, the status value is – 1. The application should determine whether the related API can be executed according to the connection state. For example, when sending a control command, if the status of the command channel is not 0, the command will not be sent to ACS Nexus.
Parameter	Output: The status of command channel (for sending command to ACS Nexus).

```
static int sendCommand(const TestCell&, const Command& );
```

Description	Send the command to the connected ACS Nexus. The return value of this interface indicates whether the action of sending command is successful. ACS Nexus will execute the command asynchronously after receiving the command.
Parameter	Input: TestCell information (refer to class TestCell) Command information (refer to class oneapi::Command)
Return	0 command successfully sent to ACS Nexus -1 command failed to send to ACS Nexus -3 unknown location -4 not support

2.3.3.2 class oneapi::Monitor

This base class provides functions to consume ACS Nexus data. Application developers must inherit it and override the `consumeData` function, then write their own logic with the received data.

```
virtual void consumeData(const oneapi::TestCell&, oneapi::NexusData& )
```

Description	This is a pure virtual function that needs to be inherited and overridden. As a callback function, it will be triggered every time ACS Nexus data arrives. **NOTE 1:** When the function is triggered, the thread will be blocked. It cannot be triggered again until the callback function completes its processing logic and returns. Therefore, attention should be given to the efficiency of processing or consideration given to implementing asynchronous processing in the code. **NOTE 2:** You must call the corresponding *get* and *query* interfaces inside this function and save the values, otherwise the data will be cleared.
Parameter	• TestCell information (refer to class TestCell) • ACS Nexus data (refer to Class oneapi::NexusData)

```
virtual void consumeTPSend(const oneapi::TestCell& tc, const std::string& data)
```

Description	This function must be inherited and overridden. As a callback function, it will be triggered every time the test program initiates a request via NexusTPI. **NOTE:** When the function is triggered, the thread will be blocked. It cannot be triggered again until the callback function completes its processing logic and returns. Therefore, attention should be given to the efficiency of processing or consideration given to implementing asynchronous processing in the code.
Parameter	• TestCell information (refer to class TestCell) • Data is string type.

```
virtual std::string consumeTPRequest(const oneapi::TestCell& tc, const std::string&
request)
```

Description	This function must be inherited and overridden. As a callback function, it will be triggered every time the test program initiates a request via NexusTPI. **NOTE:** When the function is triggered, the thread will be blocked. It cannot be triggered again until the callback function completes its processing logic and returns. Therefore, attention should be given to the efficiency of processing or consideration given to implementing asynchronous processing in the code.
Parameter	• TestCell information (refer to class TestCell) • Request is string type in JSON format.
Return	string: After handle request returns your response to NexusTPI.

2.3.3.3 enum oneapi::DataType

Description	Indicate the type of ACS Nexus data. • `DATA_TYP_PRODUCTION_LOTSTART = 0` • `DATA_TYP_PRODUCTION_WAFERSTART = 1` • `DATA_TYP_PRODUCTION_TESTSTART = 2` • `DATA_TYP_PRODUCTION_TESTEND = 3` • `DATA_TYP_PRODUCTION_WAFEREND = 4` • `DATA_TYP_PRODUCTION_LOTEND = 5` • `DATA_TYP_MEASURED_PARAMETRIC = 6` • `DATA_TYP_MEASURED_FUNCTIONAL = 7` • `DATA_TYP_MEASURED_MULTI_PARAM = 8` • `DATA_TYP_DEVICE = 9` • `DATA_TYP_USERDEFINED = 10` • `DATA_TYP_FILE = 11` • `DATA_TYP_PRODUCTION_TESTFLOWSTART = 15` • `DATA_TYP_PRODUCTION_TESTFLOWEND = 16` • `DATA_TYP_DATALOGTEXT = 17` • `DATA_TYP_IDENTIFICATION = 18` • `DATA_TYP_DEVICE_PIN = 19` • `DATA_TYP_MEASURED_SCAN = 20` • `DATA_TYP_DEVICE_PATTERN = 21` • `DATA_TYP_MEASUREMENT = 22`

2.3.3.4 class oneapi::NexusData

Each time the callback function `consumeData` is triggered, a reference to the `NexusData` object will be passed in as a parameter. Detailed values can be obtained by calling the member functions of this class.

NOTE: The corresponding member functions must be called according to the type of data to obtain the current valid value.

[Production Event] Start of Lot
oneapi::DataType DATA_TYP_PRODUCTION_LOTSTART
Occurs: Once at the beginning of the first touchdown per LOT
Contains: All the global information for this LOT

Member Function	Return Value(s) Description
`int32  get_Timezone();`	The time zone where ACS Nexus is running
`uint64 get_SetupTime();`	Date and time when the test program was started (in microseconds)
`uint64 get_TimeStamp();`	Date and time when the first part was tested (in microseconds)
`uint32 get_StationNumber();`	Tester station number
`string get_ModeCode();`	Test mode code
`string get_RetestCode();`	Lot retest code
`string get_ProtectionCode();`	Data protection code
`uint32 get_BurnTimeMinutes();`	Burn-in time (in minutes)
`string get_CommandCode();`	Command mode code of the tester
`string get_LotId();`	Lot ID
`string get_PartType();`	Part type or product ID
`string get_NodeName();`	Hostname of the tester system controller
`string get_TesterType();`	Tester type
`string get_JobName();`	Test program name
`string get_JobRevision();`	Test program revision number
`string get_SublotId();`	Sublot ID
`string get_OperatorName();`	Operator name or ID at setup time
`string get_TesterosType();`	Tester software type
`string get_TesterosVersion();`	Tester software version number
`string get_TestType();`	Type of Lot (PACKAGE_TEST / WAFER_TEST)
`string get_TestStepCode();`	Test phase or step code
`string get_TestTemperature();`	Test temperature
`string get_UserText();`	User-defined text
`string get_AuxiliaryFile();`	Name of auxiliary data file
`string get_PackageType();`	Package type
`string get_FamilyId();`	Product family ID
`string get_DateCode();`	Date code
`string get_FacilityId();`	Test facility ID
`string get_FloorId();`	Test floor ID

`string get_ProcessId();`	Fabrication process ID
`string get_OperationFreq();`	Operation frequency or step
`string get_SpecName();`	Test specification name
`string get_SpecVersion();`	Test specification version number
`string get_FlowId();`	Testflow ID
`string get_SetupId();`	Test setup ID
`string get_DesignRevision();`	Device design revision
`string get_EngineeringLotId();`	Engineering lot ID
`string get_RomCode();`	ROM code ID
`string get_SerialNumber();`	Tester serial number
`string get_SupervisorName();`	Supervisor name or ID
`uint32 get_HeadNumber();`	Test head number
`uint32 get_SiteGroupNumber();`	Site group number (station number)
`uint32 get_SiteCount();`	Number of active sites described in this record
`vector<uint32> get_TotalHeadSiteList();`	Array of test site numbers with head information
`string get_ProberHandlerType();`	Handler or prober type
`string get_ProberHandlerId();`	Handler or prober ID
`string get_ProbecardType();`	Probe card type
`string get_ProbecardId();`	Probe card ID
`string get_LoadboardType();`	DUT board type
`string get_LoadboardId();`	DUT board ID
`string get_DibType();`	DIB board type
`string get_DibId();`	DIB board ID
`string get_CableType();`	Interface cable type
`string get_CableId();`	Interface cable ID
`string get_ContactorType();`	Handler contactor type
`string get_ContactorId();`	Handler contactor ID
`string get_LaserType();`	Laser type
`string get_LaserId();`	Laser ID
`string get_ExtraEquipType();`	Extra equipment type
`string get_ExtraEquipId();`	Extra equipment ID

[Production Event] End of Lot
oneapi::DataType DATA_TYP_PRODUCTION_LOTEND
Occurs: Once at the ending of the last touchdown per LOT
Contains: Information that compliments the Lot

Member Function	Return Value Description
uint64 get_TimeStamp();	Date and time last part of the lot was tested (in microseconds)
string get_DisPositionCode();	Lot disposition code
string get_UserDescription();	Lot description supplied by the user
string get_ExecDescription();	Lot description supplied by the exec

[Production Event] Start of Wafer
oneapi::DataType DATA_TYP_PRODUCTION_WAFERSTART
Occurs: Once at the beginning of the first touchdown per WAFER
Contains: All the configuration information for the tested wafer

Member Function	Return Value Description
float get_WaferSize();	Diameter of wafer in WF_UNITS
float get_DieHeight();	Height of die in WF_UNITS
float get_DieWidth();	Width of die in WF_UNITS
uint32 get_WaferUnits();	Unit for wafer and die dimensions
string get_WaferFlat();	Orientation of wafer flat
int32 get_CenterX();	X-coordinate of center die on wafer
int32 get_CenterY();	Y-coordinate of center die on wafer
string get_PositiveX();	Positive X-direction of wafer
string get_PositiveY();	Positive Y-direction of wafer
uint32 get_HeadNumber();	Test head number
uint32 get_SiteGroupNumber();	Site group number
uint64 get_TimeStamp();	Date and time when the first part of wafer was tested (in microseconds)
string get_WaferId();	Wafer ID

[Production Event] End of Wafer

oneapi::DataType DATA_TYP_PRODUCTION_WAFEREND

Occurs: Once at the ending of the last touchdown per WAFER
Contains: The result information for the tested wafer

Member Function	Return Value Description
uint32 get_HeadNumber();	Test head number
uint32 get_SiteGroupNumber();	Site group number
uint64 get_TimeStamp();	Date and time when the last part of wafer was tested (in microseconds)
uint32 get_TestedCount();	Number of tested parts
uint32 get_RetestedCount();	Number of retested parts
uint32 get_GoodCount();	Number of tested parts that have passed
string get_WaferId();	Wafer ID
string get_UserDescription();	Wafer description supplied by user
string get_ExecDescription();	Wafer description supplied by exec

[Production Event] End of Test

oneapi::DataType DATA_TYP_PRODUCTION_TESTEND

Occurs: Once at the ending of each touchdown
Contains: Site, part, and result information for this touchdown

Member Function	Return Value Description
uint64 get_TimeStamp();	Date and time when the part completes test (in microseconds)
uint32 get_ResultCount();	Count of test result
uint32 query_HeadSite(uint32 index);	Test site number with head information
string query_PartFlag(uint32 index);	Part information flags
uint32 query_SBinResult(uint32 index);	Software bin
uint32 query_HBinResult(uint32 index);	Hardware bin
int32 query_XCoord(uint32 index);	Wafer X-coordinate
int32 query_YCoord(uint32 index);	Wafer Y-coordinate
uint32 query_TestTime(uint32 index);	Elapsed test time in microseconds
string query_PartId(uint32 index);	Part identification
string query_PartText(uint32 index);	Part description text

[Production Event] Start of Test

oneapi::DataType DATA_TYP_PRODUCTION_TESTSTART

Occurs: Once at the beginning of each touchdown
Contains: Site information for this touchdown

Member Function	Return Value Description
`uint64 get_TimeStamp();`	Date and time when the part starts to test (in microseconds)
`uint32 query_HeadSite(uint32 index);`	Test site number with head information
`int32  query_XCoord(uint32 index);`	Wafer X-coordinate
`int32  query_YCoord(uint32 index);`	Wafer Y-coordinate

[Production Event] Start of Test Flow

oneapi::DataType DATA_TYP_PRODUCTION_TESTFLOWSTART

Occurs: Once at the beginning of each test flow
Contains: Test flow name

Member Function	Return Value Description
`uint64 get_TimeStamp();`	Date and time when the test flow starts (in microseconds)
`string get_TestFlowName();`	Test flow name

[Production Event] End of Test Flow

oneapi::DataType DATA_TYP_PRODUCTION_TESTFLOWEND

Occurs: Once at the ending of each test flow
Contains: Test flow name

Member Function	Return Value Description
`uint64 get_TimeStamp();`	Date and time when the test flow completes (in microseconds)
`string get_TestFlowName();`	Test flow name

[Measured Value] Parametric Test Result

oneapi::DataType DATA_TYP_MEASURED_PARAMETRIC

Occurs: Once per parametric test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`uint32 get_ResultCount();`	Count of result for the test item
`uint32 query_HeadSite(uint32 index);`	Test site number with head information
`uint32 query_TestNumber(uint32 index);`	Test number
`string query_TestText(uint32 index);`	Test description text or label[1]
`float  query_LowLimit(uint32 index);`	Low test limit value
`float  query_HighLimit(uint32 index);`	High test limit value
`string query_Unit(uint32 index);`	Test units
`string query_TestFlag(uint32 index);`	Test flags (fail, alarm, etc.)
`float  query_Result(uint32 index);`	Test result
`string query_TestSuite(uint32 index);`	Test suite name
`string query_MeasurementName(uint32 index);`	The unique fully qualified name of measurement

[1] In SmarTest 7, the priority is the test comment, followed by "Test suite name: test name." If there is a pin name, it will be "Test suite name: test name : pin name." In SmarTest 8, it will be always the test descriptor.

[Measured Value] Functional Test Result

oneapi::DataType DATA_TYP_MEASURED_FUNCTIONAL

Occurs: Once per functional test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`uint32 get_ResultCount();`	Count of result for the test item
`uint32 query_HeadSite(uint32 index);`	Test site number with head information
`uint32 query_TestNumber(uint32 index);`	Test number
`string query_TestText(uint32 index);`	Test description text or label
`string query_TestFlag(uint32 index);`	Test flags (fail, alarm, etc.)
`uint32 query_CycleCount(uint32 index);`	Vector cycle count
`uint32 query_NumberFail(uint32 index);`	The number of logic pin names with one or more failures
`vector<uint64> query_PinResults(uint64 index);`	List of fail Pin index which belong to the given Result index
`string query_PinName(uint64 pinID);`	Pin name
`vector<uint32> query_FailCycles(uint64 pinID);`	List of fail cycle numbers which belong to the given Pin
`string query_VectNam(uint32 index);`	Vector module pattern name
`string query_TestSuite(uint32 index);`	Test suite name

`string query_MeasurementName(uint32 index);`	The unique fully qualified name of measurement

[Measured Value] Multiple-Result Parametric Test Record

oneapi::DataType DATA_TYP_MEASURED_MULTI_PARAM

Occurs: Once per multiple-result parametric test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	**Return Value Description**
`uint32 get_ResultCount();`	Count of result for the test item
`uint32 query_HeadSite(uint32 index);`	Test site number with head information
`uint32 query_TestNumber(uint32 index);`	Test number
`string query_TestText(uint32 index);`	Test description text or label
`float  query_LowLimit(uint32 index);`	Low test limit value
`float  query_HighLimit(uint32 index);`	High test limit value
`string query_Unit(uint32 index);`	Test units
`string query_TestFlag(uint32 index);`	Test flags (fail, alarm, etc.)
`vector<float> query_Results(uint32 index);`	List of Test results which belong to the given Result index
`vector<uint64> query_PinResults(uint64 index);`	List of Pin index which belong to the given Result index
`string query_PinName(uint64 pinID);`	Pin name
`string query_TestSuite(uint32 index);`	Test suite name
`string query_MeasurementName(uint32 index);`	The unique fully qualified name of measurement

[Notify] Scan Test Result

oneapi::DataType DATA_TYP_MEASURED_SCAN

Occurs: Once per scan test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`uint32 get_ResultCount();`	Count of result for the test item
`uint32 query_HeadSite(uint32 index);`	Test site number with head information
`uint32 query_TestNumber(uint32 index);`	Test number
`string query_TestText(uint32 index);`	Test description text or label
`string query_TestFlag(uint32 index);`	Test flags (fail, alarm, , etc.)
`int32 query_TotalCycleCount(uint32 index);`	Total number of cycles
`int32 query_FailCycleCount(uint32 index);`	Total number of failing cycles
`string query_OpSequence(uint32 index);`	Operating sequence
`vector<uint64> query_PatternResults(uint32 index);`	List of Pattern index which belong to the given Result
`string query_PatternName(uint64 patternID);`	Pattern name
`vector<uint64> query_PinResults(uint64 index);`	List of Pin index which belong to the given Pattern
`string query_PinName(uint64 pinID);`	Pin name
`vector<uint32> query_FailCycles(uint64 pinID);`	List of fail cycle numbers which belong to the given Pin
`string query_TestSuite(uint32 index);`	Test suite name
`string query_MeasurementName(uint32 index);`	The unique fully qualified name of measurement

[Device Data]

oneapi::DataType DATA_TYP_DEVICE

Occurs: Once for each lot, when all the required data collection is completed
Contains: Device information

Member Function	Return Value Description
`string get_TestProgramDir();`	The absolute path of the folder of the active test program
`string get_TestProgramName();`	The fully qualified name of the active test program
`string get_TestProgramPath();`	The absolute path of the active test program file
`string get_PinConfig();`	The fully qualified name of the active DUT board description file
`string get_ChannelAttribute();`	The fully qualified name of the active Channel attributes file
`uint32 get_BinInfoCount();`	Count of defined soft bin of the active test program
`uint32 query_SBinNumber(uint32 index);`	The number of the soft bin
`string query_SBinName(uint32 index);`	The description of the soft bin
`uint32 query_SBinType(uint32 index);`	The type of the soft bin
`uint32 query_HBinNumber(uint32 index);`	The corresponding hard bin number
`string query_HBinName(uint32 index);`	The description of the corresponding hard bin
`uint32 query_HBinType(uint32 index);`	The type of the corresponding hard bin

[Notify] Pin Data

oneapi::DataType DATA_TYP_DEVICE_PIN

Occurs: Once for each lot, when all the required data collection is completed
Contains: Pin information

Member Function	Return Value Description
`uint32 get_PinInfoCount();`	The count of pin information
`uint32 query_PinIndex(uint32 index);`	Pin index
`uint32 query_ChannelType(uint32 index);`	Channel type
`string query_ChannelName(uint32 index);`	Channel name
`string query_PhysicalName(uint32 index);`	Name of physical pin
`string query_LogicalName(uint32 index);`	Name of logical pin
`uint32 query_HeadNumber(uint32 index);`	Test head number
`uint32 query_SiteNumber(uint32 index);`	Test site number

[Notify] Pattern Data
oneapi::DataType DATA_TYP_DEVICE_PATTERN
Occurs: Whenever a pattern variable is collected
Contains: Information on a pattern profile for a specific scan test

Member Function	Return Value Description
`uint32 get_PatternCount();`	The count of patterns
`string query_OpSequence(uint32 index);`	Operating sequence
`vector<string> query_PatternLabels(uint32 index);`	List of pattern labels

[User Defined Data]
oneapi::DataType DATA_TYP_USERDEFINED
Occurs: Whenever the user-defined variable is collected
Contains: Variable and its value information

Member Function	Return Value Description
`map<string,string> get_UserDefined();`	Map of user defined data <variable, value>

[Notify] File Transfer
oneapi::DataType DATA_TYP_FILE
Occurs: When each file transfer is completed
Contains: The file name

Member Function	Return Value Description
`string get_FileName();`	The absolute path of the received file

[Notify] Datalog Text
oneapi::DataType DATA_TYP_DATALOGTEXT
Occurs: Whenever a datalog-text variable is collected
Contains: The expression of datalog-text

Member Function	Return Value Description
`uint64 get_TimeStamp();`	Date and time when the test flow completes (in microseconds)
`string get_DataLogText();`	The expression of datalog text

[Notify] Measurement Data

oneapi::DataType DATA_TYP_MEASUREMENT

Occurs: Once a measurement has ended
Contains: The information of measurement

Member Function	**Return Value Description**
`string get_MeasurementName();`	The unique fully qualified name of the measurement
`uint32 get_GroupCount();`	Total count of parallel groups
`string query_GroupName(uint32 index);`	The name of the parallel group
`int32 query_GroupBypassed(uint32 index);`	Whether the parallel group was bypassed
`vector<uint32> query_GroupSites(uint32 index);`	The number of the site(s) where the parallel group was bypassed
`uint32 get_SequenceCount();`	Total count of operating sequence
`string query_SequenceName(uint32 index);`	The name of the operating sequence
`int32 query_SequenceBypassed(uint32 index);`	Whether the operating sequence was bypassed
`vector<uint32> query_SequenceSites(uint32 index);`	The number of site(s) where the operating sequence was bypassed
`vector<uint32> query_SequenceGroupResults (uint32 index);`	The list of unique "GroupID" for the specific sequence
`string query_SequenceGroupName(uint32 groupID);`	The name of the parallel group that belongs to the current sequence
`int32 query_SequenceGroupBypassed(uint32 groupID);`	Whether the parallel group was bypassed
`vector<uint32> query_SequenceGroupSites(uint32 groupID);`	The number of the site(s) where the parallel group was bypassed

2.3.3.4.1 Function Introduction

`oneapi::DataType NexusData::getType();`

Description	Get the type of data currently consumed. **NOTE:** This function must be called before all other functions in the consumeData code.
Return	One of the following: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_TESTSTART` • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_PRODUCTION_WAFEREND` • `DATA_TYP_PRODUCTION_LOTEND` • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_DEVICE` • `DATA_TYP_USERDEFINED` • `DATA_TYP_FILE` • `DATA_TYP_PRODUCTION_TESTFLOWSTART` • `DATA_TYP_PRODUCTION_TESTFLOWEND` • `DATA_TYP_DATALOGTEXT` • `DATA_TYP_IDENTIFICATION` • `DATA_TYP_DEVICE_PIN` • `DATA_TYP_MEASURED_SCAN` • `DATA_TYP_DEVICE_PATTERN`

`uint64 NexusData::get_TimeStamp();`

Description	Get the date and time in microseconds. Valid for: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_LOTEND` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_WAFEREND` • `DATA_TYP_PRODUCTION_TESTSTART` • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_PRODUCTION_TESTFLOWSTART` • `DATA_TYP_PRODUCTION_TESTFLOWEND`
Return	If by one of the above events, returns a valid value. For example, if event is `DATA_TYP_PRODUCTION_LOTSTART`, then the value represents the timestamp of `Lot_Start`. Otherwise, returns 0.

`string NexusData::get_TestProgramDir();`

Description	Get the absolute path of the folder of the active test program. Valid always, refreshed by `DATA_TYP_DEVICE.`
Return	Valid value.

`string NexusData::get_TestProgramName()`

Description	Get the fully qualified name of the active test program. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`string NexusData::get_TestProgramPath();`

Description	Get the absolute path of the active test program file. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`string NexusData::get_PinConfig();`

Description	Get the fully qualified name of the active DUT board description file. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`string NexusData::get_ChannelAttribute();`

Description	Get the fully qualified name of the active Channel attributes file. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`uint32 NexusData::get_BinInfoCount();`

Description	Get the count of defined soft bin of the active test program. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`uint32 NexusData::query_SBinNumber(uint32 index);`

Description	Query the number of the soft bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

`string NexusData::query_SBinName(uint32 index);`

Description	Query the description of the soft bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`uint32 NexusData::query_SBinType(uint32 index);`

Description	Query the type of soft bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	If valid input index, returns: 0 – PASS 1 – FAIL 2 – UNKONWN If input index is out of bounds, returns 2.

```
uint32 NexusData::query_HBinNumber(uint32 index);
```

Description	Query the number of hard bins. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

```
string NexusData::query_HBinName(uint32 index);
```

Description	Query the description of the hard bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

```
uint32 NexusData::query_HBinType(uint32 index);
```

Description	Query the type of hard bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	If valid input index, returns: 0 – PASS 1 – FAIL 2 – UNKONWN If input index is out of bounds, returns 2.

```
uint32 NexusData::get_PatternCount();
```

Description	Get the count of patterns. Valid always, refreshed by DATA_TP_DEVICE_PATTERN.
Return	Returns valid value in all cases.

`uint32 NexusData::query_OpSequence(uint32 index);`

Description	Query the operating sequence. Valid always, refreshed by • `DATA_TYP_DEVICE_PATTERN` • `DATA_TYP_MEASURED_`
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`vector<string> NexusData::query_PatternLables(uint32 index);`

Description	Query the list of pattern labels. Valid for `DATA_TYP_DEVICE_PATTERN`.
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`map<string,string> NexusData::get_UserDefined();`

Description	Get the user defined key and value pairs. Valid for `DATA_TYP_USERDEFINED`.
Return	Returns valid value if by above event, otherwise returns map with no elements.

`string NexusData::get_FileName();`

Description	Get the absolute path of the received file. The file is in the 'home' folder of the current user. Valid always, refreshed by `DATA_TYP_FILE`.
Return	Returns valid value if by above event, otherwise returns "" (empty string).

`int32  NexusData::get_Timezone();`

Description	Get the time zone where ACS Nexus is running (value in integer). Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`uint64 NexusData::get_SetupTime();`

Description	Get the date and time (in microseconds) when the test program was started. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`uint32 NexusData::get_StationNumber();`

Description	Get the tester station number (value in integer). Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ModeCode();`

Description	Get the test mode code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_RetestCode();`

Description	Get the lot retest code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ProtectionCode();`

Description	Get the data protection code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`uint32 NexusData::get_BurnTimeMinutes();`

Description	Get the burn-in time (in minutes). Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_CommandCode();`

Description	Get the command mode code of the tester. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns " (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_LotId();`

Description	Get the lot ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_PartType();`

Description	Get the part type or product ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_NodeName();`

Description	Get the hostname of the tester system controller. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_TesterType();`

Description	Get the tester type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_JobName();`

Description	Get the test program name. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_JobRevision();`

Description	Get the test program revision number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_SublotId();`

Description	Get the sublot ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_OperatorName();`

Description	Get the operator name or ID at setup time. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_TesterosType();`

Description	Get the software type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_TesterosVersion();`

Description	Get the tester software version number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_TestType();`

Description	Get the type of lot. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value (PACKAGE_TEST or WAFER_TEST) in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_TestStepCode();`

Description	Get the test phase or step code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_TestTemperature();`

Description	Get the test temperature. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_UserText();`

Description	Get the user defined text. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_AuxiliaryFile();`

Description	Get the name of the auxiliary data file. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_PackageType();`

Description	Get the package type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_FamilyId();`

Description	Get the product family ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_DateCode();`

Description	Get the date code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_FacilityId();`

Description	Get the test facility ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_FloorId();`

Description	Get the test floor ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ProcessId();`

Description	Get the fabrication process ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_OperationFreq();`

Description	Get the operation frequency or step. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

string NexusData::get_SpecName();

Description	Get the test specification name. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

string NexusData::get_SpecVersion();

Description	Get the specification version number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

string NexusData::get_FlowId();

Description	Get the test flow ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_SetupId();`

Description	Get the test setup ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_DesignRevision();`

Description	Get the device design revision. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_EngineeringLotId();`

Description	Get the engineering lot ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_RomCode();`

Description	Get the ROM code ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_SerialNumber();`

Description	Get the tester serial number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_SupervisorName();`

Description	Get the supervisor name or ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`uint32 NexusData::get_HeadNumber();`

Description	Get the test head number. Valid by: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_WAFEREND`
Return	In the case of Lot Start, returned value represents the test head number of the lot. In the case of Wafer Start or Wafer End, returned value represents test head number of the wafer. Otherwise, returns 0.

`uint32 NexusData::get_SiteGroupNumber();`

Description	Get the site group number. Valid by: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_WAFEREND`
Return	In the case of Lot Start, returned value represents the site group number of the lot. In the case of Wafer Start or Wafer End, returned value represents site group number of the wafer. Otherwise, returns 0.

`uint32 NexusData::get_SiteCount();`

Description	Get the number of active sites described in this record. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`vector<uint32> NexusData::get_TotalHeadSiteList();`

Description	Get all test site numbers. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns no element in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ProberHandlerType();`

Description	Get handler or prober type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ProberHandlerId();`

Description	Get the handler or prober ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ProbecardType();`

Description	Get the probe card type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ProbecardId();`

Description	Get the probe card ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_LoadboardType();`

Description	Get the DUT board type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_LoadboardId();`

Description	Get the DUT board ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_DibType();`

Description	Get the DIB board type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_DibId();`

Description	Get the DIB board ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_CableType();`

Description	Get the interface cable type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_CableId();`

Description	Get the interface cable ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ContactorType();`

Description	Get the hander contactor type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ContactorId();`

Description	Get the hander contactor ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_LaserType();`

Description	Get the laser type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_LaserId();`

Description	Get the laser ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ExtraEquipType();`

Description	Get extra equipment type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_ExtraEquipId();`

Description	Get extra equipment ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`string NexusData::get_DisPositionCode();`

Description	Get lot disposition code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of the above event. Otherwise, returns "" (empty string)

string NexusData::get_UserDescription();

Description	Get the description supplied by the user. Valid by: • DATA_TYP_PRODUCTION_LOTEND • DATA_TYP_PRODUCTION_WAFEREND
Return	In the case of Lot End, value represents the lot description supplied by the user. In the case of Wafer End, value represents the wafer description supplied by the user. Otherwise, returns "" (empty string).

string NexusData::get_ExecDescription();

Description	Get the description supplied by the executive. Valid by: • DATA_TYP_PRODUCTION_LOTEND • DATA_TYP_PRODUCTION_WAFEREND
Return	In the case of Lot End, value represents the lot description supplied by the user. In the case of Wafer End, value represents the wafer description supplied by the user. Otherwise, returns "" (empty string).

float NexusData::get_WaferSize();

Description	Get the diameter of the wafer in WF_UNITS. Refreshed by: • DATA_TYP_PRODUCTION_WAFERSTART
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0.0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`float  NexusData::get_DieHeight();`

Description	Get the height of the die in WF_UNITS. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0.0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`float  NexusData::get_DieWidth();`

Description	Get the width of the die in WF_UNITS. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0.0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`uint32 NexusData::get_WaferUnits();`

Description	Get the unit for wafer and die dimensions. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

```
string NexusData::get_WaferFlat();
```

Description	Get the orientation of the wafer flat. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

```
int32  NexusData::get_CenterY();
```

Description	Get the Y-coordinate of the center die on the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns -32787 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

```
int32  NexusData::get_CenterX();
```

Description	Get the X-coordinate of the center die on the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns -32787 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`string NexusData::get_PositiveX();`

Description	Get the positive X-direction of the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`string NexusData::get_PositiveY();`

Description	Get the positive Y-direction of the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`string NexusData::get_WaferId();`

Description	Get the wafer ID. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`uint32 NexusData::get_TestedCount();`

Description	Get the number of the tested parts. Refreshed by: • `DATA_TYP_PRODUCTION_WAFEREND`
Return	Returns a valid value in the case above, otherwise returns 0.

`uint32 NexusData::get_RetestedCount();`

Description	Get the number of the retested parts. Refreshed by: • `DATA_TYP_PRODUCTION_WAFEREND`
Return	Returns a valid value in the case above, otherwise returns 0.

`uint32 NexusData::get_GoodCount();`

Description	Get the number of the tested parts that have passed. Refreshed by: • `DATA_TYP_PRODUCTION_WAFEREND`
Return	Returns a valid value in the case above, otherwise returns 0.

`vector<uint32> NexusData::get_HeadSiteList();`

Description	Get array of test site number with head information. Valid for `DATA_TYP_PRODUCTION_TESTSTART` **NOTE:** "headsite" is a new concept proposed by OneAPI. It integrates head and site information into an integer. The following methods can convert headsite to head or site: • uint32 toHead(uint32 index); • uint32 toSite(uint32 index);
Return	Returns a valid value in the case above, otherwise returns no element.

`uint32 NexusData::get_ResultCount();`

Description	Get the count of the test result**.** Valid for: • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	In the case of the above events, returns a valid value. For example, if event is DATA_TYP_PRODUCTION_TESTEND, then the value represents the count of DUTs of the current touchdown. If event is DATA_TYP_MEASURED_FUNCTIONAL, then the value represents the count of functional test result for the current test item. Otherwise, returns 0

`uint32 NexusData::query_HeadSite(uint32 index);`

Description	Query the test site number with head information. Valid by: • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN` • `DATA_TYP_PRODUCTION_TESTSTART`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`string NexusData::query_PartFlag(uint32 index);`

Description	Query the part information flags. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

uint32 NexusData::query_SBinResult(uint32 index);

Description	Query the soft bin number. Valid by: • DATA_TYP_PRODUCTION_TESTEND
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

uint32 NexusData::query_HBinResult(uint32 index);

Description	Query the hard bin number. Valid by: • DATA_TYP_PRODUCTION_TESTEND
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

int32 NexusData::query_XCoord(uint32 index);

Description	Query the X-coordinate. Valid by: • DATA_TYP_PRODUCTION_TESTEND • DATA_TYP_PRODUCTION_TESTSTART
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

int32 NexusData::query_YCoord(uint32 index);

Description	Query the Y-coordinate. Valid by: • DATA_TYP_PRODUCTION_TESTEND • DATA_TYP_PRODUCTION_TESTSTART
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

`uint32 NexusData::query_TestTime(uint32 index);`

Description	Query the elapsed test time in microseconds. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`string NexusData::query_PartId(uint32 index)`

Description	Query the part identification. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`string NexusData::query_PartText (uint32 index);`

Description	Query the part description text. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`uint32 NexusData::query_TestNumber(uint32 index)`

Description	Get the number of current test item. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the number of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`string NexusData::query_TestText(uint32 index);`

Description	Query test description text or label. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test text of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`float  NexusData::query_LowLimit(uint32 index);`

Description	Query the low test limit value. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the low limit of the current parametric test. Returns 0.0 if input index is out of bounds.

`float  NexusData::query_HighLimit(uint32 index);`

Description	Query the high test limit value. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the high limit of the current parametric test. Returns 0.0 if input index is out of bounds.

`string NexusData::query_Unit(uint32 index);`

Description	Query the test units. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test unit of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`string NexusData::query_TestFlag(uint32 index);`

Description	Query the test flags, including fail, alarm, etc. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test flags of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`string NexusData::query_TestSuite(uint32 index);`

Description	Query the test suite name. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test suite name of the current parametric test. Returns "" (empty string) if input index is out of bounds.

string NexusData::query_MeasurementName(uint32 index);

Description	Query the measurement name. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the measurement name of the current parametric test. Returns "" (empty string) if input index is out of bounds.

float NexusData::query_Result(uint32 index);

Description	Query the test result. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC`
Return	Returns a valid value if valid input index. Returns 0.0 if input index is out of bounds.

uint32 NexusData::query_CycleCount(uint32 index);

Description	Query the vector cycle count. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

uint32 NexusData::query_NumberFail(uint32 index);

Description	Query the vector cycle count. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`vector<string> NexusData::query_FailPins(uint32 index);`

Description	Query the list of failing pin bit field. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`string NexusData::query_VectNam(uint32 index);`

Description	Query the vector module pattern name. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`vector<float> NexusData::query_Results(uint32 index);`

Description	Query the list of all test results. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`string NexusData::get_TestFlowName();`

Description	Get the test flow name. Valid by: • `DATA_TYP_PRODUCTION_TESTFLOWSTART` • `DATA_TYP_PRODUCTION_TESTFLOWEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`string NexusData::get_DataLogText();`

Description	Get the data log text. Valid by: • DATA_TYP_DATALOGTEXT
Return	Returns a valid value in the case of the above event. Otherwise, returns "" (empty string).

`int32 NexusData::query_TotalCycleCount(uint32 index);`

Description	Query the total number of cycles. Valid by: • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`int32 NexusData::query_FailCycleCount(uint32 index);`

Description	Query the total number of failing cycles. Valid by: • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`vector<uint64> NexusData::query_PatternResults(uint32 index);`

Description	Query the list of pattern index results. Valid by: • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`string NexusData::query_PatternName(uint32 index);`

Description	Query the pattern name. Valid by: • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Otherwise returns "NOT_EXIST"

`vector<uint64> NexusData::query_PinResults(uint64 index);`

Description	Query the list of pin index results. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`string NexusData::query_PinName(uint64 index);`

Description	Query the pin name. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Otherwise, returns "NOT_EXIST"

`vector<uint64> NexusData::query_FailCycles(uint64 pinID);`

Description	Query the list of fail cycle numbers. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`uint32 NexusData::get_PatternCount();`

Description	Retrieve the total number of patterns. Valid by: • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if by above event. Otherwise, returns 0.

`vector<string> NexusData::query_PatternLabels(uint32 index);`

Description	Query the list of pattern names. Valid by: • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`uint32 NexusData::get_PinInfoCount();`

Description	Retrieve the total number of pin information. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if by above event. Otherwise, returns 0.

`uint32 NexusData::query_ChannelType(uint32 index);`

Description	Query the channel type. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if valid input index. Returns 0 if valid input index is out of bounds.

`string NexusData::query_ChannelName(uint32 index);`

Description	Query the channel name. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`string NexusData::query_PhysicalName(uint32 index);`

Description	Query the name of the physical pin. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`string NexusData::query_LogicalName(uint32 index);`

Description	Query the name of the logical pin. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`uint32 NexusData::query_HeadNumber(uint32 index);`

Description	Query the test head number. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`uint32 NexusData::query_SiteNumber(uint32 index);`

Description	Query the test site number. Valid by: • `DATA_TYP_DEVICE_PIN`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`uint32 NexusData::get_MeasurementName();`

Description	Get the measurement name. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if by above event. Otherwise, returns "" (empty string).

`uint32 NexusData::get_GroupCount();`

Description	Get the total count of parallel groups. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if by above event. Otherwise, returns 0.

`string NexusData::query_GroupName(uint32 index);`

Description	Query the name of parallel groups. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`int32 NexusData::query_GroupBypassed(uint32 index);`

Description	Query whether the parallel group was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	If valid input index, returns one of the following: • 0: bypassed is set to ‘false’ • 1: bypassed is set to ‘true’ • -1: bypassed is not set Returns -1 if input index is out of bounds.

`vector<uint32> NexusData::query_GroupSites(uint32 index);`

Description	Query the number of the site where the parallel group was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element (meaning that all sites were bypassed) if input index is out of bounds.

`uint32 NexusData::get_SequenceCount();`

Description	Get the total count of operating sequence call. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if by above event. Otherwise, returns 0.

```
string NexusData::query_SequenceName(uint32 index);
```

Description	Query the name of operating sequence call. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

```
int32 NexusData::query_SequenceBypassed(uint32 index);
```

Description	Query whether the operating sequence call was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	If valid input index, returns one of the following: • 0: bypassed is set to ‘false’ • 1: bypassed is set to ‘true’ • -1: bypassed is not set Returns -1 if input index is out of bounds.

```
vector<uint32> NexusData::query_SequenceSites(uint32 index);
```

Description	Query the number of the site where the operating sequence call was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element (meaning that all sites were bypassed) if input index is out of bounds.

```
vector<uint32> NexusData::get_SequenceGroupResults(uint32 index);
```

Description	Query the list of unique "Group ID" for the specific sequence. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`string NexusData::query_SequenceGroupName(uint32 groupID);`

Description	Query the name of the parallel group that belongs to the current sequence. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`int32 NexusData::query_SequenceGroupBypassed(uint32 groupID);`

Description	Query whether the parallel group was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	If valid input index, returns one of the following: • 0: bypassed is set to ‘false’ • 1: bypassed is set to ‘true’ • -1: bypassed is not set Returns -1 if input index is out of bounds.

`vector<uint32> NexusData::query_SequenceGroupSites(uint32 groupID);`

Description	Query the number of the site where the parallel group was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element (meaning that all sites for the current sequence were bypassed) if input index is out of bounds.

2.3.3.5 class oneapi::Command

This class is used to schedule the control capability of ACS Nexus through the oneapi::Interface:: sendCommand. Each time the interface is called, an object of oneapi::Command needs to be created and passed in to the interface as an argument, as shown in the example below.

```
oneapi::Command cmd;
cmd.name = “PAUSE”;
cmd.reason = “Yield is lower than 80 percent.”;
oneapi::TestCell tc;
int res = oneapi::Interface::sendCommand(tc, cmd);
if(res != 0)
    cout << "Send command fail. code = " << res << endl;
```

Member

`string name;`

Description	The command name should be one of the following: • PAUSE • STOP • SetNewBin • SetNewBinConfig • SiteActivityControl Set • SiteActivityControl Config

`string param;`

Description	The necessary parameters for this command.
	This variable is string type, so all information must be assembled into a string. For example:
	`oneapi::Command cmd;`
	`cmd.name = “SetNewBinConfig”;`
	`cmd.param = “Timeout 2 TimeoutAction Abort IllBinAction AltBin`
	`IllAltBin 9”;`
	`PAUSE`
	• None
	`STOP`
	• None
	`SetNewBin`
	• Mapping of Site and Bin
	`SetNewBinConfig`
	• Enabled
	o 0
	o 1
	• Timeout (in seconds)
	• TimeoutAction
	o Abort
	o OriginalBin
	o TimeoutBin
	• BinningHistoryDir (in string)
	• TimeoutBin (in integer)
	• IllAltBin (in integer) : illegal alt bin
	• IllBinAction : illegal bin action
	o Abort
	o AltBin
	• MinValidAltBin (in integer)
	• MaxValidAltBin (in integer)
	See Table 2-5 for additional descriptive information for SetNewBinConfig.
	NOTE: Below are additional considerations for using `SetNewBinConfig` and `SetNewBin`.
	• Bin ID "-1" is special for the Equipment Driver which is loaded with this BinControl function. If this number is set as AltBin or IllAltBin, the process in the driver changes to "Abort."
	• If Bin ID "-1" is dedicated into either or both `MinValidAltBin` / `MaxValidAltBin`, the illegal bins checking process is skipped. The specified Bin ID with `SetNewBin` command evaluates in the driver process.

	• Logical inconsistencies about configuration by `SetNewBinConfig` command are not evaluated. **`SiteActivityControl Set`** • Variable name and value **`SiteActivityControl Config`** • Enabled o 0 o 1 • Timeout (in seconds) • TimeoutAction o Abort o Ignore • Sync o 0 o 1 • StatusQuery o None o Lot o Device *See Table 2-6 for additional descriptive information for SiteActivityControl Config.*

`string reason;`

Description	The reason for sending this command to ACS Nexus. The reason will be displayed in the ACS Nexus GUI.

Table 2-5. SetNewBinConfig Descriptions

Item	Description
Enabled	Defines whether the Bin Control feature is enabled or not enabled.
Timeout	Defines timeout value in seconds.
TimeoutAction	Defines the action caused by timeout.
TimeoutBin	Defines the special Bin which is sent when timeout is detected.
BinningHistoryDir	Defines the directory path for binning history file.
IllBinAction	Defines the action caused by illegal bins specified.
IllAltBin	Defines the special Bin where the specified illegal bins are sent.
MinValidAltBin	Defines the lower bin code of the enabled range.
MaxValidAltBin	Defines the higher bin code of the enabled range.

Table 2-6. SiteActivityControl Config Descriptions

Item	Description
Enabled	Defines whether the test site is activated or deactivated. 1 means activated.
Timeout	Defines timeout value in seconds.
TimeoutAction	Defines the action to take if timeout occurs. The actions to take are: • Abort - aborts Lot • Ignore - continues with the current setting
Sync	Defines whether synchronous mode or asynchronous mode is enabled. 0 means asynchronous mode is enabled. If asynchronous mode is set, the Timeout configuration will be ignored.
StatusQuery	Defines checking the difference between the driver status and handler status regarding the activated site. Values include: • NONE – never synchronized with handler setting • LOT – only lot start block • DEVICE – every touchdown (default setting)

2.3.3.6 class oneapi::AppInfo

This class describes the application information which is filled in by the developer and passed in as an argument when calling oneapi::Interface::connect. It will be used as the identification of this OneAPI client in the interaction with ACS Nexus. For example, when ACS Nexus receives a control command, it will know who sent the command.

Member

```
string name;
```

Description	The name of this application.

```
string vendor;
```

Description	The vendor of this application.

```
string version;
```

Description	The version of this application.

2.3.3.7 class oneapi::QueryResponse

This class describes the return information when calling the DFF data reading interfaces.

Member

```
int code;
```

Description	Error code for calling the interface. • 0: invoke success • 1: common error • 2: connection error

```
string result;
```

Description	Result for calling the interface. • jobID • status: COMPLETE, RUNNING, FAILED • DFF data string

```
string errmsg;
```

Description	Detail error information when calling the interface. The error message is null if invoke interface is successful.

2.3.3.8 class oneapi::DFFData

This is a set of static functions which are used for Data Reading of DFF. During the entire lifecycle of the application, no object of this class needs to be generated.

Member

```
static const int SUCCEED;
```

Description	The return code of query interfaces. Value is 0.

```
string vendor;
```

Description	The return code of query interfaces. Value is 1.

Functions

```
static oneapi::QueryResponse createQueryRequest(const std::string& lotID, const
std::string& deviceIDKey, const std::string& deviceIDVal, const std::string&
testNumber);
```

Description	Create the query request to the ACS Unified Server.
Parameter	Input: • Lot ID • DeviceIDKey o "STDF.PART_TXT" o "STDF.PART_ID" • deviceIDVal: The value corresponding to deviceIDKey • Test number **NOTE:** When the `deviceIDKey` is any other value except `STDF.PART_TXT” and “STDF.PART_ID`, the device id will be part_id.
Return	QueryResponse res res.result: The value express jobID.

```
static oneapi::QueryResponse createRawQueryRequest(const std::string& sql);
```

Description	Create the raw query request to ACS Unified Server.
Parameter	Input: • SQL statement **NOTE**: Make sure to enter a valid SQL query.
Return	QueryResponse res res.result: The value express jobID.

```
static oneapi::QueryResponse getQueryTaskStatus(const std::string& jobID);
```

Description	Get the status of the query task.
Parameter	Input: • Lot ID
Return	QueryResponse res res.result: The value express status. • “COMPLETE” • “FAILED” • “RUNNING” **NOTE:** When the result is *RUNNING*, you need to invoke this interface again after a period of time until the result is *COMPLETE* or *FAILED*.

```
static oneapi::QueryResponse getQueryResult(const std::string& jobID, const std::string&
format);
```

Description	Get the query result (DFFData).
Parameter	Input: • jobID • format o CSV o JSON **NOTE:** If the `format` is any other value except *CSV* or *JSON*, the data format will be JSON.
Return	QueryResponse res res.result: The value express DFF data string.

2.3.3.9 class oneapi::TestCell

This class describes the information of the Tester (which is provided by oneapi::Monitor callback functions) and is used as the identification of the data source when consuming Nexus data. For example, when data is received, ACS Nexus can know which Tester sent the data.

Member

`String testerId;`

Description	The hostname of the tester.

`string testerIP;`

Description	The IP of the tester.

2.3.4 Configuration

As a dynamic library, OneAPI will statically read the contents of the environment variable for initialization when the application process is started. The environment variable contents are described in Table 2-7 below.

Table 2-7. OneAPI Environment Variable Description

Item	Description
`ONEAPI_DEBUG`	Developers must include and call the interface provided by OneAPI in their application. OneAPI will record status and information when executing these commands. This environment variable is used to set the output mode of the records. Values: 0 -- no output 1 -- enable output INFO level logs to console 2 -- enable output INFO level logs to file 3 -- enable output INFO level logs to both console and file 4 -- enable output DEBUG level logs to console 5 -- enable output DEBUG level logs to file 6 -- enable output DEBUG level logs to both console and file **NOTE:** The path of local log file is fixed to `${home}/.log/`. The name follows the rule indicated below: `ACS_ONEAPI_yyyy-mm-dd.log.` (for example, `ACS_ONEAPI_2023-02-21.log`) To write log files successfully, ensure that the running environment of the application meets one of the following conditions: • `${home}/.log/` folder does not exist, and the application has permission to create it. • `${home}/.log/` folder exists, and the application has permission to write log file in it.

2.4 OneAPI Python SDK

OneAPI is the standard bi-directional communication interface that enable containerized/non-containerized applications to consume real time data from (and send control command to) ACS Nexus. An application developer can use OneAPI to develop an application to consume the real time data and send control instructions during production testing. ACS Nexus invokes a callback function of the containerized application per event.

In this version of OneAPI, only C++ SDK and Python SDK are supported. This section describes Python SDK usage.

2.4.1 General Information

Package Contents

OneAPI Python SDK provides programming interfaces for the user to develop applications. All contents (see below) are packaged in a tar.gz file. Note that the library Python 3.9, 3.10, and 3.11 library files are in different directories.

```
oneAPI_py3.9  (oneAPI_py3.10 and oneAPI_py3.11 have the same file structure)
|-- bin
|    |-- AdvantestLogging.py
|    |-- oneapi_DFF.py
|    |-- oneapi.py
|    |-- liboneAPI.so
|    |-- main.py
|    |-- requirements.txt
|    |-- sample.py
|-- build_base_image.sh
|-- centos.Dockerfile.example  (for py3.9 only)
|-- examples
|    |-- main.py
|    |-- sample.py
|-- ubuntu.Dockerfile.example
```

Table 2-8. OneAPI Python Package Content Descriptions

Content	Description
`bin`	This folder contains all library files provided by OneAPI for Python. Developers need to link these files and include them in the application release-package or image.
`examples`	This folder includes example code that demonstrates the use of OneAPI to developers.
`build_base_image.sh`	This script provides a reference for users on how to build an image with Dockerfile.
`centos.Dockerfile.example` `ubuntu.Dockerfile.example`	**NOTE:** centos.Dockerfile.example is for python 3.9 only. These files provide a reference for users on how to build a OneAPI Application image based on the base image. Users need to rewrite Dockerfile according to their actual situation. Refer to the comments in this file for specific rewriting methods.

Environment Requirements

Users can develop containerized applications running on the ACS Edge Server or containerized/non-containerized applications running on a test floor server that is based on OneAPI Python SDK. For both scenarios, specific environment conditions are required, as noted below.

Application Development Environment	
Python 3.9 SDK	
Supported OS (any of)	• Red Hat 7 • CentOS 7 • Ubuntu 22.04
Necessary Software	• Python 3.9 • Python module: jsonschema
Python 3.10 SDK	
Supported OS	• Ubuntu 22.04
Necessary Software	• Python 3.10 • Python module: jsonschema
Python 3.11 SDK	
Supported OS	Ubuntu 22.04
Necessary Software	• Python 3.11 • Python module: jsonschema

Application Operating Environment
A successful connection at least needs to ensure that the network between Application Operating environment and ACS Nexus operation environment is enabled, and other dependent guarantees, such as ports availability, firewall permission, etc.

2.4.2 Usage Scenario

Basic and Advanced Test Cell Control

Table 2-9. Basic and Advanced Control Commands

Command	Description
`PAUSE`	This is a basic control command that provides real-time pause of wafer testing or final testing when user applications identify production issues. Actions taken after a pause are determined by the user (for example, automatic actions performed by other tools or manual action by the operator).
`STOP`	This is a basic control command that provides real-time stop of wafer testing or final testing when user applications identify production issues.
`Bin Control`	Bin Control is advanced control command: • /F 1: SetNewBin parameter: New bin of each site • I/F 2: SetNewBinConfig parameter: Enabled flag, time out, etc.

Real Time Test Cell Data Collection

Comprehensive real time test cell data collection enables applications to perform data analysis in real time. Below is a list of OneAPI data types and a brief description of information that can be collected for each data type.

Table 2-10. OneAPI Data Types

Data Type	Information Collected
`PRODUCTION_LOTSTART`	Lot Start event includes the following information: • Lot start time • Lot ID, Sublot ID • Test type • Site list • Prober/Handler, LB . . .
`PRODUCTION_LOTEND`	Lot End event includes the following information: • Lot complete time • Lot ID . . .
`PRODUCTION_WAFERSTART`	Wafer Start event includes the following information: • Wafer start time • Wafer ID • Wafer layout information . . .
`PRODUCTION_WAFEREND`	Wafer End event includes the following information: • Wafer complete time • Wafer ID . . .
`PRODUCTION_TESTSTART`	Test Start event includes the following information: • Test start time • Site list
`PRODUCTION_TESTEND`	Test End event includes the following information: • Test complete time • Bin result of each site • Part ID of each site • X/Y Coordinates of each site • Test time . . .
`PRODUCTION_TESTFLOWSTART`	Test Flow Start event includes the following information: • Test flow start time • Test flow name

`PRODUCTION_TESTFLOWEND`	Test Flow End event includes the following information: • Test flow end time • Test flow name
`MEASURED_PARAMETRIC`	Parametric Test Result event includes: • Test method information: number, name, lo/hi limit, unit • Result of each site: pass/fail, measured value, etc. . . .
`MEASURED_FUNCTIONAL`	Functional Test Result event includes: • Test method information: number, name • Result of each site: pass/fail, cycle count, fail-pins etc.
`MEASURED_MULTI_PARAM`	Multi-Parametric Test Result event includes: • Test method information: number, name, lo/hi limit, unit • Multiple measured values of each site . . .
`MEASURED SCAN`	Scan Test Result event includes: • Test method info: number • Result of each Site: total cycle count, fail cycle count • Result of each Pattern: name, pins information • Result of each Pin: name, failing cycles …
`DEVICE`	Device data includes: • Test Program info: name, path, pin config file etc. • Bin table: soft-bin, hard-bin • Pin data: index, type, name, number etc. • Pattern data: index, name, pattern files etc.
`USERDEFINED`	User Defined data includes: • User specified variable name and its value
`DATALOGTEXT`	Datalog Text data includes: • Datalog text time stamp • Datalog text expression

2.4.3 Python API

This is a set of static functions which are used for communication between the user application and ACS Nexus. During the lifecycle of the application, no instance of this class needs to be generated.

2.4.3.1 class Interface

```
def connect(me: AppInfo, IP: str, cmdPort: int = 0, dataPort: int = 0) -> int
```

Description	Initiate the request to connect to ACS Nexus with the specified address. Normally, the application just needs to input `AppInfo` and the IP address, in which case OneAPI will read the port information from the configuration file `oneAPI_conf.ini`. Connection must be established before sending control commands or receiving data.
Parameter	Input: • Information of the application • Whether to enable Nexus Data Streaming and Control • Whether to enable TPService for communication with NexusTPI
Return	0 – initial connection succeeded -1 – failed to enable Nexus Data Streaming and Control -2 – failed to enable TPService -3 – unknown location -4 – not support **NOTES:** The return value 0 does not mean a connection with ACS Nexus has been established. Another interface `getConnectionState()` can provide the current connection state of the command channel. For a return value of -1, a possible reason could be failed to connect to nexus-broker. For a return value of -2, a possible reason could be TPService port is occupied. (the default TPService port is 21122). For a return value of -3, a possible reason could be OneAPI runs in an unknown location. For a return value -4, a possible reason could be OneAPI does not support the corresponding feature in the current location. For any exception case, all enabled connection will be disconnected.

```
def disconnect() -> int
```

Description	Disconnects from ACS Nexus.
Return	0 disconnect succeeded -1 disconnect failed

```
def registerMonitor(myMonitor: Monitor) -> None
```

Description	Registers the event monitor to receive real-time data through a callback function. For additional details, refer to class Monitor.
Return	0 – disconnect success -1 – disconnect fail

```
def getConnectionState(cmdChannel: int) -> None
```

Description	Get the connection status of the command channel. For command channel, the status value is 0 only when the current status is established, and communication is available. Otherwise, the status value is – 1. The application should determine whether the related API can be executed according to the connection state. For example, when sending a control command, if the status of the command channel is not 0, the command will not be sent to ACS Nexus.
Parameter	Output: The status of command channel (for sending command to ACS Nexus).

```
def sendCommand(tc: TestCell, cmd: Command ) -> int
```

Description	Send the command to the connected ACS Nexus. The return value of this interface indicates whether the action of sending command is successful. ACS Nexus will execute the command asynchronously after receiving the command.
Parameter	Input: • TestCell information (refer to class TestCell) • Command information (refer to class Command)
Return	0 – command successfully sent to ACS Nexus -1 – failed to send command to ACS Nexus -3 – unknown location -4 – not support

2.4.3.2 class Monitor

```
def consumeData(tc: TestCell, data: NexusData) -> None
```

Description	This is a pure virtual function that needs to be inherited and overridden. As a callback function, it will be triggered every time ACS Nexus data arrives. **NOTE1:** When the function is triggered, the thread will be blocked. It cannot be triggered again until the callback function completes its processing logic and returns. Therefore, attention should be given to the efficiency of processing or consideration given to implementing asynchronous processing in the code. **NOTE 2:** You must call the corresponding *get* and *query* interfaces inside this function and save the values, otherwise the data will be cleared.
Parameter	• TestCell information (refer to class TestCell) • ACS Nexus data (refer to Class class NexusData)

```
def consumeTPSend( tc: TestCell, data: str) -> None
```

Description	This is a function that needs to be inherited and overridden. As a callback function, it will be triggered every time TP sends data via NexusTPI. **NOTE:** When the function is triggered, the thread will be blocked. It cannot be triggered again until the callback function completes its processing logic and returns. Therefore, attention should be given to the efficiency of processing or consideration given to implementing asynchronous processing in the code.
Parameter	• TestCell information (refer to class TestCell) • Data type is string

```
def consumeTPRequest ( tc: TestCell, request: str) -> str
```

Description	This is a function that needs to be inherited and overridden. As a callback function, it will be triggered every time TP initiates a request via NexusTPI. **NOTE:** When the function is triggered, the thread will be blocked. It cannot be triggered again until the callback function completes its processing logic and returns. Therefore, attention should be given to the efficiency of processing or consideration given to implementing asynchronous processing in the code.
Parameter	• TestCell information (refer to class TestCell) • Request type is string in JSON format
Return	string: after handle request, return your response to NexusTPI.

2.4.3.3 enum DataType

Description	Indicate the type of ACS Nexus data. - `DATA_TYP_PRODUCTION_LOTSTART` - `DATA_TYP_PRODUCTION_WAFERSTART` - `DATA_TYP_PRODUCTION_TESTSTART` - `DATA_TYP_PRODUCTION_TESTEND` - `DATA_TYP_PRODUCTION_WAFEREND` - `DATA_TYP_PRODUCTION_LOTEND` - `DATA_TYP_MEASURED_PARAMETRIC` - `DATA_TYP_MEASURED_FUNCTIONAL` - `DATA_TYP_MEASURED_MULTI_PARAM` - `DATA_TYP_DEVICE` - `DATA_TYP_USERDEFINED` - `DATA_TYP_FILE` - `DATA_TYP_PRODUCTION_TESTFLOWSTART` - `DATA_TYP_PRODUCTION_TESTFLOWEND` - `DATA_TYP_DATALOGTEXT` - `DATA_TYP_IDENTIFICATION` - `DATA_TYP_DEVICE_PIN` - `DATA_TYP_MEASURED_SCAN` - `DATA_TYP_DEVICE_PATTERN` - `DATA_TYP_MEASUREMENT`

2.4.3.4 class NexusData

Each time the callback function consumeData is triggered, a reference to the NexusData object will be passed in as a parameter. Detailed values can be obtained by calling the member functions of this class.

NOTE: The corresponding member functions must be called according to the type of data to obtain the current valid value.

[Production Event] Start of Lot
DataType DATA_TYP_PRODUCTION_LOTSTART
Occurs: Once at the beginning of the first touchdown per LOT
Contains: All the global information for this LOT

Member Function	Return Value(s) Description
`get_Timezone() -> int32`	The time zone where ACS Nexus is running
`get_SetupTime() -> uint64`	Date and time when the test program was started (in microseconds)
`get_TimeStamp() -> uint64`	Date and time when the first part was tested (in microseconds)
`get_StationNumber() -> int32`	Tester station number
`get_ModeCode() -> str`	Test mode code
`get_RetestCode() -> str`	Lot retest code
`get_ProtectionCode() -> str`	Data protection code
`get_BurnTimeMinutes() -> int32`	Burn-in time (in minutes)
`get_CommandCode() -> str`	Command mode code of the tester
`get_LotId() -> str`	Lot ID
`get_PartType() -> str`	Part type or product ID
`get_NodeName() -> str`	Hostname of the tester system controller
`get_TesterType() -> str`	Tester type
`get_JobName() -> str`	Test program name
`get_JobRevision() -> str`	Test program revision number
`get_SublotId() -> str`	Sublot ID
`get_OperatorName() -> str`	Operator name or ID at setup time
`get_TesterosType() -> str`	Tester software type
`get_TesterosVersion() -> str`	Tester software version number
`get_TestType() -> str`	Type of Lot (PACKAGE_TEST / WAFER_TEST)
`get_TestStepCode() -> str`	Test phase or step code
`get_TestTemperature() -> str`	Test temperature
`get_UserText() -> str`	User-defined text
`get_AuxiliaryFile() -> str`	Name of auxiliary data file
`get_PackageType() -> str`	Package type
`get_FamilyId() -> str`	Product family ID
`get_DateCode() -> str`	Date code
`get_FacilityId() -> str`	Test facility ID
`get_FloorId() -> str`	Test floor ID

`get_ProcessId() -> str`	Fabrication process ID
`get_OperationFreq() -> str`	Operation frequency or step
`get_SpecName() -> str`	Test specification name
`get_SpecVersion() -> str`	Test specification version number
`get_FlowId() -> str`	Testflow ID
`get_SetupId() -> str`	Test setup ID
`get_DesignRevision() -> str`	Device design revision
`get_EngineeringLotId() -> str`	Engineering lot ID
`get_RomCode() -> str`	ROM code ID
`get_SerialNumber() -> str`	Tester serial number
`get_SupervisorName() -> str`	Supervisor name or ID
`get_HeadNumber() -> uint32`	Test head number
`get_SiteGroupNumber() -> uint32`	Site group number (station number)
`get_SiteCount() -> uint32`	Number of active sites described in this record
`get_TotalHeadSiteList() -> list`	Array of test site numbers with head information
`get_ProberHandlerType() -> str`	Handler or prober type
`get_ProberHandlerId() -> str`	Handler or prober ID
`get_ProbecardType() -> str`	Probe card type
`get_ProbecardId() -> str`	Probe card ID
`get_LoadboardType() -> str`	DUT board type
`get_LoadboardId() -> str`	DUT board ID
`get_DibType() -> str`	DIB board type
`get_DibId() -> str`	DIB board ID
`get_CableType() -> str`	Interface cable type
`get_CableId() -> str`	Interface cable ID
`get_ContactorType() -> str`	Handler contactor type
`get_ContactorId() -> str`	Handler contactor ID
`get_LaserType() -> str`	Laser type
`get_LaserId() -> str`	Laser ID
`get_ExtraEquipType() -> str`	Extra equipment type
`get_ExtraEquipId() -> str`	Extra equipment ID

[Production Event] End of Lot
DataType DATA_TYP_PRODUCTION_LOTEND
Occurs: Once at the ending of the last touchdown per LOT
Contains: Information that compliments the Lot

Member Function	Return Value Description
`get_TimeStamp() -> uint64`	Date and time last part of the lot was tested (in microseconds)
`get_DisPositionCode() -> str`	Lot disposition code
`get_UserDescription() -> str`	Lot description supplied by the user
`get_ExecDescription() -> str`	Lot description supplied by the exec

[Production Event] Start of Wafer
DataType DATA_TYP_PRODUCTION_WAFERSTART
Occurs: Once at the beginning of the first touchdown per WAFER
Contains: All the configuration information for the tested wafer

Member Function	Return Value Description
`get_WaferSize() -> float`	Diameter of wafer in WF_UNITS
`get_DieHeight() -> float`	Height of die in WF_UNITS
`get_DieWidth() -> float`	Width of die in WF_UNITS
`get_WaferUnits() -> uint32`	Unit for wafer and die dimensions
`get_WaferFlat() -> str`	Orientation of wafer flat
`get_CenterX() -> int32`	X-coordinate of center die on wafer
`get_CenterY() -> int32`	Y-coordinate of center die on wafer
`get_PositiveX() -> str`	Positive X-direction of wafer
`get_PositiveY() -> str`	Positive Y-direction of wafer
`get_HeadNumber() -> uint32`	Test head number
`get_SiteGroupNumber() -> uint32`	Site group number
`get_TimeStamp() -> uint64`	Date and time when the first part of wafer was tested (in microseconds)
`get_WaferId() -> str`	Wafer ID

[Production Event] End of Wafer

DataType DATA_TYP_PRODUCTION_WAFEREND

Occurs: Once at the ending of the last touchdown per WAFER
Contains: The result information for the tested wafer

Member Function	Return Value Description
`get_HeadNumber() -> uint32`	Test head number
`get_SiteGroupNumber() -> uint32`	Site group number
`get_TimeStamp() -> uint64`	Date and time when the last part of wafer was tested (in microseconds)
`get_TestedCount() -> uint32`	Number of tested parts
`get_RetestedCount() -> uint32`	Number of retested parts
`get_GoodCount()) -> uint32`	Number of tested parts that have passed
`get_WaferId() -> str`	Wafer ID
`get_UserDescription() -> str`	Wafer description supplied by user
`get_ExecDescription() -> str`	Wafer description supplied by exec

[Production Event] End of Test

DataType DATA_TYP_PRODUCTION_TESTEND

Occurs: Once at the ending of each touchdown
Contains: Site, part, and result information for this touchdown

Member Function	Return Value Description
`get_TimeStamp() -> uint64`	Date and time when the part completes test (in microseconds)
`get_ResultCount() -> uint32`	Count of test result
`query_HeadSite(uint32 index) -> uint32`	Test site number with head information
`query_PartFlag(uint32 index) -> uint32`	Part information flags
`query_SBinResult(uint32 index) -> uint32`	Software bin
`query_HBinResult(uint32 index) -> uint32`	Hardware bin
`query_XCoord(uint32 index) -> int32`	Wafer X-coordinate
`query_YCoord(uint32 index) -> int32`	Wafer Y-coordinate
`query_TestTime(uint32 index) -> uint32`	Elapsed test time in microseconds
`query_PartId(uint32 index) -> str`	Part identification
`query_PartText(uint32 index) -> str`	Part description text

[Production Event] Start of Test

DataType DATA_TYP_PRODUCTION_TESTSTART

Occurs: Once at the beginning of each touchdown
Contains: Site information for this touchdown

Member Function	Return Value Description
`get_TimeStamp() -> uint64`	Date and time when the part starts to test (in microseconds)
`get_HeadSiteList() -> list`	Array of test site number with head information
`query_HeadSite(index: uint32) -> uint32`	Test site number with head information
`query_XCoord(index: uint32) -> int32`	Wafer X-coordinate
`query_YCoord(index: uint32) -> int32`	Wafer Y-coordinate

[Production Event] Start of Test Flow

DataType DATA_TYP_PRODUCTION_TESTFLOWSTART

Occurs: Once at the beginning of each test flow
Contains: Test flow name

Member Function	Return Value Description
`get_TimeStamp() -> uint64`	Date and time when the test flow starts (in microseconds)
`get_TestFlowName() -> str`	Test flow name

[Production Event] End of Test Flow

DataType DATA_TYP_PRODUCTION_TESTFLOWEND

Occurs: Once at the ending of each test flow
Contains: Test flow name

Member Function	Return Value Description
`get_TimeStamp() -> uint64`	Date and time when the test flow completes (in microseconds)
`get_TestFlowName() -> str`	Test flow name

[Measured Value] Parametric Test Result

DataType DATA_TYP_MEASURED_PARAMETRIC

Occurs: Once per parametric test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`get_ResultCount() -> uint32`	Count of result for the test item
`query_HeadSite(uint32 index) -> uint32`	Test site number with head information
`query_TestNumber(uint32 index) -> uint32`	Test number
`query_TestText(uint32 index) -> str`	Test description text or label[1]
`query_LowLimit(uint32 index) -> float`	Low test limit value
`query_HighLimit(uint32 index) -> float`	High test limit value
`query_Unit(uint32 index) -> str`	Test units
`query_TestFlag(uint32 index) -> str`	Test flags (fail, alarm, etc.)
`query_Result(uint32 index) -> float`	Test result
`query_TestSuite(uint32 index) -> str`	Test suite name
`query_MeasurementName(index: uint32) -> str`	The unique fully qualified name of measurement

[1] In SmarTest 7, the priority is the test comment, followed by "Test suite name: test name." If there is a pin name, it will be "Test suite name: test name : pin name." In SmarTest 8, it will be always the test descriptor.

[Measured Value] Functional Test Result

DataType DATA_TYP_MEASURED_FUNCTIONAL

Occurs: Once per functional test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`get_ResultCount() -> uint32`	Count of result for the test item
`query_HeadSite(uint32 index) -> uint32`	Test site number with head information
`query_TestNumber(uint32 index) -> uint32`	Test number
`query_TestText(uint32 index) -> str`	Test description text or label
`query_TestFlag(uint32 index) -> str`	Test flags (fail, alarm, etc.)
`query_CycleCount(uint32 index) -> uint32`	Vector cycle count
`query_NumberFail(uint32 index) -> uint32`	The number of logic pin names with one or more failures
`query_PinResults(index: uint64) -> list`	List of fail Pin index which belong to the given Result index
`query_PinName(pinID: uint64) -> str`	Pin name
`query_FailCycles(pinID: uint64) -> list`	List of fail cycle numbers which belong to the given Pin
`query_VectNam(index: uint32) -> str`	Vector module pattern name
`query_TestSuite(uint32 index) -> str`	Test suite name
`query_MeasurementName(index: uint32) -> str`	The unique fully qualified name of measurement

[Measured Value] Multiple-Result Parametric Test Record

DataType DATA_TYP_MEASURED_MULTI_PARAM

Occurs: Once per multiple-result parametric test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`get_ResultCount() -> uint32`	Count of result for the test item
`query_HeadSite(uint32 index) -> uint32`	Test site number with head information
`query_TestNumber(uint32 index) -> uint32`	Test number
`query_TestText(uint32 index) -> str`	Test description text or label
`query_LowLimit(uint32 index) -> float`	Low test limit value
`query_HighLimit(uint32 index) -> float`	High test limit value
`query_Unit(uint32 index) -> str`	Test units
`query_TestFlag(uint32 index) -> str`	Test flags (fail, alarm, etc.)
`query_Results(uint32 index) -> list`	List of Test results which belong to the given Result index
`query_PinResults(index: uint64) -> list`	List of Pin index which belong to the given Result index
`query_PinName(pinID: uint64) -> str`	Pin name
`query_TestSuite(uint32 index) -> str`	Test suite name
`query_MeasurementName(index: uint32) -> str`	The unique fully qualified name of measurement

[Notify] Scan Test Result

DataType DATA_TYP_MEASURED_SCAN

Occurs: Once per scan test item during the test of each touchdown
Contains: Results information for all tested sites, and basic information of the test item

Member Function	Return Value Description
`get_ResultCount() -> uint32`	Count of result for the test item
`query_HeadSite(index: uint32) -> uint32`	Test site number with head information
`query_TestNumber(index: uint32) -> uint32`	Test number
`query_TestText(index: uint32) -> str`	Test description text or label
`query_TestFlag(index: uint32) -> str`	Test flags (fail, alarm, etc.)
`query_TotalCycleCount(index: uint32) -> int32`	Total number of cycles
`query_FailCycleCount(index: uint32) -> int32`	Total number of failing cycles
`query_OpSequence(index: uint32) -> str`	Operating sequence
`query_PatternResults(index: uint32) -> list`	List of pattern index which belong to the given Result
`query_PatternName(patternID: uint64) -> str`	Pattern name
`query_PinResults(index: uint64) -> list`	List of pin index which belong to the given Pattern
`query_PinName(pinID: uint64) -> str`	Pin name

`query_FailCycles(pinID: uint64) -> list`	List of fail cycle numbers which belong to the given Pin
`query_TestSuite(uint32 index) -> str`	Test suite name
`query_MeasurementName(index: uint32) -> str`	The unique fully qualified name of measurement

[Device Data]

DataType DATA_TYP_DEVICE

Occurs: Once for each lot, when all the required data collection is completed
Contains: Device information

Member Function	Return Value Description
`get_TestProgramDir() -> str`	The absolute path of the folder of the active test program
`get_TestProgramName() -> str`	The fully qualified name of the active test program
`get_TestProgramPath()) -> str`	The absolute path of the active test program file
`get_PinConfig() -> str`	The fully qualified name of the active DUT board description file
`get_ChannelAttribute() -> str`	The fully qualified name of the active Channel attributes file
`get_BinInfoCount() -> uint32`	Count of defined soft bin of the active test program
`query_SBinNumber(uint32 index) -> uint32`	The number of the soft bin
`query_SBinName(uint32 index) -> str`	The description of the soft bin
`query_SBinType(uint32 index) -> uint32`	The type of the soft bin
`query_HBinNumber(uint32 index) -> uint32`	The corresponding hard bin number
`query_HBinName(uint32 index) -> str`	The description of the corresponding hard bin
`query_HBinType(uint32 index) -> uint32`	The type of the corresponding hard bin

[Notify] Pin Data

DataType DATA_TYP_DEVICE_PIN

Occurs: Once for each lot, when all the required data collection is completed.
Contains: Pin information

Member Function	Return Value Description
`get_PinInfoCount() -> uint32`	The count of pin information
`query_PinIndex(index: uint32) -> uint32`	Pin index
`query_ChannelType(index: uint32) -> uint32`	Channel type
`query_ChannelName(index: uint32) -> str`	Channel name
`query_PhysicalName(index: uint32) -> str`	Name of physical pin
`query_LogicalName(index: uint32) -> str`	Name of logical pin
`query_HeadNumber(index: uint32) -> str`	Test head number
`query_SiteNumber(index: uint32) -> uint32`	Test site number

[Notify] Pattern Data
DataType DATA_TYP_PATTERN
Occurs: Whenever a pattern variable is collected.
Contains: Information on a pattern profile for a specific scan test.

Member Function	Return Value Description
`get_PatternCount() -> uint32`	The count of patterns
`query_OpSequence(index: uint32) -> str`	Operating sequence
`query_PatternLabels(index: uint32) -> list`	List of pattern labels

[User Defined Data]
DataType DATA_TYP_USERDEFINED
Occurs: Whenever the user-defined variable is collected
Contains: Variable and its value information

Member Function	Return Value Description
`get_UserDefined() -> dict`	Map of user defined data <variable, value>

[Notify] File Transfer
DataType DATA_TYP_FILE
Occurs: When each file transfer is completed
Contains: The file name

Member Function	Return Value Description
`get_FileName() -> str`	The absolute path of the received file

[Notify] Datalog Text
oneapi::DataType DATA_TYP_DATALOGTEXT
Occurs: Whenever a datalog-text variable is collected
Contains: The expression of datalog-text

Member Function	Return Value Description
`get_TimeStamp() -> uint64`	Date and time when the test flow completes (in microseconds)
`get_DataLogText() -> str`	The expression of datalog text

[Notify] Measurement Data **oneapi::DataType** DATA_TYP_MEASUREMENT **Occurs:** Once a measurement has ended **Contains:** The information of measurement	
Member Function	**Return Value Description**
`get_MeasurementName() -> str`	The unique fully qualified name of measurement
`get_GroupCount() -> uint32`	Total count of parallel groups
`query_GroupName(index: uint32) -> str`	The name of the parallel group
`query_GroupBypassed(index: uint32) -> int32`	Whether the parallel group was bypassed
`query_GroupSites(index: uint32) -> list`	The number of the site(s) where the parallel group was bypassed
`get_SequenceCount() -> uint32`	Total count of the operating sequence
`query_SequenceName(index: uint32) -> str`	The name of the operating sequence
`query_SequenceBypassed(index: uint32) -> int32`	Whether the operating sequence was bypassed
`query_SequenceSites(index: uint32) -> list`	The number of site where the operating sequence was bypassed
`query_SequenceGroupResults(index: uint32) -> list`	list of unique "GroupID" for the specific sequence
`query_SequenceGroupName(groupID: uint32) -> str`	The name of the parallel group that belongs to current sequence
`query_SequenceGroupBypassed(groupID: uint32) -> int32`	Whether the parallel group was bypassed
`query_SequenceGroupSites(groupID: uint32) -> list`	The number of the site where the parallel group was bypassed

2.4.3.4.1 Function Introduction

`getType() -> DataType`

Description	Get the type of data currently consumed. **NOTE:** This function must be called before all other functions in the consumeData code.
Return	One of the following: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_TESTSTART` • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_PRODUCTION_WAFEREND` • `DATA_TYP_PRODUCTION_LOTEND` • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_DEVICE` • `DATA_TYP_USERDEFINED` • `DATA_TYP_FILE` • `DATA_TYP_PRODUCTION_TESTFLOWSTART` • `DATA_TYP_PRODUCTION_TESTFLOWEND` • `DATA_TYP_DATALOGTEXT` • `DATA_TYP_IDENTIFICATION` • `DATA_TYP_DEVICE_PIN` • `DATA_TYP_MEASURED_SCAN` • `DATA_TYP_DEVICE_PATTERN`

`get_TimeStamp() -> uint64`

Description	Get the date and time in microseconds. Valid for: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_LOTEND` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_WAFEREND` • `DATA_TYP_PRODUCTION_TESTSTART` • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_PRODUCTION_TESTFLOWSTART` • `DATA_TYP_PRODUCTION_TESTFLOWEND`
Return	If by one of the above events, returns a valid value. For example, if event is `DATA_TYP_PRODUCTION_LOTSTART`, then the value represents the timestamp of `Lot_Start`. Otherwise, returns 0.

`get_TestProgramDir() -> str`

Description	Get the absolute path of the folder of the active test program. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`get_TestProgramName() -> str`

Description	Get the fully qualified name of the active test program. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`get_TestProgramPath() -> str`

Description	Get the absolute path of the active test program file. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`get_PinConfig() -> str`

Description	Get the fully qualified name of the active DUT board description file. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`get_ChannelAttribute() -> str`

Description	Get the fully qualified name of the active Channel attributes file. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`get_BinInfoCount() -> uint32`

Description	Get the count of defined soft bin of the active test program. Valid always, refreshed by `DATA_TYP_DEVICE`.
Return	Valid value.

`query_SBinNumber(index: uint32) -> uint32`

Description	Query the number of the soft bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

`query_SBinName(index: uint32) -> str`

Description	Query the description of the soft bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`query_SBinType(index: uint32) -> uint32`

Description	Query the type of soft bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	If valid input index, returns: 0 – PASS 1 – FAIL 2 – UNKONWN If input index is out of bounds, returns 2.

`query_HBinNumber(index: uint32) -> uint32`

Description	Query the number of hard bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

`query_HBinName(index: uint32) -> str`

Description	Query the description of the hard bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`query_HBinType(index: uint32) -> uint32`

Description	Query the type of hard bin. Valid always, refreshed by `DATA_TYP_DEVICE`.
Parameter	Input: Index number
Return	If valid input index, returns: 0 – PASS 1 – FAIL 2 – UNKONWN If input index is out of bounds, returns 2.

`get_PinInfoCount() -> uint32`

Description	Get the total number of pin information. Valid for `DATA_TYP_DEVICE_PIN`.
Return	Returns a valid value if by above event. Otherwise, returns 0.

`query_ChannelType(index: uint32) -> uint32`

Description	Query the channel type. Valid for `DATA_TYP_DEVICE_PIN`.
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_ChannelName(index: uint32) -> str`

Description	Query the channel name. Valid for `DATA_TYP_DEVICE_PIN`.
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`query_PhysicalName(index: uint32) -> str`

Description	Query the name of the physical pin. Valid for `DATA_TYP_DEVICE_PIN`.
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`query_LogicalName(index: uint32) -> str`

Description	Query the name of the logical pin. Valid for `DATA_TYP_DEVICE_PIN`.
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

query_HeadNumber(index: uint32) -> uint32

Description	Query the test head number. Valid for DATA_TYP_DEVICE_PIN.
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

query_SiteNumber(index: uint32) -> uint32

Description	Query the test site number. Valid for DATA_TYP_DEVICE_PIN.
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

get_PatternCount() -> uint32

Description	Get the count of patterns. Valid always, refreshed by DATA_TYP_DEVICE_PATTERN.
Return	Returns a valid value in all cases.

query_OpSequence(uint32 index) -> str

Description	Query the operating sequence. Valid always, refreshed by DATA_TYP_DEVICE_PATTERN & DATA_TYP_MEASURED_SCAN.
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

query_PatternLables(uint32 index) -> list

Description	Query the list of pattern labels. Valid for DATA_TYP_DEVICE_PATTERN.
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`get_UserDefined() -> dict`

Description	Get the user defined key and value pairs. Valid for `DATA_TYP_USERDEFINED`.
Return	Returns valid value if by above event, otherwise returns map with no elements.

`get_FileName() -> str`

Description	Get the absolute path of the received file. The file is in the 'home' folder of the current user. Valid always, refreshed by `DATA_TYP_FILE`.
Return	Returns valid value if by above event, otherwise returns "" (empty string).

`get_Timezone() -> int32`

Description	Get the time zone where ACS Nexus is running (value in integer). Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`get_SetupTime() -> uint64`

Description	Get the date and time (in microseconds) when the test program was started. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`get_StationNumber() -> uint32`

Description	Get the tester station number (value in integer). Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`get_ModeCode() -> str`

Description	Get the test mode code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_RetestCode() -> str`

Description	Get the lot retest code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_ProtectionCode() -> str`

Description	Get the data protection code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_BurnTimeMinutes() -> uint32`

Description	Get the burn-in time (in minutes). Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`get_CommandCode() -> str`

Description	Get the command mode code of the tester. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns " (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_LotId() -> str`

Description	Get the lot ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_PartType() -> str`

Description	Get the part type or product ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_NodeName() -> str`

Description	Get the hostname of the tester system controller. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_TesterType() -> str`

Description	Get the tester type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_JobName() -> str`

Description	Get the test program name. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_JobRevision() -> str`

Description	Get the test program revision number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_SublotId() -> str`

Description	Get the sublot ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_OperatorName() -> str`

Description	Get the operator name or ID at setup time. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_TesterosType() -> str`

Description	Get the software type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_TesterosVersion() -> str`

Description	Get the tester software version number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_TestType() -> str`

Description	Get the type of lot. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value (PACKAGE_TEST or WAFER_TEST) in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_TestStepCode() -> str`

Description	Get the test phase or step code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_TestTemperature() -> str`

Description	Get the test temperature. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_UserText() -> str`

Description	Get the user defined text. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_AuxiliaryFile() -> str`

Description	Get the name of the auxiliary data file. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_PackageType() -> str`

Description	Get the package type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_FamilyId() -> str`

Description	Get the product family ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_DateCode() -> str`

Description	Get the date code. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_FacilityId() -> str

Description	Get the test facility ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_FloorId() -> str

Description	Get the test floor ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_ProcessId() -> str

Description	Get the fabrication process ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_OperationFreq() -> str`

Description	Get the operation frequency or step. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_SpecName() -> str`

Description	Get the test specification name. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_SpecVersion() -> str`

Description	Get the specification version number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_FlowId() -> str`

Description	Get the test flow ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_SetupId() -> str`

Description	Get the test setup ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_DesignRevision() -> str`

Description	Get the device design revision. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_EngineeringLotId() -> str`

Description	Get the engineering lot ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_RomCode() -> str`

Description	Get the ROM code ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_SerialNumber() -> str`

Description	Get the tester serial number. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_SupervisorName() -> str`

Description	Get the supervisor name or ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_HeadNumber() -> uint32`

Description	Get the test head number. Valid by: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_WAFEREND`
Return	In the case of Lot Start, returned value represents the test head number of the lot. In the case of Wafer Start or Wafer End, returned value represents test head number of the wafer. Otherwise, returns 0.

`get_SiteGroupNumber() -> uint32`

Description	Get the site group number. Valid by: • `DATA_TYP_PRODUCTION_LOTSTART` • `DATA_TYP_PRODUCTION_WAFERSTART` • `DATA_TYP_PRODUCTION_WAFEREND`
Return	In the case of Lot Start, returned value represents the site group number of the lot. In the case of Wafer Start or Wafer End, returned value represents site group number of the wafer. Otherwise, returns 0.

`get_SiteCount() -> uint32`

Description	Get the number of active sites described in this record. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns 0 in the case of: • Device Data • User Defined Data • File Transfer

`get_TotalHeadSiteList() -> list`

Description	Get all test site numbers. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns no element in the case of: • Device Data • User Defined Data • File Transfer

`get_ProberHandlerType() -> str`

Description	Get handler or prober type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_ProberHandlerId() -> str

Description	Get the handler or prober ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_ProbecardType() -> str

Description	Get the probe card type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_ProbecardId() -> str

Description	Get the probe card ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_LoadboardType() -> str`

Description	Get the DUT board type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_LoadboardId() -> str`

Description	Get the DUT board ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_DibType() -> str`

Description	Get the DIB board type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_DibId() -> str`

Description	Get the DIB board ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_CableType() -> str`

Description	Get the interface cable type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_CableId() -> str`

Description	Get the interface cable ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_ContactorType() -> str`

Description	Get the hander contactor type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_ContactorId() -> str`

Description	Get the hander contactor ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_LaserType() -> str`

Description	Get the laser type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_LaserId() -> str`

Description	Get the laser ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_ExtraEquipType() -> str`

Description	Get extra equipment type. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

`get_ExtraEquipId() -> str`

Description	Get extra equipment ID. Refreshed by `DATA_TYP_PRODUCTION_LOTSTART`.
Return	Returns a valid value in the case of: • Production Events • Measured Value Events Returns "" (empty string) in the case of: • Device Data • User Defined Data • File Transfer

get_DisPositionCode() -> str

Description	Get lot disposition code. Refreshed by DATA_TYP_PRODUCTION_LOTSTART.
Return	Returns a valid value in the case of the above event. Otherwise, returns "" (empty string)

get_UserDescription() -> str

Description	Get the description supplied by the user. Valid by: • DATA_TYP_PRODUCTION_LOTEND • DATA_TYP_PRODUCTION_WAFEREND
Return	In the case of Lot End, value represents the lot description supplied by the user. In the case of Wafer End, value represents the wafer description supplied by the user. Otherwise, returns "" (empty string).

get_ExecDescription() -> str

Description	Get the description supplied by the executive. Valid by: • DATA_TYP_PRODUCTION_LOTEND • DATA_TYP_PRODUCTION_WAFEREND
Return	In the case of Lot End, value represents the lot description supplied by the user. In the case of Wafer End, value represents the wafer description supplied by the user. Otherwise, returns "" (empty string).

`get_WaferSize() -> float`

Description	Get the diameter of the wafer in WF_UNITS. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0.0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_DieHeight() -> float`

Description	Get the height of the die in WF_UNITS. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0.0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_DieWidth() -> float`

Description	Get the width of the die in WF_UNITS. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0.0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_WaferUnits() -> uint32`

Description	Get the unit for wafer and die dimensions. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns 0 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_WaferFlat() -> str`

Description	Get the orientation of the wafer flat. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_CenterY() -> int32`

Description	Get the Y-coordinate of the center die on the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns -32787 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

get_CenterX() -> int32

Description	Get the X-coordinate of the center die on the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns -32787 in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

get_PositiveX() -> str

Description	Get the positive X-direction of the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

get_PositiveY() -> str

Description	Get the positive Y-direction of the wafer. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_WaferId() -> str`

Description	Get the wafer ID. Refreshed by: • `DATA_TYP_PRODUCTION_WAFERSTART`
Return	Returns a valid value in the case of: • Production Events (except for Lot Start) • Measured Value Events Returns "" (empty string) in the case of: • Lot Start • Device Data • User Defined Data • File Transfer

`get_TestedCount() -> uint32`

Description	Get the number of the tested parts. Refreshed by: • `DATA_TYP_PRODUCTION_WAFEREND`
Return	Returns a valid value in the case above, otherwise returns 0.

`get_RetestedCount() -> uint32`

Description	Get the number of the retested parts. Refreshed by: • `DATA_TYP_PRODUCTION_WAFEREND`
Return	Returns a valid value in the case above, otherwise returns 0.

`get_GoodCount() -> uint32`

Description	Get the number of the tested parts that have passed. Refreshed by: • `DATA_TYP_PRODUCTION_WAFEREND`
Return	Returns a valid value in the case above, otherwise returns 0.

`get_HeadSiteList() -> list`

Description	Get array of test site number with head information. Valid for `DATA_TYP_PRODUCTION_TESTSTART` **NOTE:** "headsite" is a new concept proposed by OneAPI. It integrates head and site information into an integer. The following methods can convert headsite to head or site: • uint32 toHead(uint32 index); • uint32 toSite(uint32 index);
Return	Returns a valid value in the case above, otherwise returns no element.

`get_ResultCount() -> uint32`

Description	Get the count of the test result**.** Valid for: • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	In the case of the above events, returns a valid value. For example, if event is DATA_TYP_PRODUCTION_TESTEND, then the value represents the count of DUTs of the current touchdown. If event is DATA_TYP_MEASURED_FUNCTIONAL, then the value represents the count of functional test result for the current test item. Otherwise, returns 0

`query_HeadSite(index: uint32) ->  uint32;`

Description	Query the test site number with head information. Valid by: • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_PRODUCTION_TESTSTART`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_PartFlag(index: uint32) ->  str`

Description	Query the part information flags. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`query_SBinResult(index: uint32) -> uint32`

Description	Query the soft bin number. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_HBinResult(index: uint32) -> uint32`

Description	Query the hard bin number. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_XCoord(index: uint32) ->  int32`

Description	Query the X-coordinate. Valid by: • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_PRODUCTION_TESTSTART`
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

`query_YCoord(index: uint32) ->  int32`

Description	Query the Y-coordinate. Valid by: • `DATA_TYP_PRODUCTION_TESTEND` • `DATA_TYP_PRODUCTION_TESTSTART`
Return	Returns a valid value if valid input index. Returns 65535 if input index is out of bounds.

`query_TestTime(index: uint32) ->  uint32`

Description	Query the elapsed test time in microseconds. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_PartId(index: uint32) ->  str`

Description	Query the part identification. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`query_PartText (index: uint32) ->  str`

Description	Query the part description text. Valid by: • `DATA_TYP_PRODUCTION_TESTEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`query_TestNumber(index: uint32) -> uint32`

Description	Get the number of current test item. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the number of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`query_TestText(index: uint32) ->  str`

Description	Query test description text or label. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test text of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`query_LowLimit(index: uint32) ->  float`

Description	Query the low test limit value. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the low limit of the current parametric test. Returns 0.0 if input index is out of bounds.

`query_HighLimit(index: uint32) -> float`

Description	Query the high test limit value. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the high limit of the current parametric test. Returns 0.0 if input index is out of bounds.

`query_Unit(index: uint32) -> str`

Description	Query the test units. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_MULTI_PARAM`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test unit of the current parametric test. Returns "" (empty string) if input index is out of bounds.

`query_TestFlag(index: uint32) -> str`

Description	Query the test flags, including fail, alarm, etc. Valid by: • `DATA_TYP_MEASURED_PARAMETRIC` • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test flags of the current parametric test. Returns "" (empty string) if input index is out of bounds.

NexusData::query_TestSuite(index: uint32) -> str

Description	Query the test suite name. Valid by: • DATA_TYP_MEASURED_PARAMETRIC • DATA_TYP_MEASURED_FUNCTIONAL • DATA_TYP_MEASURED_MULTI_PARAM • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the test suite name of the current parametric test. Returns "" (empty string) if input index is out of bounds.

NexusData::query_MeasurementName(index: uint32) -> str

Description	Query the measurement name. Valid by: • DATA_TYP_MEASURED_PARAMETRIC • DATA_TYP_MEASURED_FUNCTIONAL • DATA_TYP_MEASURED_MULTI_PARAM • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. For example, if event is DATA_TYP_MEASURED_PARAMETRIC, then the value represents the measurement name of the current parametric test. Returns "" (empty string) if input index is out of bounds.

query_Result(index: uint32) -> float

Description	Query the test result. Valid by: • DATA_TYP_MEASURED_PARAMETRIC
Return	Returns a valid value if valid input index. Returns 0.0 if input index is out of bounds.

`query_CycleCount(index: uint32) -> uint32`

Description	Query the vector cycle count. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_NumberFail(index: uint32) ->  uint32`

Description	Query the vector cycle count. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

`query_FailPins(index: uint32) ->  list`

Description	Query the list of failing pin bit field. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`query_VectNam(index: uint32) ->  str`

Description	Query the vector module pattern name. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`query_Results(index: uint32) -> list`

Description	Query the list of all test results. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`get_TestFlowName() ->  str`

Description	Get the test flow name. Valid by: • `DATA_TYP_PRODUCTION_TESTFLOWSTART` • `DATA_TYP_PRODUCTION_TESTFLOWEND`
Return	Returns a valid value if valid input index. Returns "" (empty string) if input index is out of bounds.

`get_DataLogText() ->  str`

Description	Get the data log text. Valid by: • `DATA_TYP_DATALOGTEXT`
Return	Returns a valid value in the case of the above event. Otherwise, returns "" (empty string).

`query_TotalCycleCount(index: uint32) -> int32`

Description	Query the total number of cycles. Valid by: • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

query_FailCycleCount(index: uint32) -> int32

Description	Query the total number of failing cycles. Valid by: • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Returns 0 if input index is out of bounds.

query_PatternResults(index: uint32) -> list

Description	Query the list of pattern index results. Valid by: • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

query_PatternName(index: uint32) -> str

Description	Query the pattern name. Valid by: • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Otherwise, returns "NOT EXIST".

query_PinResults(index: uint64) -> list

Description	Query the list of pin index results. Valid by: • DATA_TYP_MEASURED_FUNCTIONAL • DATA_TYP_MEASURED_MULTI_PARAM • DATA_TYP_MEASURED_SCAN
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`query_PinName(pinID: uint64) -> str`

Description	Query the pin name. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_MULTI_PARAM` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Otherwise, returns "NOT EXIST".

`query_FailCycles(pinID: uint64) -> list`

Description	Query the list of fail cycle numbers. Valid by: • `DATA_TYP_MEASURED_FUNCTIONAL` • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`get_PatternCount() -> uint32`

Description	Get the total number of patterns. Valid by: • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if by above event. Otherwise, returns 0.

`query_PatternLabels(index: uint32) -> list`

Description	Query the list of pattern names. Valid by: • `DATA_TYP_MEASURED_SCAN`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`get_MeasurementName() -> str`

Description	Get the measurement name. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if by above event. Otherwise, returns "" (empty string).

`get_GroupCount() -> uint32`

Description	Get the total count of parallel groups. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if by above event. Otherwise, returns 0.

`query_GroupName(index: uint32) -> str`

Description	Query the name of the parallel group. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`query_GroupBypassed(index: uint32) -> int32;`

Description	Query whether the parallel group was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	If valid input index, returns one of the following: • 0: bypassed is set to ‘false’ • 1: bypassed is set to ‘true’ • -1: bypassed is not set Returns -1 if input index is out of bounds.

`query_GroupSites(index: uint32) -> list`

Description	Query the number of the site where the parallel group was bypassed. Valid by: - `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element (meaning that all sites were bypassed) if input index is out of bounds.

`get_SequenceCount() -> uint32`

Description	Get the total count of operating sequence call. Valid by: - `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if by above event. Otherwise, returns 0.

`query_SequenceName(index: uint32) -> str`

Description	Query the name of the operating sequence call. Valid by: - `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`query_SequenceBypassed(index: uint32) -> int32;`

Description	Query whether the operating sequence call was bypassed. Valid by: - `DATA_TYP_MEASUREMENT`
Return	If valid input index, returns one of the following: - 0: bypassed is set to ‘false’ - 1: bypassed is set to ‘true’ - -1: bypassed is not set Returns -1 if input index is out of bounds.

`query_SeqeunceSites(index: uint32) -> list`

Description	Query the number of the site where the operating sequence call was bypassed. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element (meaning that all sites were bypassed) if input index is out of bounds.

`query_SeqeunceGroupResults(index: uint32) -> list`

Description	Query the list of unique "GroupID" for the specific sequence. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element if input index is out of bounds.

`query_SequenceGroupName(GroupID: uint32) -> str`

Description	Query the name of the parallel group for the current sequence. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Otherwise, returns "" (empty string).

`query_SequenceGroupBypassed(groupID: uint32) -> int32;`

Description	Query whether the parallel group was bypassed for the current sequence. Valid by: • `DATA_TYP_MEASUREMENT`
Return	If valid input index, returns one of the following: • 0: bypassed is set to ‘false’ • 1: bypassed is set to ‘true’ • -1: bypassed is not set Returns -1 if input index is out of bounds.

`query_SeqeunceGroupSites(groupID: uint32) -> list`

Description	Query the number of the site where the parallel group was bypassed for the current sequence. Valid by: • `DATA_TYP_MEASUREMENT`
Return	Returns a valid value if valid input index. Returns no element (meaning that all sites for the current sequence were bypassed) if input index is out of bounds.

2.4.3.5 class Command

This class is used to schedule the control capability of ACS Nexus through the Interface.sendCommand. Each time the interface is called, an object of Command needs to be created and passed into the interface as an argument, as shown in the example below

```
cmd = Command()
cmd.name = “PAUSE”
cmd.reason = “Yield is lower than 80 percent.”
tc = TestCell()
int res = Interface.sendCommand(tc, cmd)
if res != 0:
    print(f"Send command fail. code = {res}")
```

Member

`name = ""`

Description	The command name should be one of the following (case sensitive): • `PAUSE` • `STOP` • `SetNewBin` (see SetNewBin Example for details on how to use SetNewBin) • `SetNewBinConfig`

`param = ""`

Description	The necessary parameters for this command.
	This variable is string type, so all information must be assembled into a string. For example:
	`oneapi::Command cmd;`
	`cmd.name = “SetNewBinConfig”;`
	`cmd.param = “Timeout 2 TimeoutAction Abort IllBinAction AltBin IllAltBin 9”;`
	`PAUSE`
	• None
	`STOP`
	• None
	`SetNewBin`
	• Mapping of Site and Bin
	`SetNewBinConfig`
	• Enabled
	o 0
	o 1

	• Timeout (in seconds) • TimeoutAction o Abort o OriginalBin o TimeoutBin • BinningHistoryDir (in string) • TimeoutBin (in integer) • IllAltBin (in integer) : illegal alt bin • IllBinAction : illegal bin action o Abort o AltBin • MinValidAltBin (in integer) • MaxValidAltBin (in integer) *See Table 2-11 for additional descriptive information for SetNewBinConfig.* **NOTE:** Below are additional considerations for using `SetNewBinConfig` and `SetNewBin`. • Bin ID "-1" is special for the Equipment Driver which is loaded with this BinControl function. If this number is set as AltBin or IllAltBin, the process in the driver changes to "Abort." • If Bin ID "-1" is dedicated into either or both `MinValidAltBin` / `MaxValidAltBin`, the illegal bins checking process is skipped. The specified Bin ID with `SetNewBin` command evaluates in the driver process. • Logical inconsistencies about configuration by `SetNewBinConfig` command are not evaluated.

`reason = ""`

Description	The reason for sending this command to ACS Nexus. The reason will be displayed in the ACS Nexus GUI.

SetNewBin Example

At the start of the python container application, the below imports are needed from liboneAPI:

import Monitor
import DataType
import TestCell
import Command
import Interface
import toHead
import toSite

1. Setting the SetNewBin config in the container app:

```
tc = TestCell()
cmd = Command()
cmd.name = "SetNewBinConfig"
cmd.param = "Enabled 1 Timeout 1 TimeoutAction OriginalBin"
cmd.reason = f"test {cmd.name} command"
res = Interface.sendCommand(tc, cmd)
if res != 0:
    print(f"Send SetNewBinConfig command fail. code = {res}")
else:
    print(f"Send SetNewBinControl Config command: {cmd.param}")
```

2. Dummy SetNewBin set when not updating the bin. This is used to bypass the timeout time.

```
tc = TestCell()
cmd = Command()
cmd.name = "SetNewBin"
cmd.param = ""  # send bin=null for no bin change
cmd.reason = ""
res = Interface.sendCommand(tc, cmd)
if res != 0:
    print(f"Send SetNewBin command fail. code = {res}")
else:
    print("Send SetNewBin command dummy")
```

3. Set a new bin using SetNewBin:

```
tc = TestCell()
cmd = Command()
cmd.name = "SetNewBin"
cmd.param = "1:14"            # syntax is siteNum:NewBinNum
cmd.reason = f"test {cmd.name} command"
res = Interface.sendCommand(tc, cmd)
if res != 0:
    print(f"Send SetNewBin command fail. code = {res}")
else:
    print("Sending: TN=621: SetNewBin command sent: 1:14")
```

Description for SetNewBin()

- Forces to change the BIN value which is to be applied to the next BIN_DEVICE operation.
- Returns as soon as the command is accepted by Nexus (does not wait for new bin to actually be applied).
- Lot test execution is blocked until new bin information is set to Nexus by this command.

Command parameters for SetNewBin()

- params[“CMD_NAME”] : “SetNewBin”
- params[“CMD_PARAM”] : <SITE> “:” <BIN> [<SITE> “:” <BIN>]…

Description for SetNewBinConfig()

- Changes the corresponding configuration parameters.
- Is effective before the first BIN_DEVICE, after LotStart (for FT), or WaferStart (for WS).
- Resets to values in the configuration file at LotStart.

Command parameters for SetNewBinConfig()

- params[“CMD_NAME”] : “SetNewBinConfig”
- params[“CMD_PARAM”] : <ConfigParameter>
  <ConfigParameters>:
  “Enabled“ <“0” | “1”>
  “Timeout” <TimeoutValueInSec>
  “TimeoutAction” < “Abort” | “OriginalBin” | “TimeoutBin” >
  “TimeoutBin” <BinCodeInDecimal>

Table 2-11. SetNewBinConfig Descriptions

Item	Description
Enabled	Defines whether the Bin Control feature is enabled or not enabled.
Timeout	Defines timeout value in seconds.
TimeoutAction	Defines the action caused by timeout.
TimeoutBin	Defines the special Bin which is sent when timeout is detected.
BinningHistoryDir	Defines the directory path for binning history file.
IllBinAction	Defines the action caused by illegal bins specified.
IllAltBin	Defines the special Bin where the specified illegal bins are sent.
MinValidAltBin	Defines the lower bin code of the enabled range.
MaxValidAltBin	Defines the higher bin code of the enabled range.

2.4.3.6 class AppInfo

This class describes the application information which is filled in by the developer and passed in as an argument when calling Interface.connect. It will be used as the identification of this OneAPI client in the interaction with ACS Nexus. For example, when ACS Nexus receives a control command, it will know who sent the command.

Member

`name = ""`

Description	The name of this application.

`vendor = ""`

Description	The vendor of this application.

`version = ""`

Description	The version of this application.

2.4.3.7 class QueryResponse

This class describes the return information when calling the DFF data reading interfaces.

Member

`code -> int`

Description	Error code for calling the interface. • 0: invoke success • 1: common error • 2: connection error

`result -> str`

Description	Result for calling the interface. • jobID • status: COMPLETE, RUNNING, FAILED • DFF data string

`errmsg -> str`

Description	Detail error infomation when calling the interface. The error message is null if invoke interface is successful.

2.4.3.8 class DFFData

This is a set of static functions which are used for Data Reading of DFF. During the entire lifecycle of the application, no object of this class needs to be generated.

Members

`SUCCEED -> int`

Description	The return code of query interfaces. Value is 0.

`COMMON_ERROR -> int`

Description	The return code of query interfaces. Value is 1.

Functions

`def createQueryRequest(lotID: str, deviceIDKey: str, deviceIDVal: str, testNumber: str) -> QueryResponse`

Description	Create the query request to ACS Unified Server.
Parameter	Input: • Lot ID • DeviceIDKey o "STDF.PART_TXT" o "STDF.PART_ID" • deviceIDVal: The value corresponding to deviceIDKey • Test number **NOTE:** • When the `deviceIDKey` is any other value except `STDF.PART_TXT" and "STDF.PART_ID`, the device id will be part_id.
Return	QueryResponse res res.result: The value express jobID.

`def createRawQueryRequest(sql: str) -> QueryResponse`

Description	Create the raw query request to ACS Unified Server.
Parameter	Input: • SQL statement **NOTE:** Make sure to enter a valid SQL query.
Return	QueryResponse res res.result: The value express jobID.

```
def getQueryTaskStatus(jobID: str) -> QueryResponse
```

Description	Get the status of the query task.
Parameter	Input: • Lot ID
Return	QueryResponse res res.result: The value express status. • "COMPLETE" • "FAILED" • "RUNNING" **NOTE:** When the result is *RUNNING*, you need to invoke this interface again after a period of time until the result is *COMPLETE* or *FAILED*.

```
def getQueryResult(jobID: str, format: str) -> QueryResponse
```

Description	Get the query result (DFFData).
Parameter	Input: • jobID • format o CSV o JSON **NOTE:** When the `format` is any other value except *CSV* or *JSON*, the data format will be JSON.
Return	QueryResponse res res.result: The value express DFF data string.

2.4.3.9 class DFF

This class provides functions that are used for Data Writing of the Application. During the entire lifecycle of the application, no object of this class needs to be generated.

Member

Error codes -> int

<table>
<tr><td><b>Description</b></td><td>Error code for calling the interface.
<table>
<tr><th>Name</th><th>Value</th><th>Expression</th></tr>
<tr><td>SUCCEED</td><td>0</td><td>Communication succeeded</td></tr>
<tr><td>COMMON_ERROR</td><td>1</td><td>Errors that are not specifically defined</td></tr>
<tr><td>TIMEOUT</td><td>2</td><td>Communication timeout</td></tr>
<tr><td>TARGET_INVALID</td><td>3</td><td>Target is invalid (due to AppDescriptor issue)</td></tr>
<tr><td>CONNECT_FAILED</td><td>4</td><td>Unable to connect to the target App</td></tr>
<tr><td>CERTIFICATE_ERROR</td><td>5</td><td>Certificate related errors</td></tr>
<tr><td>UNKNOWN_LOCATION</td><td>6</td><td>Unknown location</td></tr>
<tr><td>NOT_SUPPORT</td><td>7</td><td>Unsupported DFF feature</td></tr>
<tr><td>SCHEMA_LOAD_FAIL</td><td>101</td><td>Schema file failed to load (possible cause is that the file is not in JSON format)</td></tr>
<tr><td>SCHEMA_SAME</td><td>102</td><td>Schema with the same $id has already been registered by this program</td></tr>
<tr><td>SCHEMA_REGIST_FAIL</td><td>103</td><td>Schema registration failed (possible cause is that the server is down)</td></tr>
<tr><td>SCHEMA_REGISTERED</td><td>104</td><td>Schema with the same $id has already been registered by others</td></tr>
<tr><td>DFF_CONNECTED</td><td>111</td><td>Properties are valid and upload started, but upload did not complete</td></tr>
<tr><td>DFF_PROPERTY_INVALID</td><td>112</td><td>Properties invalid for the schema</td></tr>
<tr><td>DFF_CONNECT_FAIL</td><td>113</td><td>Failed to establish data-upload connection</td></tr>
<tr><td>DFF_DATA_EMPTY</td><td>114</td><td>Data is empty</td></tr>
<tr><td>SCHEMA_NOT_REGISTERED</td><td>115</td><td>The process has not successfully registered the schema</td></tr>
</table>
</td></tr>
</table>

Functions

```
def importSchema(schema_path: str) -> int
```

Description	Load schema from the input JSON file and register the schema to ACS Unified Server.
Parameter	Input: - JSONschema file path
Return	Error codes: - SUCCEED - SCHEMA_LOAD_FAIL - SCHEMA_SAME - SCHEMA_REGIST_FAIL - SCHEMA_REGISTERED **NOTE**: When the return is SCHEMA_LOAD_FAIL or SCHEMA_REGIST_FAIL, try to stop code from continuing execution.

```
def set(key: str, value: str) -> type[DFF]
```

Description	Set property key-value pairs for subsequent data uploads.
Parameter	Input: - key - value (the value corresponding to the key)
Return	Method chaining is used here to implement a fluent interface, allowing multiple method calls to be chained together in a single statement.

```
def regUploadCallback(callback: (int, Dict[str, str], str) -> None) -> None
```

Description	Register user-defined callback function to receive upload result.
Parameter	Input: - A function that takes three arguments: result, headers, data **NOTE:** Make sure to define callback as a global function, not as an inner function.

```
def upload(data: str) -> int
```

Description	Asynchronously upload data to the ACS Unified Server.
Parameter	Input: • The data to be uploaded
Return	Error codes: • DFF_CONNECTED • DFF_PROPERTY_INVALID • DFF_CONNECT_FAIL • DFF_DATA_EMPTY • SCHEMA_NOT_REGISTERED

2.4.3.10 class TestCell

This class describes the information of the Tester (which is provided by oneapi::Monitor callback functions) and is used as the identification of the data source when consuming Nexus data. For example, when data is received, ACS Nexus can know which Tester sent the data.

Member

```
testerId -> str;
```

Description	The hostname of the tester.

```
testerIP -> str;
```

Description	The IP of the tester.

2.4.4 Configuration

As a dynamic library, OneAPI will statically read the contents of the environment variable for initialization when the application process is started. The contents of the environment variable are described in the table below.

Table 2-12. OneAPI Environment Variable Descriptions

Item	Description
`ONEAPI_DEBUG`	Developers must include and call the interface provided by OneAPI in their application. OneAPI will record status and information when executing these commands. This environment variable is used to set the output mode of the records. Values: 0 -- no output 1 -- enable output INFO level logs to console 2 -- enable output INFO level logs to file 3 -- enable output INFO level logs to both console and file 4 -- enable output DEBUG level logs to console 5 -- enable output DEBUG level logs to file 6 -- enable output DEBUG level logs to both console and file **NOTE:** The path of local log file is fixed to `${home}/.log/`. The name follows the rule indicated below: `ACS_ONEAPI_yyyy-mm-dd.log.` (for example, `ACS_ONEAPI_2023-02-21.log`) To write log files successfully, ensure that the running environment of the application meets one of the following conditions: • `${home}/.log/` folder does not exist, and the application has permission to create it. `${home}/.log/` folder exists, and the application has permission to write log file in it.

2.5 Supporting Containerized Application on the ACS Edge and ACS Unified Servers

ACS Nexus provides support for real-time data and control features to the containerized application running on the ACS Edge Server and ACS Unified Server, without requiring modification of the test program. Multiple containerized applications running on the ACS Edge and ACS Unified Server are supported.

The below diagram offers a general view of how ACS Nexus supports the containerized application running on the ACS Edge Server.

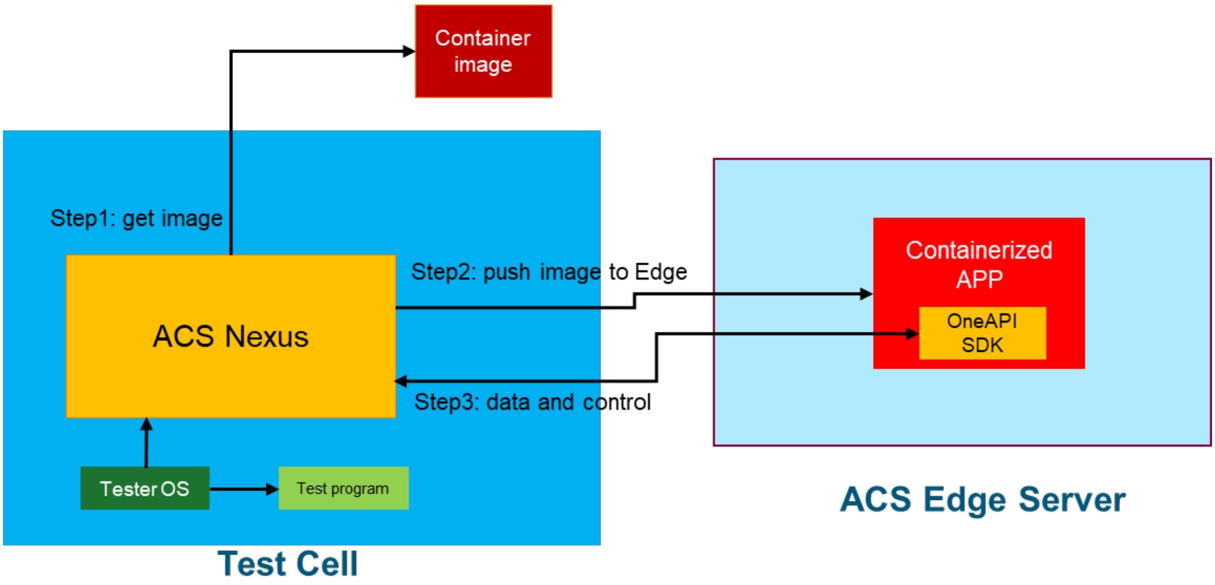


Figure 2-1. ACS Nexus Support for Containerized Application on ACS Edge Server

To support the containerized application running on the ACS Edge Server and ACS Unified, ACS Nexus must first be configured. ACS Nexus configuration is described in the subsections that follow.

2.5.1 Nexus Configuration File

The `acs_nexus.ini` configuration file is used for initialization of the ACS Nexus service. After modifying content in this file, ACS Nexus should be manually restarted for any changes to take effect (see Restarting ACS Nexus).

The path for the acs_nexus.ini file is `/opt/acs/nexus/conf/acs_nexus.ini`. To support the containerized application running on the ACS Edge Server, you need to open this file (from the Host controller) and confirm that `Edge.Enabled` is set as **true** and that `HostController.Enabled` is set as **false**. To support the containerized application running on the ACS Unified Server, you need to confirm that TestFloor_Server.Enabled is set as true.

The applicable contents (and default values) of the Nexus configuration file are shown in the example below. For a description of each item in this file, see Table 2-13.

```
[TestFloor_Server]
Enabled = true
Control_Port = 5001
Data_Port = 5002

[Edge]
Enabled = true
Control_Port = 7001
Data_Port = 7002
Image_Info = images.ini

[HostController]
Enabled = false
Control_Port = 7001
Data_Port = 7002
Hooks = app.ini

[GUI]
Auto_Popup = false
Auto_Close = true
```

NOTE: For the "Port" items in the table below, the port that you use should be open and available.

Table 2-13. acs_nexus.ini Content Descriptions

Item	Description
`TestFloor_Server`	`Enabled` Whether connection is supported from TestFloor_Server.
	`Control_Port` The port listening for control connection from TestFloor_Server. The default port is 5001. Under normal circumstances, this port does not need to be modified unless port listening fails to start.*
	`Data_Port` The port listening for data connection from TestFloor_Server. The default port is 5002. Under normal circumstances, this port does not need to be modified unless port listening fails to start.*

	`Brokers`
	Brokers to upload Nexus data to ACS Unified Server (AUS). The default brokers are:
	`unifiedserver.local:29092,unifiedserver.local:39092,unifiedserver.local:49092`
	The above default brokers do not need to be modified under normal circumstances, unless the Kafka service configuration on AUS has changed. For example, if the AUS provides Kafka service on ports 5092, 5093, and 5094, the Brokers setting needs to be updated, and ACS Nexus must be restarted for the changes to take effect.

`Edge`	`Enabled` Whether connection is supported from the ACS Edge Server. **NOTE:** Only either `Edge` or `HostController` can be set to enabled, not both.
	`Control_Port` The port listening for control connection from the ACS Edge Server. The default port is 7001. Under normal circumstances, this port does not need to be modified unless port listening fails to start.*
	`Image_Info` The name of image configuration file. This file must be deployed in the folder `/opt/acs/nexus/conf/`.
`HostController`	`Enabled` Whether connection is supported from the Host controller. **NOTE:** Only either `Edge` or `HostController` can be set to enabled, not both.
	`Control_Port` The port listening for control connection from the Host controller. The default port is 7001. Under normal circumstances, this port does not need to be modified unless port listening fails to start.*
	`Data_Port` The port listening for data connection from the Host controller. The default port is 7002. Under normal circumstances, this port does not need to be modified unless port listening fails to start.*
	`Hooks`

Item	Description
	The name of the application configuration file. This file must be deployed in the folder `/opt/acs/nexus/conf/`.
`GUI`	`Auto_Popup` Whether to display the ACS Nexus GUI after SmarTest session is ready. When ACS Nexus is started after SmarTest is started, the ACS Nexus GUI will display after ACS Nexus is started.
	`Auto_Close` Whether to close the ACS Nexus GUI after SmarTest session is terminated. When SmarTest session is terminated, the ACS Nexus GUI will automatically close.
`Auto_Deploy`	`Enabled` Whether to support auto deploy mode. Set to 'true' to enable auto deploy mode, otherwise set to 'false.' For more information, see Auto Deploy Mode.

* For example, if the default port has been occupied or blocked by the firewall (which can be confirmed through the control log). If this happens, the port needs to be modified and ACS Nexus needs to be restarted for the change to take effect. Successful listening can be confirmed through the control log or by using the Linux system command 'netstat'.

Restarting ACS Nexus

For any changes to acs_nexus.ini to take effect, ACS Nexus should be stopped and manually restarted. Follow the procedure below to manually restart ACS Nexus.

1. Ensure SmarTest is already shutdown.

2. Open a command terminal and stop ACS Nexus by entering the following command (root user only):

```
#systemctl stop acs_nexus.service
```

3. To confirm that no ACS_Nexus processes are running, enter:

```
#ps -ef | grep acs/nexus | grep -v grep
```

   The output for the above command should be empty.

4. Start ACS Nexus by entering the following command (root user only):

```
#systemctl start acs_nexus.service
```

5. To confirm that ACS_Nexus processes are started, enter:

```
#ps -ef | grep acs/nexus | grep -v grep
```

   Below is an output example (SmarTest 7) for the above command.

```
root   112973    1  0 Jul07 ?    00:00:17 /opt/acs/nexus/bin/acs_nexus
root   112985    1  0 Jul07 ?    00:07:07 /opt/acs/nexus/bin/control_smt7 &
```

2.5.2 Auto Deploy Mode

NOTE: Auto Deploy mode is the default setting. In this mode, ACS Nexus supports production together with other RTDI products.

To dynamically deploy the specified Applications to the ACS Edge Server in real time according to production requirements, RTDI users must be able to query Application information from the ACS Container Hub through ACS Nexus before each production start. This mode is called Auto_Deploy mode, and the Application information file is called AppDescriptor.

To change the configuration to Auto Deploy mode, set Auto_Deploy.Enabled to 'true' in `/opt/acs/nexus/conf/acs_nexus.ini`.

When Auto_Deploy mode is enabled, ACS Nexus will use AppDescriptor (QueryAppDescriptor) as the configuration file to manage the Application lifecycle during production. ACS Nexus loads AppDescriptor when the SmarTest session starts. Therefore, make sure to call the query tool (QueryAppDescriptor) before the SmarTest session starts.

The path for AppDescriptor is `/opt/acs/nexus/bin/QueryAppDescriptor`. The table below provides information on the Command Line Interface parameters.

Table 2-14. Command Line Interface Parameter Details

Parameter	Meaning	Required	Description	Default Value
-h	Help	No	Show help information and then exit.	NA
-v	Version	No	Show the version and then exit.	NA
-c	Customer name	Yes	Set the customer name. Identifies the current customer so that it can be mapped to the according Container Hub Organization.	(empty)
-d	Device name	Yes	Set the device name.	(empty)
-f	Product family	No	Set the product family.	(empty)
-n	Test Program name	No	Set the test program name.	(empty)
-r	Test Program revision	No	Set test program revision.	(empty)
-u	URL	No	Set the basic url address for sending the query command. The default address is: https://unifiedserver.local/deployment/v1/application-descriptors/query	
-o	Output file	No	Set the absolute path of the output file, which stores the data part of the returned json format information. The default path is: /opt/acs/nexus/conf/app_descriptor.json	

2.5.3 Non-Auto Deploy Mode (Auto Deploy Mode Disabled)

NOTE: Non-Auto Deploy mode is used for engineering purposes only. To use Non-Auto Deploy mode, set `Auto_Deploy.Enabled` to false /opt/acs/nexus/conf/acs_nexus.ini.

The images configuration file (`images.json)` is used to define the containerized application that needs to be deployed and executed on the ACS Edge Server by ACS Nexus for a specific device.

The path for the images.json file is `/opt/acs/nexus/conf/images.json`. The contents and default values of this file are shown below. For a description of each item in this file, see Table 2-15. For any modifications to this file to take effect, SmarTest must be restarted.

NOTE: Starting from ACS Nexus v2.0.0, the "images" file format is changed from .ini to .json. With this change, all image-related functions require ACS Edge Server version 2.3.1 or higher.

```
{
    "selector": {
        "device_name": "xxx.tp"
    },
    "edge": {
        "address": "xxx.xxx.xxx.xxx",
        "registry": {
            "address": "xxx.xxx.xxx.xxx",
            "user": "xxx",
            "password": "xxx"
        },
        "containers": [
            {
                "name": "mycontainer",
                "image": "test/myapp:latest",
                "requirements": {
                    "gpu": false,
                    "exposed_ports": [
                        80,
                        443
                    ],
                    "mapped_ports": [
                        "9001:9001",
                        "9002:9002/tcp"
                    ]
                },
                "environment": {
                    "VARNAME": "value",
                    "VARNAME2": "value2"
                },
                "metrics": {
                    "port": 9001,
                    "path": "/metrics",
                    "scrape_interval": 90,
                    "scrape_timeout": 5
                },
                "volume_attachments": [
                        "myvolume:rw",
                        "myvolume2:ro"

                ]
            }
        ],
        "volumes": [
            {
                "name": "myvolume",
                "image": "myorg-myproject/mydata:latest"
            },
            {
                "name": "myvolume2",
                "image": "myorg-myproject/mydata:latest"
            }
        ]
    }
}##############################################
```

Table 2-15. images.json Content Descriptions

Item	Description
`selector`	`device_name` Set the device name that will run.
`edge`	`address` Configure the Edge Server IP. Used for development testing only. Do not configure in production.
`edge.registry`	`address` The location of container registry, such as ACS Container Hub or Docker hub.
	`user` Authentication username for registry when pulling an image.
	`password` Authentication password for registry when pulling an image.
`edge.containers`	`name` The name of the container.
	`image` The name of the image.
	`environment` Set the environment variable for the container.
	`metrics` Set the metrics for the container.
`edge.containers.requirements`	`gpu` Configure whether the corresponding container enables GPU. Valid value: true/false
	`exposed_ports` This option can be used to expose container ports.

	`mapped_ports` Use this option to expose and publish container ports.
`edge.containers.volume_attachments`	This option is used to set the attach mounted volumes.
`edge.containers.volumes`	`name` Define the volume name to create.
	`image` Define the image to create the volume.

2.5.4 Container Configuration File

NOTE: Starting in ACS Nexus v2.0.0, the container configuration file is not supported. The configuration file can be placed into a volume which is attached to a specific container.

2.5.5 Workflow

NOTE 1: Make sure the application is running before starting production from Prober/Handler side, otherwise there could be data lost at the beginning of lot. Refer to the Nexus GUI for application status.

NOTE 2: Do not stop or restart Nexus during SmarTest session, otherwise Applications on the Edge server cannot be destroyed automatically. This will result in an exception in the deployment of Applications on the Edge server the next time SmarTest is started.

Following is a diagram that illustrates workflow of the Application on the ACS Edge Server.

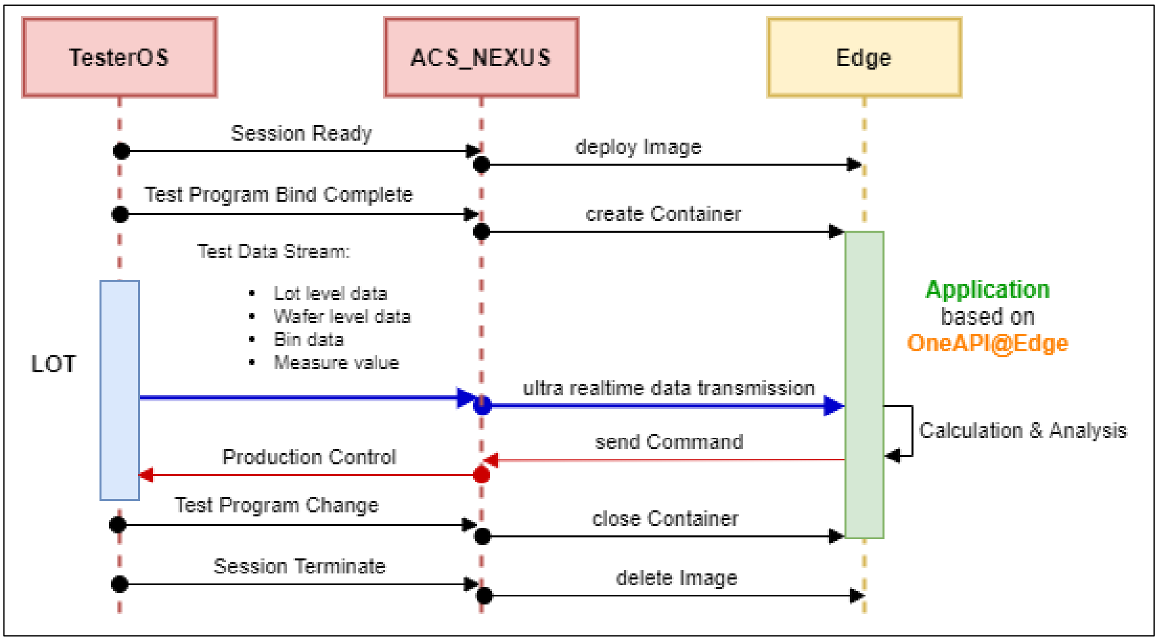


Figure 2-2. Application Workflow on ACS Edge Server

2.6 Supporting Application on the Host Controller

ACS Nexus provides support for real-time data and control capability to the application running on the Host controller. Only one application runtime is supported inside the Host controller, meaning that only one application can connect to consume data and send commands. For other applications, the application vendor must determine how to share the data.

The below diagram offers a general view of how ACS Nexus supports the containerized application running on the Host controller.

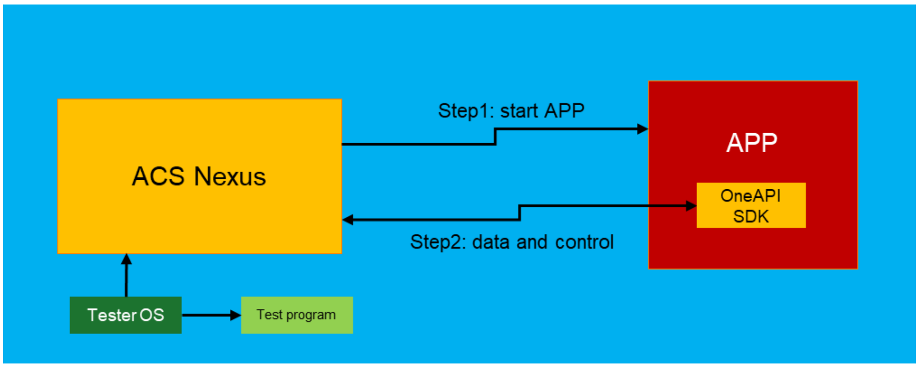


Figure 2-3. ACS Nexus Support for Containerized Application on Host Controller

2.6.1 Nexus Configuration File

The acs_nexus.ini configuration file is used for initialization of the ACS Nexus service. After modifying content in this file, ACS Nexus should be manually restarted for any changes to take effect (see Restarting ACS Nexus).

The path for the acs_nexus.ini file is `/opt/acs/nexus/conf/acs_nexus.ini`. To support the application running on the Host controller, you need to open this file (from the Host controller) and set `Edge.Enabled` as **false** and set `HostController.Enabled` as **true**, as shown below. For a description of each item in this file, see Table 2-13.

```
[TestFloor_Server]
Enabled = true
Control_Port = 5001
Data_Port = 5002

[Edge]
Enabled = false
Control_Port = 7001
Data_Port = 7002
Image_Info = images.ini

[HostController]
Enabled = true
Control_Port = 7001
Data_Port = 7002
Hooks = app.ini

[GUI]
Auto_Popup = false
Auto_Close = true
```

2.6.2 Application Configuration File

The application configuration file (app.ini) is used for hooks of the application in the Host controller. After modifying content in this file, ACS Nexus should be manually restarted for any changes to take effect (see Restarting ACS Nexus).

The path for the app.ini file is `/opt/acs/nexus/conf/app.ini`. The contents (and default values) of this file are shown below. For a description of each item in this file, see Table 2-16.

```
[app1]
smt.session_ready = /home/demo/app/bin/start.sh
smt.session_terminated = /home/demo/app/bin/stop.sh
nexus.completed = =latest
operate_timeout=300
```

Table 2-16. Container Configuration File Content Descriptions

Item	Description
`[demoAppName]`	This is the host controller application name. The user can assign any name, but the item name must be unique. Every item configured means that ACS Nexus need to handle this application.
`smt.session.ready`	The script that will be executed when the SmarTest session is read. This item is normally used for starting the application.
`smt.session_terminated`	The script that will be executed when SmarTest session is terminated. This item is normally used for stopping the application.
`nexus.completed`	The script which will be executed when Nexus is terminated. This item is normally used for stopping the application.

NOTE: Make sure that the application is connected to ACS Nexus before starting production from the prober/handler side. Otherwise, data can be lost at the beginning of the lot.

2.6.3 Workflow

Following is a diagram that illustrates workflow of the Application on the Host Controller.

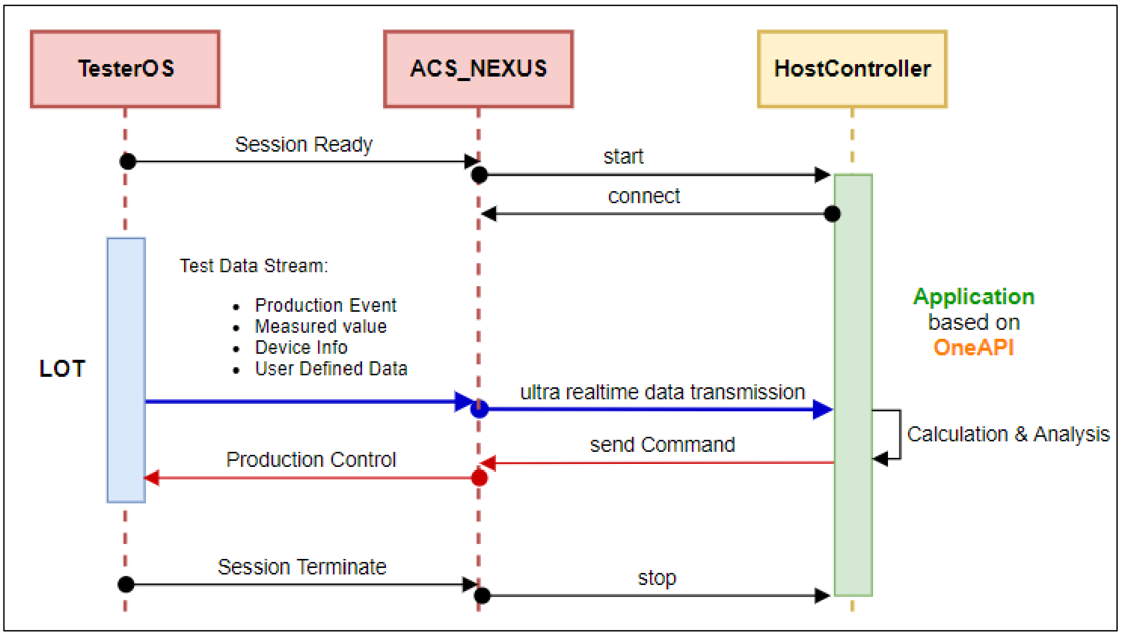


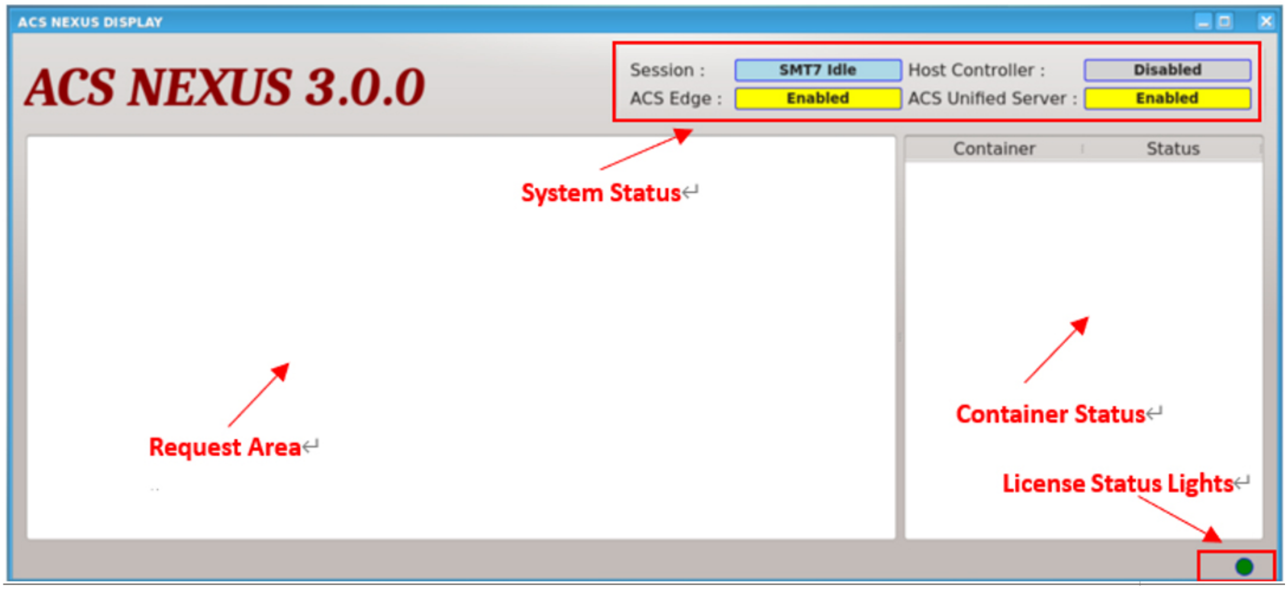
Figure 2-4. Application Workflow on Host Controller

2.7 ACS Nexus GUI

NOTE: ACS Nexus automatically starts and runs perpetually on the Host Workstation after installation. If an ACS Nexus restart is needed, see Restarting ACS Nexus.

ACS Nexus provides a graphic interface to present the system status, command records, and exception information, all of which can be automatically started and closed according to ACS Nexus configuration. The ACS Nexus GUI consists of a System Status area that provides connection status information for all connected systems, and an Information area that provides command records and exception information, as indicated in the figure below.

NOTE: Only one user at a time can run ACS Nexus on the same Host Workstation.



Session : SMT7 Idle
Host Controller : Disabled
ACS Edge : Enabled
ACS Unified Server : Enabled

Figure 2-5. ACS Nexus GUI

License availability information is displayed by circular indicator in the lower right of the window. This indicator represents data collection license availability. Green indicates the license is available, red indicates that the license is either missing or invalid, and grey indicates that the license is not checked.

For additional details on the ACS Nexus GUI, see Table 2-17 and Table 2-18.

Table 2-17. System Status Indicators

System	Status Indicator
Session	**Ready** SmarTest Session has been started. Shows the SmarTest version (SMT7 or SMT8). **Idle** SmarTest Session has been already terminated. Shows the SmarTest version (SMT7 or SMT8)
Host Controller	**Disabled** ACS Nexus does not provide connection service to the application based on OneAPI at the Host Controller. **Initial Fail** ACS Nexus failed to provide connection service to the application based on OneAPI at the Host controller. **Disconnected** ACS Nexus is ready to provide connection service to the application based on OneAPI at the Host controller, but no application is connected at present. **Connected** ACS Nexus is ready to provide connection service to the application based on OneAPI at the Host controller, and the application is currently connected.
ACS Edge Server	**Disabled** ACS Nexus is configured not to provide service to ACS Edge Server. **Enabled** ACS Nexus is configured to provide service to ACS Edge Server. **Down** ACS Nexus detects that ACS Edge Server is not available. **Available** ACS Nexus detects that ACS Edge Server is available, and ACS Nexus is ready to provide service to the Edge Server.
ACS Unified Server	**Enabled** ACS Nexus is configured to monitor connection status to the ACS Unified Server. **No Monitoring** ACS Nexus is configured not to monitor connection status to ACS Unified Server. **Not Available** ACS Nexus is configured to monitor connection status to the ACS Unified Server, but the ACS Unified Server is not presently responding. **Available** ACS Nexus is configured to monitor connection status to the ACS Unified Server, and the ACS Unified Server is currently responding.

Table 2-18. Information Area Contents

Item	Status Indicator

Record of Control Information*	This area will record the information of control commands triggered by the application. Each line presents a command with the following information, separate by "|". For example:
	`2023–07-15 10:00:15 Edge|DebugApp_ADV_01|at 2023-07-15 10:00:15|PAUSE|just test for Pause Command sent from Edge`
	`2023–07-15 10:00:30 Edge|DebugApp_ADV_01|at 2023-07-15 10:00:30|STOP|just test for Stop Command sent from Edge`
	Received time
	The timestamp when ACS Nexus received this command.
	Resource
	The resource of the command, either `Edge` or `TestFloor_Server`
	Application
	The application information in the form of: {Name}_{Vendor}_{Version}.
	Sent time
	The timestamp when the application sent this command.
	Command
	The name of command, either `Pause` or `Stop`.
	Reason
	The reason why application sends this command. There is no limit to the text format.

Exception Information*	This area will print the exception information related to dynamic switch. Each line presents an exception with the following information starting with "Warning." For example: `Warning|2023-06-08 14:56:44|Edge -> Edge connection is enabled, but no images are configured` **Received time** The timestamp when ACS Nexus threw this exception. **Resource** The resource of the exception (switch container). **Reason** The reason why application throws this exception. There is no limit to the text format.
Container Status Information	The container name and status information is displayed in a table. Hover over the container name to display the image:tag. Image-related status or container status is displayed as described below: **pullStarting** ACS Nexus is starting to pull image of target application. **pullSuccess** ACS Nexus pulls image of target application successfully. **pullFailed** ACS Nexus image pull of the target application failed. **pullTimeout** ACS Nexus pulls image of target application timeout. **ImageDeleteStarting** ACS Nexus is starting to delete the image of target application.

Item	Status Indicator
	imageDeleteSuccess ACS Nexus deleted image of the target application successfully. **imageDeleteFailed** ACS Nexus failed to delete image of the target application. **creating** ACS Nexus is starting to create the target application container. **created** ACS Nexus creates target application container successfully and container has never started since creation. **createFailed** ACS Nexus failed to create target application container. **runStarting** ACS Nexus is starting to run target application container. **running** The target application container is running. **runFailed** The target application container run failed. **stopStarting** ACS Nexus is starting to stop target application container. **stopFailed** ACS Nexus failed to stop target application container. **removeStarting** ACS Nexus is starting to remove target application container. **removed** The target application container is removed. **removeFailed** Removal of the target application container failed. **restarting** The target application container is in the process of restarting. **paused** The target application suspends all the processes for an indefinite time. **exited** The target application process inside the container terminated. **dead** This state is achieved when trying to remove the target application container, but it cannot be removed because some resources are still in use by an external process. **removing** The target application container is being removed. **unknown** An error occurred while getting the status.

Item	Status Indicator
License Status	Indicates the current license status. The status light will show one of two states: Green - indicates that the ACS Nexus DataCollection license is valid Red - indicates that the ACS Nexus DataCollection license is invalid. Grey – indicates that the ACS Nexus DataCollection license is not checked.
Certificate Expiration Reminder	When the GUI is launched, ACS Nexus will check the validity of the certificate. No prompt will be given if the usable period is greater than seven days, if the usable period is less than seven days or the certificate has expired, this area will print the exception information related to certificate status, with following information: **Received time** The timestamp when ACS Nexus threw this exception. **Reason** The reason why ACS Nexus threw this exception. There is no limit to the text format.

*This area will be cleared in the event of SmarTest startup or ACS Nexus reboot.

2.8 STDF Replay

This section describes how to use STDF replay. The basic steps are as follows:

1. Change the nexus mode in acs_nexus.ini configuration file.
2. Use the Edge Control Tool to start the container.
3. Use the Data Replay Tool to replay the STDF file.

WARNING: Only one client can connect at a time. If multiple clients connect at the same time, it may cause some unknown errors.

1. Change the Nexus Mode.

The Nexus mode must be changed in the acs_nexus.ini file. The path for this configuration file is /opt/acs/nexus/conf/acs_nexus.ini. In this file, change the Nexus Mode to "replay."

```
[Mode]
Nexus_Mode=replay
```

Confirm that Edge is enabled (true). The default setting is true, so there should be no need to modify this parameter.

```
[Edge]
Enabled=true
```

After the Nexus Mode is changed, the ACS Nexus must be restarted for the change to take effect. Manually restart ACS Nexus by entering the following in a command line on the Host workstation:

```
systemctl restart acs_nexus.service
```

2. Use the Edge Control Tool to start the container.

The path for the Edge Control Tool is /opt/acs/nexus/bin/EdgeControlTool. This is a command line tool. Use -h to see command help.

WARNINGS: Observe the below warnings when using the Edge Control Tool.

- **This tool can only reply one STDF file at one time.**
- **The STDF file size should be less than 10GB.**
- **The input parameters are required to be in a fixed order. For example, the following would be considered invalid input:**

  ```
  # ./EdgeControlTool stop -registry 192.168.0.1:5000  -edge_ip 127.0.0.1 -
  name spc-app -tag latest -user default -passwd default -operate_time 300
  -host_ip 192.168.0.1 -container_name app1
  ```

- **Ensure that the image has been pulled successfully before starting a container.**
- **Ensure that the container has been closed successfully before deleting the image.**
- **This tool does not support multiple processes running at same time. For example, it may behave unexpectedly in the following case:**

  ```
  ./EdgeControlTool -h  &  ./EdgeControlTool pull xxx  &&
  ./EdgeControlTool start xxx
  ```

Table 2-19. Edge Control Tool Usage

Action	Example
Command Help	`./EdgeControlTool  -h`
Pull Image	`./EdgeControlTool pull -edge_ip 127.0.0.1 -registry 192.168.0.1:5000 -name spc-app -tag latest -user default -passwd default -operate_time 300 -host_ip 192.168.0.1 -container_name app1`
Delete Image	`./EdgeControlTool delete -edge_ip 127.0.0.1 -registry 192.168.0.1:5000 -name spc-app -tag latest -user default -passwd default -operate_time 300 -host_ip 192.168.0.1 -container_name app1`
Start Container	`./EdgeControlTool start -edge_ip 127.0.0.1 -registry 192.168.0.1:5000 -name spc-app -tag latest -user default -passwd default -operate_time 300 -host_ip 192.168.0.1 -container_name app1`
Stop Container	`./EdgeControlTool stop -edge_ip 127.0.0.1 -registry 192.168.0.1:5000 -name spc-app -tag latest -user default -passwd default -operate_time 300 -host_ip 192.168.0.1 -container_name app1`

3. Use the Data Replay Tool to Replay the STDF File

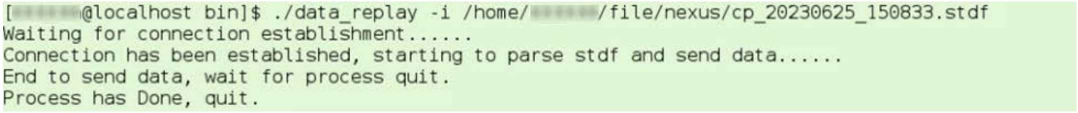
See Data Replay Tool.

2.9 Data Replay Tool

After starting a container, use the Data Replay to replay the STDF file. This tool it is a command line tool. Use -h to see command help.

Table 2-20. Data Replay Tool Usage

Action	Example
Command Help	`./data_replay -h`

Replay STDF	`./data_replay -I XXX.stdf`
	
	`[[redacted]@localhost bin]$ ./data_replay -i /home/[redacted]/file/nexus/cp_20230625_150833.stdf`
	`Waiting for connection establishment......`
	`Connection has been established, starting to parse stdf and send data......`
	`End to send data, wait for process quit.`
	`Process has Done, quit.`

For other functions, see the configuration file /opt/acs/nexus/conf/data_replay.ini.

NOTES: For the "indextime" key in `[test_item]`, it will delay from the touchdown end to the next touchdown start and support a range from 0-10000 ms. If the configuration is more than 10000 ms, use 10000.

For the "test_num" key, it is configured for every "`testnumber:delaytime`" and split by `";"`. The delaytime unit is ms. The supported range is 0-10000. If the configuration is more than 10000 ms, use 10000. If there is an error in the format, some unexpected issues may occur.

For the "quittime" key in the process, it is sometimes necessary for the process to continue for a while before exiting.

All index time will delay 200ms. After test number reaches 100, ACS Nexus will delay 300ms before sending the next event. After making changes to data_replay.ini, data_replay will need to be restarted.

WARNING: The test_num key needs to be configured as test number and the delay time (testnumber:delaytime) or unexpected errors may occur.

2.10 FAST-API Support on ACS Edge

FAST-API service is supported for the containerized application running on the ACS Edge Server to communicate with the Host Controller test program. The below diagram illustrates the flow for FAST-API support within the RTDI environment.

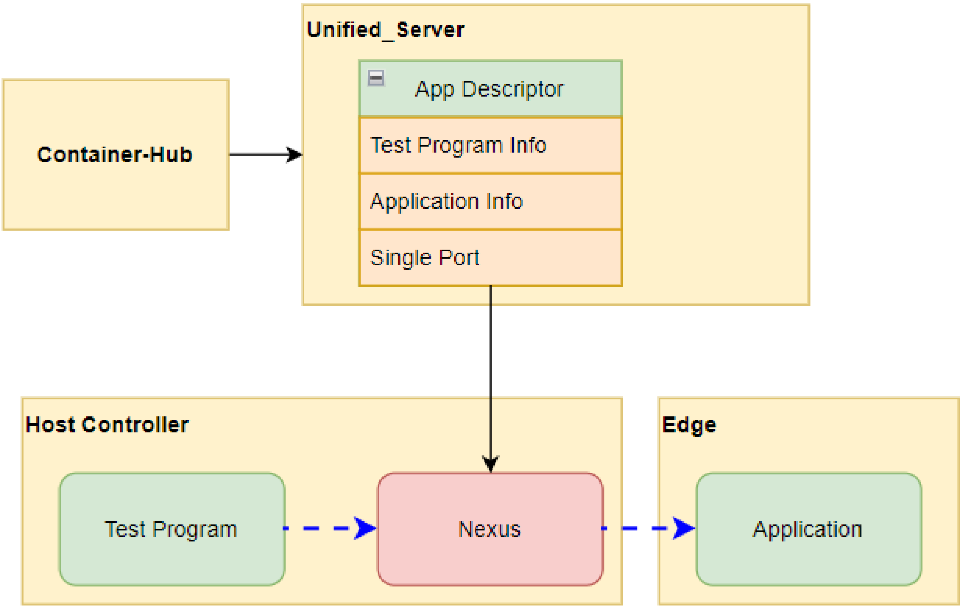


Figure 2-6. Fast API Support in RTDI Environment

2.10.1 Configuration

To set-up FAST-API support, the first step is to login to the ACS Container Hub to configure your own application by specifying the mapping relationship between the port your application listens to and the open port of the ACS Edge Server. For specific operation steps, contact your Advantest representative.

NOTE: ACS Nexus currently only supports a single application and a single mapped_port.

2.10.1.1 Nexus Query Application Description File

Use the ACS Nexus query tool (QueryAppDescriptor) to obtain the application description file before starting Smartest. For specific operations relating to the query tool, see Auto Deploy Mode.

2.10.1.2 Nexus Configuration

In the ACS Nexus configuration file (acs_nexus.ini), set `[Gateway/Auto_Start]` and `[Gateway/Auto_Close]` to true, as shown below.

```
[Gateway]
Auto Start=true
Auto Close=true
```

2.10.2 Access FAST-API Service

After the container is started, you can access the fast API service through the following URL:

http://localhost:8888/**[your_api_path]**

2.11 ACS Edge Redundancy Support

ACS Nexus automatically switches customer containerized application between the ACS Edge Server of test cell and the Edge Services on the ACS Unified Server.

The Gateway redirects the request from the customer-specific http client to the customer containerized application running on the ACS Edge Server or customer containerized application running in the Edge Services on the ACS Unified Server, as illustrated in the diagram below.

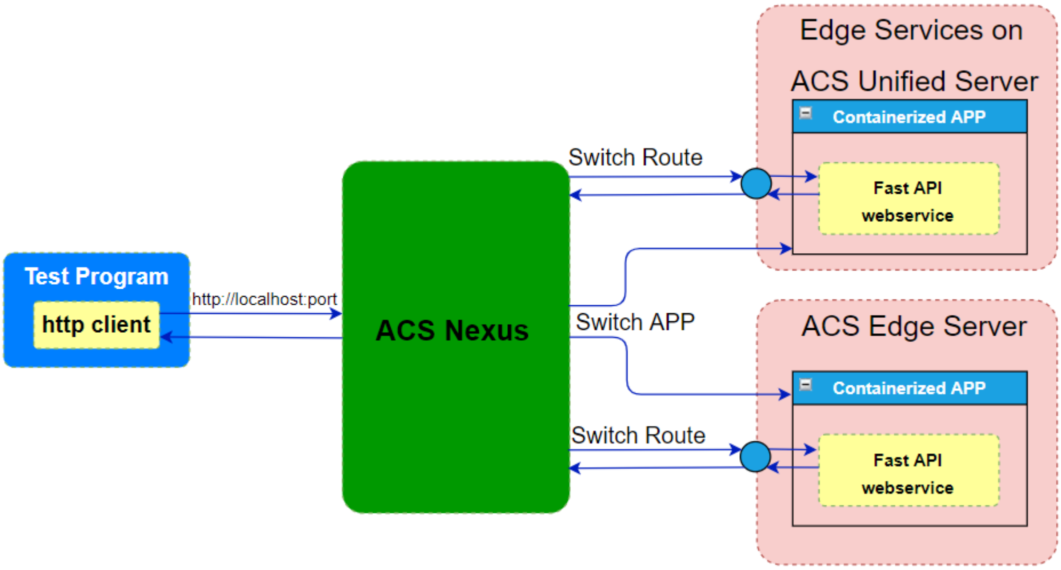


Figure 2-7. Edge Redundancy Gateway Redirection

Following are limitations of the ACS Edge redundancy feature:

- This feature only supports applications that do not include OneAPI.
- This feature requires a Gateway to work together to support it.
- This feature only supports one application.
- ACS Nexus only switches applications to Edge Services on the Unified Server, so pulling an image to the Unified Server may take a long time and some results will be lost.

2.11.1 Usage Cases

Usage Case 1:

Upon starting Smartest, ACS Nexus will check if the ACS Edge Server is online. If the ACS Edge server is not online or not operating, ACS Nexus will open the customer containerized Application in Edge Services on the Unified Server.

Usage Case 2:

While the customer containerized Application is running, if ACS Nexus detects that the ACS Edge Server is down, it will open the customer containerized Application in the Edge Services on the Unified Server.

Usage Case 3:

If the ACS Nexus opened the customer containerized Application on the Unified Server previously due to the ACS Edge Server being down or offline, when ACS Nexus starts the container next time and the ACS Edge Server has already recovered by then, the customer containerized Application will be running in the ACS Edge Server again.

2.11.2 Configuration

The ACS Edge redundancy feature is disabled by default. To enable this feature, the correct configuration must be implemented in acs_nexus.ini, as indicated below:

- `[Edge/Redundancy_Enabled]` must be set to true
- `[Auto_Deploy/ Enabled]` must be set to true

2.11.3 Gateway

Gateway must be installed in the following folder:

```
/opt/acs/nexus/gateway
```

Gateway runs on port 8888 by default, but can be changed to a different port. To change the port, modify the port number in the configuration file located in /opt/acs/nexus/gateway/conf . The modified configuration file will not take effect until the next time Gateway is started.

The Gateway lifecycle is managed using the ACS Nexus configuration file (acs_nexus.ini) as follows:

- When `[Gateway/Auto_Start = true]`, the Gateway will be started when Smartest starts.
- When `[Gateway/Auto_Close=true]`, the Gateway will be closed when Smartest is terminated.

2.12 Bi-Directional Communication between TP and RTDI Application

To facilitate a modern test scheme, whereby the test flow and test parameters can be dynamically adjusted for each touch down, ACS RTDI can open the communication channel between the test program and RTDI Application so that they can transmit data and calculation results in real time. This bi-directional communication is supported using NexusTPI and OneAPI.

2.12.1 NexusTPI

NexusTPI is a dynamic library for the test program. When the user needs to utilize communication between the test program and RTDI Application, the interfaces provided by this library needs to be called in the test program code.

2.12.1.1 NexusTPI for SMT8

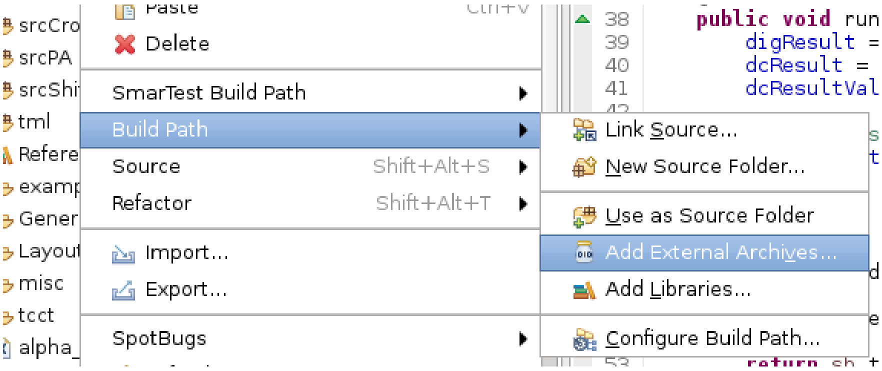
NOTE: NexusTPI is supported starting in Nexus v2.1.0. Supported for SMT 8.6.x and Java 11/17.

After installing Nexus (v2.1.0 or higher), NexusTPI.jar can be found in the following path:

```
/opt/acs/nexus/testmethod/smt8/
```

To import NexusTPI.jar, follow these steps:

1. On the SmarTest WorkCenter, select Device. Right-click and select [**Build Path**] in the pop-up menu, then select [**Add External Archives…**] in the subsequent pop-up menu as shown below.



```
public void run
    digResult =
    dcResult =
    dcResultVal
```

Figure 2-8. Add External Archives

2. In the new dialog box called *JAR Selection*, find the /opt/acs/nexus/testmethod/smt8/ folder, click **NexusTPI.jar**, and click [**OK**].

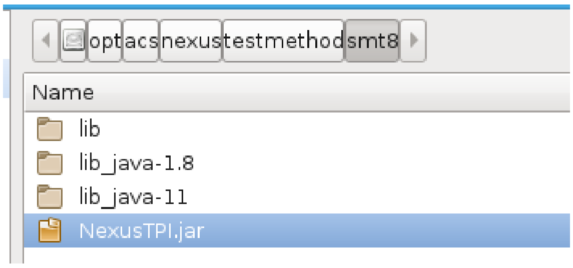
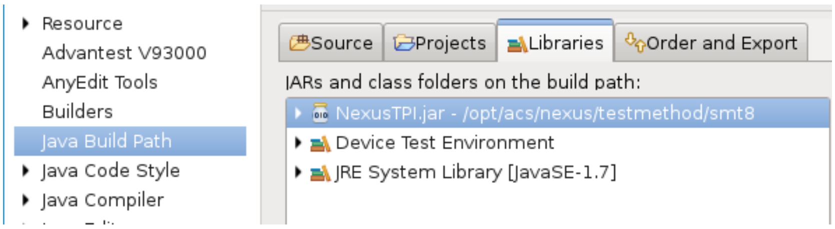


Figure 2-9. Select NexusTPI.jar

3. For a successful addition, you should be able to see NexusTPI.jar in the [Libraries] tab in the Device's Build Path, as shown below.



JARs and class folders on the build path:

- NexusTPI.jar - /opt/acs/nexus/testmethod/smt8
- Device Test Environment
- JRE System Library [JavaSE-1.7]

Figure 2-10. NexusTPI.jar in Device Build Path

Using NexusTPI interfaces

Table 2-21 below provides information on the NexusTPI interfaces. Table 2-22 provides error code information.

Table 2-21. NexusTPI Interfaces

Interface	Function	Parameters	Return	Example
`init()`	Specify initialize step before executing other NexusTPI interface functions. **NOTE:** It is strongly recommended to invoke `NexusTPI.init()` in the setup stage.	None	Error Code (int)	See Example 1
`location(String)`	Specify the location of the RTDI Application	"aus" or "edge" (string)	NexusTPI	See Example 2
`target(String)`	Specify target RTDI Application.	Container Name	NexusTPI	See Example 3
`timeout(int)`	Set timeout for the communication. • minimum 1 • maximum 10 If the input parameter is outside this range, the upper and lower limits of this range are taken.	Timeout value (in seconds)	NexusTPI	See Example 4
`send(String)`	Send data to target application. No response required.	The data to be sent	Error Code (int)	See Example 5

Interface	Function	Parameters	Return	Example
`request(String)`	Initial request to target application. Response is required.	Content of the request (JSON formatted string)	Error Code (int)	See Example 6
`getResponse()`	Get response message. Should be used after "request" interface succeeds.	None	Response (string)	See Example 6
`DFF`	Get single instance of DFF implementation class to perform subsequent DFF operations.	None	N/A	
`DFF.importSchema(String)`	Load schema from input JSON file, and register the schema to the ACS Unified Server	Schema file path (string)	Schema code success or error (int)	
`DFF.set(String,Object)`	Set property key-value pairs for subsequent data uploads	Property key (string) Property value (object)	This	
`DFF.regUploadCallback (Callback)`	Register user-defined callback function to receive upload result	Methods for implementing callback functions	No	
`DFF.upload(String)`	Asynchronous upload data to ACS Unified Server	The DFF data to be sent (string)	DFF upload code success or error (int)	
`DFF.query(String)`	Create query request with direct SQL query statement	DB query statement (string)	Response (NexusTPI.Result) code (API return code) body[0] (jobIDF)	
`DFF. getJobResult(String,String)`	Obtain the query status and DFF data according to the JobID	JobID (string) Format (string) a:"json"	Response (NexusTPI.Result) code (API return code) body[0] (error msg) body[1] (dff-data)	

Example Code 1: init()

```
import nexus.tpi.NexusTPI;

public class TPI_Test_Suite extends TestMethod {

    @Override
    public void setup(){
        int code = NexusTPI.init() ;
        if(code == NexusTPI.SUCCEED){
           //successful initialization
        }else{
           //fail initialization,for example NexusTPI.LICENSE_INVALID = 6
        }
    }

    @Override
    public void excute(){
       NexusTPI.location("aus").target("appName").send("command");
    }
}
```

Example Code 2: location()

```
import nexus.tpi.NexusTPI;

String location= "aus";
NexusTPI.location(location) ;
```

Example Code 3: target()

```
import nexus.tpi.NexusTPI;

String appName = "app1";
NexusTPI.target(appName) ;  // specify target app as "app1"
```

Example Code 4: timeout()

```
import nexus.tpi.NexusTPI;

int t_out = 1;            // timeout 1 second
NexusTPI.timeout(t_out);
```

Example Code 5: `send()`

```
import nexus.tpi.NexusTPI;

String appName = "App1";
String data = "xxxxxxxxxxxx";
int res = NexusTPI.SUCCEED;

// case 1: full chain call
res = NexusTPI.target(appName).timeout(2).send(data);

// case 2: syntax sugar – implicit use the last target and timeout
res = NexusTPI.send(data); // send data to last target with last timeout
```

Example Code 6: `request() + getResponse()`

```
import nexus.tpi.NexusTPI;

String appName = "App1";
String requestJSON = "{\"key\" : \"command_for_test\"}";
int res = NexusTPI.SUCCEED;

// case 1: full chain call
res = NexusTPI.target(appName).timeout(2).request(requestJSON);

// case 2: syntax sugar – implicit use the last target and timeout
res = NexusTPI.request(requestJSON);

String response;
if (res == NexusTPI.SUCCEED) {
    response = NexusTPI.getResponse();
}
```

Table 2-22. Nexus TPI Error Code Descriptions

Code(int)	Code(var)	Definition
0	SUCCEED	Communication succeeded
1	COMMON_ERROR	Common Error (Errors that are not specifically defined)
2	TIMEOUT	Communication time out
3	TARGET_INVALID	Target is invalid **NOTE:** This usually means that the target Application is not defined in the AppDescriptor file or the Application's configuration is illegal.
4	CONNECT_FAILED	Unable to connect to the target Application
5	CERTIFICATE_ERROR	Certificate related errors
6	LICENSE_INVALID	License is invalid
101	SCHEMA_LOAD_FAIL	Schema file failed to load, (possible cause is that the file is not in JSON format)
102	SCHEMA_SAME	Schema with the same $id has already been registered by this Program
103	SCHEMA_REGIST_FAIL	Schema registration failed (possible cause is that the server is down)
104	SCHEMA_REGISTERED	Schema with same $id has already been registered by others
111	DFF_CONNECTED	Properties are valid and upload started, but upload did not complete
112	DFF_PROPERTY_INVALID	Properties invalid for the schema
113	DFF_CONNECT_FAIL	Failed to establish data-upload connection
114	DFF_DATA_EMPTY	Data is empty
115	SCHEMA_NOT_REGISTERED	The process has not successfully registered the schema
121	JOB_FAILED	Job execution failure
122	JOB_RUNNING	Job is being executed

2.12.1.2 NexusTPI for SMT7

After installing Nexus v2.3.0 or higher, libNexusTPI.so can be found in the following path:

```
/opt/acs/nexus/testmethod/smt7/sh_lib-EL7-64bit/
```

To import libNexusTPI, follow these steps:

1. Include the header file path in the C/C++ compiler.

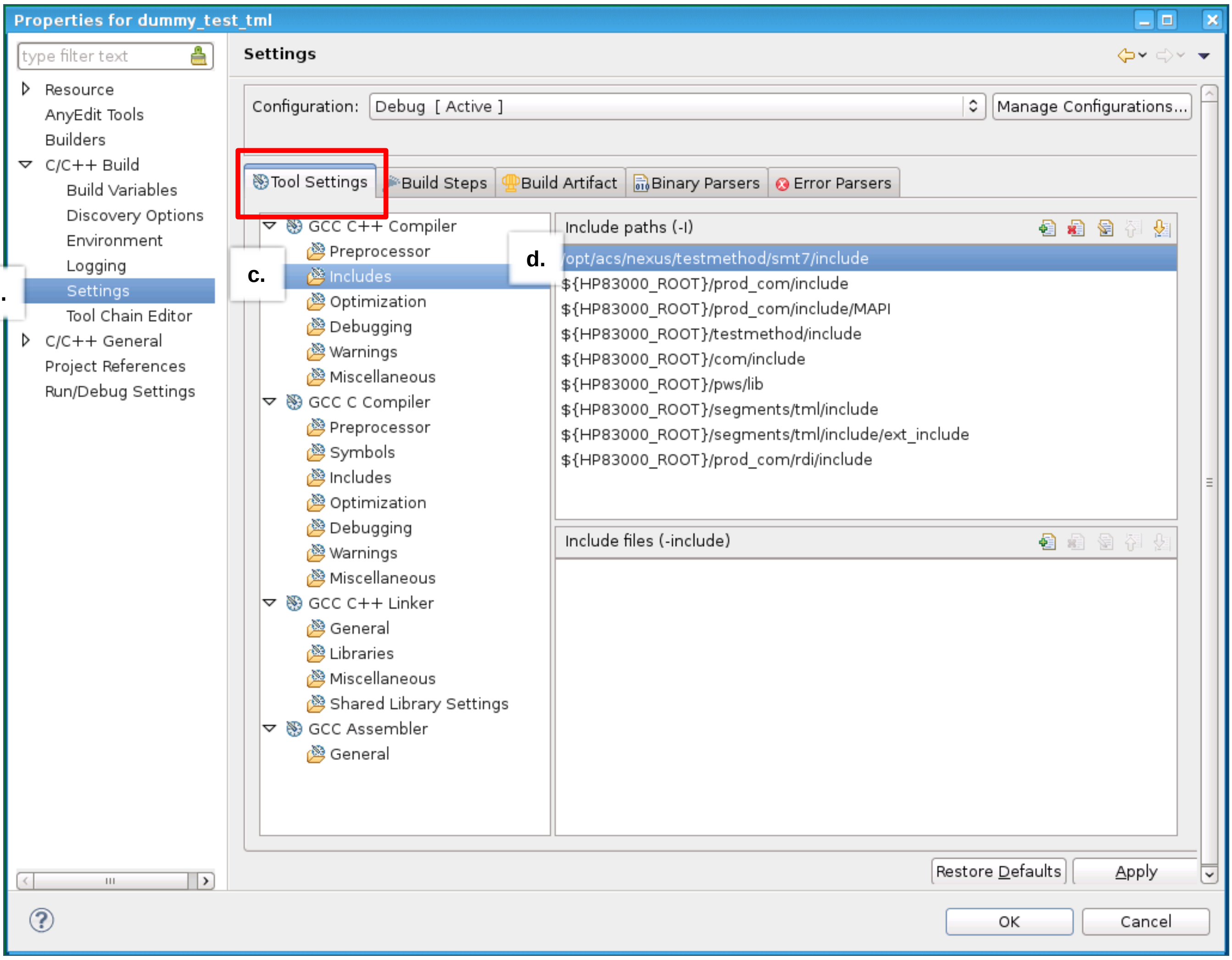
   a. In the C/C++ perspective, right-click the test method project and select the **Properties** menu to display the project Properties window.

   b. In the Properties window, click the **Settings** option under C/C++ Build (in the left-side pane).

   c. In the Tool Settings tab, click the **Includes** option under GCC C++ Compiler.

   d. Add the NexusTPI.h header file path to the Include paths list.

      Path: `/opt/acs/nexus/testmethod/smt7/include`



Configuration: Debug [Active]

Include paths (-I)

```
/opt/acs/nexus/testmethod/smt7/include
${HP83000_ROOT}/prod_com/include
${HP83000_ROOT}/prod_com/include/MAPI
${HP83000_ROOT}/testmethod/include
${HP83000_ROOT}/com/include
${HP83000_ROOT}/pws/lib
${HP83000_ROOT}/segments/tml/include
${HP83000_ROOT}/segments/tml/include/ext_include
${HP83000_ROOT}/prod_com/rdi/include
```

Figure 2-11. Include Header File Path

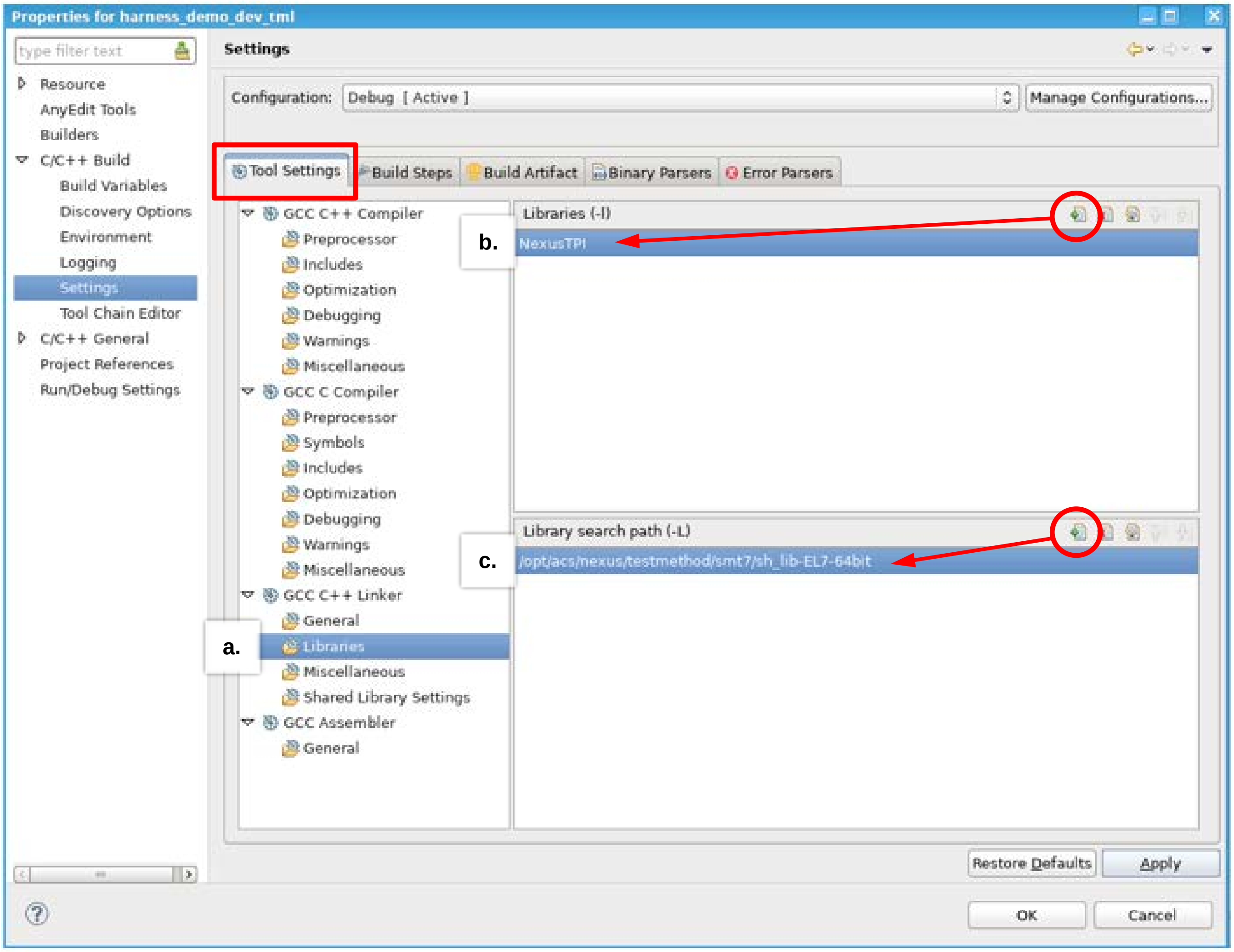
1. Include the library file path and name in the C/C++ linker.

    a. In the Tool Settings tab, click the **Libraries** option under GCC C++ Linker.

    b. Add the libNexusTPI.so library file name to the libraries list.

    c. Add the libNexusTPI.so library file path to the library search paths list.

    Path: `/opt/acs/nexus/testmethod/smt7/sh_lib-EL7-64bit`



Properties for harness_demo_dev_tml

Settings

Configuration: Debug [Active]

Libraries (-l)
NexusTPI

Library search path (-L)
/opt/acs/nexus/testmethod/smt7/sh_lib-EL7-64bit

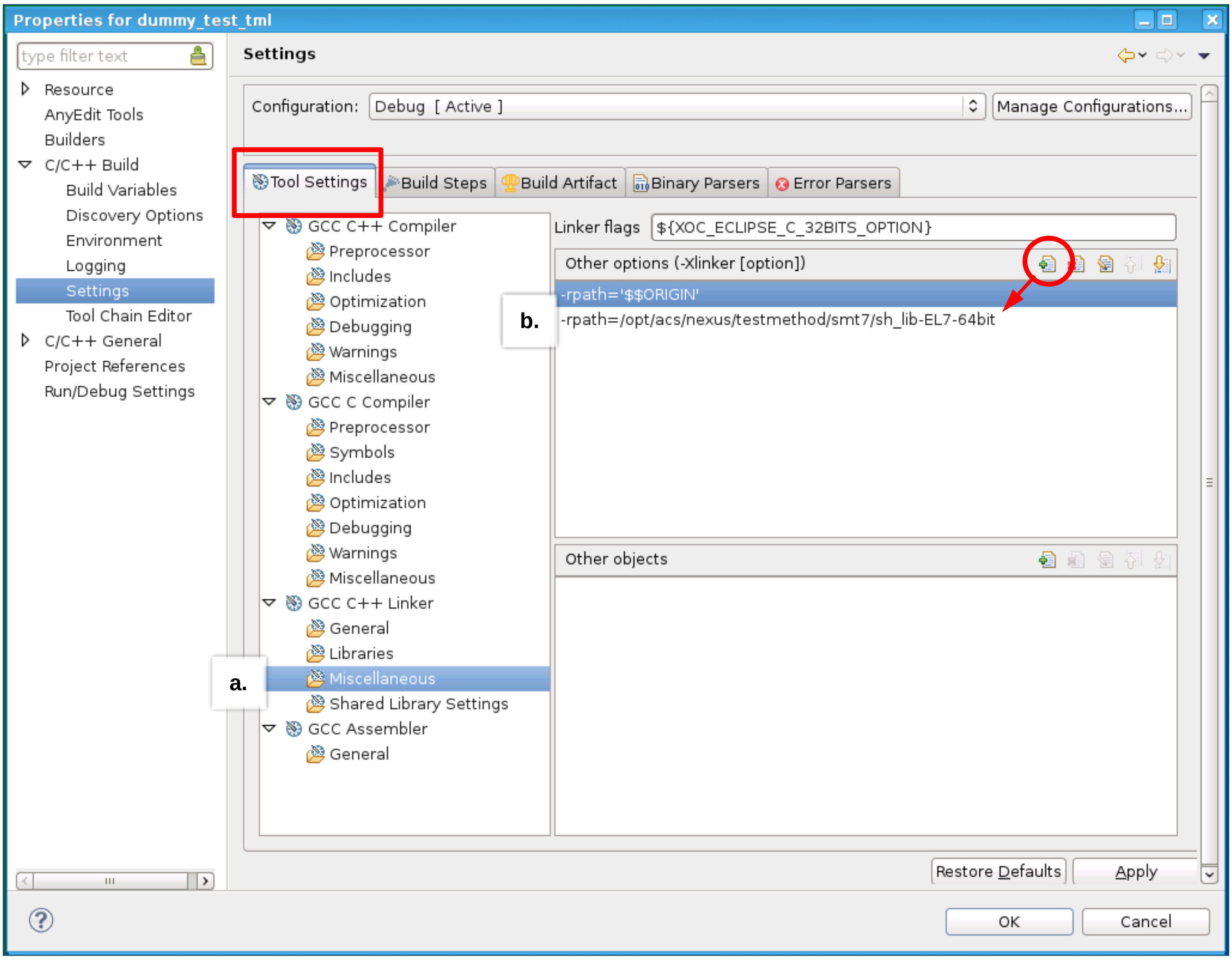
Figure 2-12. Including Library File Path and Name in Linker

2. Include the library path in the C/C++ linker option.

   a. In the Tool Settings tab, click the **Miscellaneous** option under GCC C++ Linker.

   b. Add the rpath option to the other options list.

      Path: `-rpath=/opt/acs/nexus/testmethod/smt7/sh_lib-EL7-64bit`



Configuration: Debug [Active]

Linker flags: ${XOC_ECLIPSE_C_32BITS_OPTION}

Other options (-Xlinker [option])

```
-rpath='$$ORIGIN'
-rpath=/opt/acs/nexus/testmethod/smt7/sh_lib-EL7-64bit
```

Figure 2-13. Including library path in Linker Option

3. To build the test method project, click **Project** > **Build All**.

Using NexusTPI interfaces

Table 2-23 below provides information on the NexusTPI interfaces. Table 2-24 provides error code information.

Table 2-23. NexusTPI Interfaces

Interface	Function	Parameters	Return	Example
`NexusTPI & init()`	Specify initialize step before executing other NexusTPI interface functions.	None	Return NexusTPICode: (int)	See Example 1
`NexusTPI & location(string)`	Specify the location of the communication target where the ContainerApp is running.	"aus" or "edge" (const std::string&)	NexusTPI &	
`NexusTPI & target(string)`	Specify the target RTDI Application.	Container Name (const std::string&)	NexusTPI &	
`NexusTPI & timeout(int)`	Set timeout for the communication. • minimum 1 • maximum 10 If the input parameter is outside this range, the upper and lower limits of this range are taken.	Timeout value in seconds (int)	NexusTPI &	
`send(String)`	Send data to the target application. Response is required.	The data to be sent (const std::string&)	NexusTPICode	See Example 2
`request(String)`	Initial request to the target application. Response is required.	Content of the request (JSON formatted const) std::string&	NexusTPICode	See Example 3
`getResponse()`	Get the response message. Should be used after "request" interface succeeds.	None	Response (std::string)	See Example 3

Example Code 1: init

```
NexusTPI& tpi = NexusTPI::getInstance();
int res = tpi.init();
cout << "NexusTPI init result:" << res << endl;
```

Example Code 2: send()

```
NexusTPI& tpi = NexusTPI::getInstance();

string data = "your data";

NexusTPICode res_send = tpi.target("container_tpi").location("aus").send(data);

cout << "res_send:" << res_send << endl;
```

Example Code 3: request() + getResponse()

```
NexusTPI& tpi = NexusTPI::getInstance();

string capacity = "10";
string cmd = "{\"key\": \"performance\", \"data\": \"" + capacity + "\"}";

NexusTPICode res_request = tpi.request(cmd);

cout << "res_request:" << res_send << endl;

if(res_request == NexusTPICode::SUCCEED)
{
  cout << tpi.getResponse() << endl;
}
```

Table 2-24. NexusTPICode Code Descriptions

Code(int)	Code(var)	Definition
0	`SUCCEED`	Communication succeeded
1	`COMMON_ERROR`	Common Error (Errors that are not specifically defined)
2	`TIMEOUT`	Communication time out
3	`TARGET_INVALID`	Target is invalid **NOTE:** This usually means that the target Application is not defined in the AppDescriptor file or the Application's configuration is illegal.
4	`CONNECT_FAILED`	Unable to connect to the target Application
5	CERTIFICATE_ERROR	Certificate related errors
6	LICENSE_INVALID	License is invalid

2.12.2 OneAPI

As the other endpoint of the bi-directional communication, the RTDI Application needs to use OneAPI library to receive communication requests from the test program. The following sections provide information on how the test program sends data to the RTDI Application, and how the test program can invoke a request to the RTDI Application.

2.12.2.1 Sending Data from the Test Program to RTDI Application

Before beginning, make sure the following requirements have been met:

- Nexus v2.1.0 or higher is installed
- ACS Edge Server has been deployed
- DNS of HostController is correctly configured
  (For example, /etc/hosts has the line: `{edge_ip} advantestcell.local`)

Procedure

1. Add NexusTPI lib into your device through SmarTest WorkCenter.
2. Import `nexus.tpi.NexusTPI` in test method code.
3. Set the correct target application name via NexusTPI.target(appName). This `appName` must be the same as "containers/name" field in the AppDescriptor file.
4. Build data in string.
5. Send this data via `NexusTPI.send()`.

Example 1: C++ version

```
import nexus.tpi.NexusTPI;

String appName = "yourApp";
String data    = "yourdata";

int res;  // result code

// send data to yourApp
res = NexusTPI.target(appName).send(data);

if (res == NexusTPI.SUCCEED) {
    // succeed
} else {
    // fail with code, do error handling
}
```

Example 2: Python version

```
class SampleMonitor(Monitor):
    def __init__(self):
        Monitor.__init__(self)


    # derive callback func for NexusTPI::send
    def consumeTPSend(self, tc, data):
        print(f"Received data from {tc.testerId}, length is {len(data)} ")

def main():
    myMonitor = SampleMonitor()
    Interface.registerMonitor(myMonitor)

    me = AppInfo()
    me.name = "sample"
    me.vendor = "adv"
    me.version = "1.1.0"

    NexusDataEnabled = True  # whether to enable Nexus Data Streaming and Control
    TPServiceEnabled = True  # whether to enable TPService for communication with NexusTPI
    res = Interface.connect(me, NexusDataEnabled, TPServiceEnabled)

    if res != 0:
        print(f"Connection request failed to initiate. code = {res}")
        sys.exit()


    print("Connection request initiated successfully.")

    signal.signal(signal.SIGINT, quit)
    signal.signal(signal.SIGTERM, quit)

    while True:
        signal.pause()

if __name__ == "__main__":
    main()
```

2.12.2.2 Test Program Invoking Request to RTDI Application

Before beginning, make sure the following requirements have been met:

- Nexus v2.1.0 or higher is installed
- ACS Edge Server has been deployed
- DNS of HostController is correctly configured
  (For example, /etc/hosts has the line: `{edge_ip} advantestcell.local`)

Procedure

1. Add NexusTPI lib into your device through SmarTest WorkCenter.
2. Import `nexus.tpi.NexusTPI` in test method code.
3. Set the correct target application name via NexusTPI.target(appName).

   This `appName` must be the same as "containers/name" field in the AppDescriptor file.
4. Build your request data to JSON format string.
5. Initial request to target application with NexusTPI.request(data).
6. Get response with `NexusTPI.getResonse()` when the request API returns `NexusTPI.SUCCEED`.

Example

```
import nexus.tpi.NexusTPI;

String appName = "app1";
String req_cmd = "{\"key\" : \"command_for_test\"}";
int t_out      = 1;  // timeout 1 second
int res;  // result code

// initial request to app1 with timeout 1 second
res = NexusTPI.target(appName).timeout(t_out).request(req_cmd);

if (res == NexusTPI.SUCCEED) {
    // succeed
    String response = NexusTPI.getResponse();
} else {
    // fail with code, do error handling
}
```

Example 2: OneAPI handle TPI request in application (Python version)

```
class SampleMonitor(Monitor):
    def __init__(self):
        Monitor.__init__(self)

    # derive callback func for NexusTPI::request
    def consumeTPRequest(self, tc, request):
        print(f"Received request from {tc.testerId}, command is {request}")
        jsonObj = json.loads(request)
        key  = jsonObj["key"]
        response = "your response"
        return response

def main():
    myMonitor = SampleMonitor()
    Interface.registerMonitor(myMonitor)

    me = AppInfo()
    me.name = "sample"
    me.vendor = "adv"
    me.version = "1.1.0"

    NexusDataEnabled = True  # whether to enable Nexus Data Streaming and Control
    TPServiceEnabled = True  # whether to enable TPService for communication with NexusTPI
    res = Interface.connect(me, NexusDataEnabled, TPServiceEnabled)

    if res != 0:
        print(f"Connection request failed to initiate. code = {res}")
        sys.exit()


    print("Connection request initiated successfully.")

    signal.signal(signal.SIGINT, quit)
    signal.signal(signal.SIGTERM, quit)

    while True:
        signal.pause()

if __name__ == "__main__":
    main()
```

2.13 Data Feed Forward

Data Feed Forward allows collection of test data from different test stages to be transmitted to other regions or OSAT's ACS Unified Servers. It also enables retrieval of DFF data across ACS Unified Servers for machine learning to improve production efficiency and quality. The diagram below illustrates the data feed forward concept.

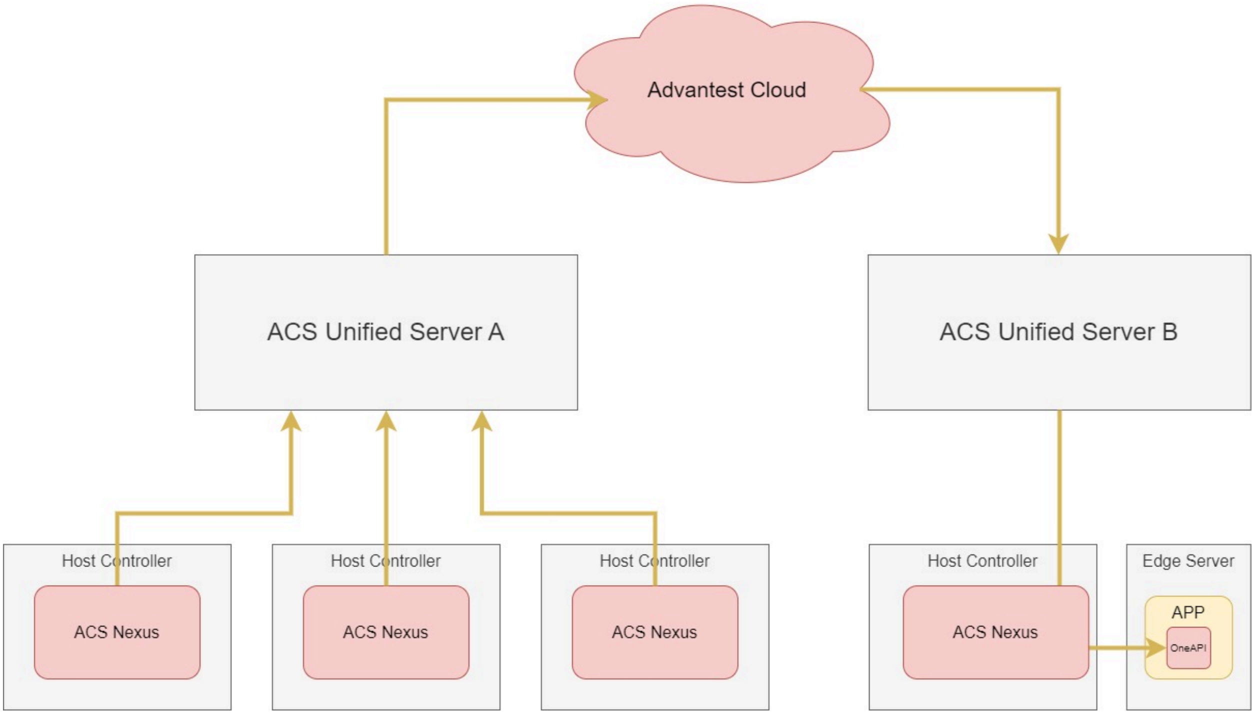


Figure 2-14. Data Feed Forward

2.13.1 Data Writing - NexusData

Data writing is implemented by ACS Nexus, which retrieves the raw data from Nexus directly, then forwards it to the ACS Unified Server. Data writing is disabled by default. To enable data writing, the `acs_nexus.ini` configuration must be modified as follows:

- [TestFloor_Server/Enabled] must be set to *true*.
- [TestFloor_Server/Brokers] must correspond to the ACS Unified Server configuration.

For more details, refer to the documentation for TestFloor_Server (see Table 2-13).

Any modifications to the configuration items will take effect after the next startup of ACS Nexus.

2.13.2 Data Writing – Customer Data from the Application

OneAPI provides standardized interfaces for Edge applications and ACS Unified Server applications (Python 3.9 - 3.11) to upload customer data to the ACS Unified Server (AUS). Follow the steps below to perform data writing:

Uploading DFF Data

1. Load and Register Schema

   Load the data schema from a file. OneAPI will automatically register it with AUS.

2. Set Properties

   Use the fluent interface provided by OneAPI to configure the properties for the next DFF data write.

3. Upload DFF Data

   Upload the DFF data to the ACS Unified Server. OneAPI supports registering callback functions to receive upload result notifications.

OneAPI Interfaces

OneAPI interfaces can be used to implement data writing functionality. For detailed information, refer to the relevant sections in the OneAPI documentation. For Python interfaces, see section 2.4.3.9.

2.13.3 Data Reading – By Application

Data reading is facilitated by OneAPI interfaces, allowing Edge applications and ACS Unified Server applications to query DFF data from the ACS Unified Server and provide customized or formatted datasets to users.

Reading NexusData

The following are prerequisites for reading NexusData:

- A unique identification **must** be defined for each device across the entire lifecycle of the device.
- The unique identification of each device **must** be put into the *STDF.PART_ID* or *STDF.PART_TXT* field for each insertion test.
- The application **must** able to get this "unique identification" (using customer's own mechanism) during testing, then put this as a key to query specific DFF data for the specific device.

Dataset

The query result dataset contains all Nexus data except for FTR and STR records. Devices with FTR/STR failures will not proceed to the next test insertion, making their data irrelevant for subsequent stages.

Data Structure in JSON Format

```
[ { “device_info”: […],  “ptr”: […],  “mpr”: […] } ]
```

Refer to the tables below for the object descriptions.

Table 2-25. "device_info" JSON Object Description

Field Name	Type	Description	Example
facility_id	string	Factory ID	my_facility_id
family_id	string	Family ID	my_family_id
floor_id	string	Floor ID	my_floor_id
hard_bin	integer	Hard bin number	2
head_number	integer	Test head number	1
id	UUID string	Unique identifier	xxxxxxxx-xxxx-xxxx-xxxx
job_name	string	Job name	my_job_name
job_revision	string	Job revision	my_job_revision
lot_id	string	Lot ID	my_lot_id
number_test	integer	Number of tests	1
operator_name	string	Operator name	my_operator_name
part_flag	string	Part flag	my_part_flag
part_id	string	Part ID	my_part_id
part_text	string	Part text	my_part_text
part_type	string	Part type	my_part_type
process_id	string	Process ID	my_process_id
serial_number	string	Serial number	my_serial_number
setup_timestamp	timestamp	Setup timestamp (µs)	1748940987950776
site_number	integer	Site number	0
soft_bin	integer	Soft bin number	3
sublot_id	string	Sub-lot ID	my_sublot_id
test_step_code	string	Test step code	my_test_step_code
test_temperature	string	Test temperature	25C
test_time	integer	Test duration	123456
test_type	string	Test type	WAFER_TEST
testend_timestamp	timestamp	Test end timestamp (µs)	1748941987950776
tester_id	string	Tester ID	my_tester_id
tester_type	string	Tester type	my_tester_type
testeros_type	string	Tester OS type	my_testeros_type
testeros_version	string	Tester OS version	my_testeros_version
testprogram_name	string	Test program name	my_testprogram_name
teststart_timestamp	timestamp	Test start timestamp (µs)	1748940987950776
timezone	number	Time zone	+8
wafer_id	string	Wafer ID	my_wafer_id
x_coord	integer	X coordinate	123
y_coord	integer	Y coordinate	456

Table 2-26. "ptr" JSON Object Description

Field Name	Type	Description	Example
alarm_id	string	Alarm ID	my_alarm_id
device_info_id	string	Unique device identifier	xxxxxxxx-xxxx-xxxx-xxxx
head_number	integer	Test head number	1
high_limit	float	High limit	3.3
high_limit_format	string	High limit format	###.###
high_spec	integer	High spec	3000
highlimit_scaling	integer	High limit scaling	-3
id	UUID string	Unique identifier	xxxxxxxx-xxxx-xxxx-xxxx
lot_id	string	Lot ID	lot_id
low_limit	float	Low limit	0.0
low_limit_format	string	Low limit format	###.###
low_spec	integer	Low spec	0
lowlimit_scaling	integer	Low limit scaling	-3
measurement_name	string	Measurement name	I_DD
optional_flag	string	Optional flag	Y
param_flag	string	Parameter flag	N
part_id	string	Part ID	my_part_id
part_text	string	Part text	my_part_text
result	float	Measurement result	1.23
result_format	string	Result format	###.###
result_scaling	integer	Result scaling	-3
save_at	timestamp	Without time zone	1748940987950776
site_number	integer	Site number	0
test_flag	string	Test flag	P
test_number	integer	Test number	1001
test_suite	string	Test suite name	my_MainSuite
test_text	string	Test description	my_Voltage measurement
units	string	Measurement units	V
wafer_id	string	Wafer ID	my_wafer_id
x_coord	integer	X coordinate	10
y_coord	integer	Y coordinate	20

Table 2-27. "mpr" JSON Object Description

Field Name	Type	Description	Example
id	UUID string	Unique identifier	xxxxxxxx-xxxx-xxxx-xxxx
device_info_id	string	Unique device identifier	xxxxxxxx-xxxx-xxxx-xxxx
lot_id	string	Lot ID	my_lot_id
wafer_id	string	Wafer ID	my_wafer_id
part_id	string	Part ID	my_part_id
x_coord	integer	X coordinate	100
y_coord	integer	Y coordinate	200
part_text	string	Part text	my_part_text
test_number	integer	Test number	2001
site_number	integer	Site number	0
test_flag	string	Test flag	F
rslt_cnt	integer	Result count	8
test_text	string	Test description	Multi-site measurement
lo_limit	float	Low limit	0.0
hi_limit	float	High limit	5.0
units	string	Measurement units	V
param_flag	string	Parameter flag	Y
rtn_icnt	integer	Return index count	8
rtn_stat	string	Return status	PASS
rtn_rslt	Float []	Returned results	[1.1, 1.2, 1.3, ..., 1.8]
alarm_id	string	Alarm ID	alarm_id
opt_flag	string	Optional flag	N
result_scaling	integer	Result scaling	-3
lowlimit_scaling	integer	Low limit scaling	-3
highlimit_scaling	integer	High limit scaling	-3
start_in	integer	Start index	0
incr_in	integer	Index increment	1
rtn_indx	string	Return indices	0,1,2,3,4,5,6,7
units_in	string	Units of increment	V
c_resfmt	string	Result format	###.###
c_llmfmt	string	Low limit format	###.###
c_hlmfmt	string	High limit format	###.###
lo_spec	float	Low spec	0.0
hi_spec	float	High spec	5.0
pin_name	string []	Pin names	["PIN1", "PIN2", ..., "PIN8"]
test_suite	string	Test suite name	my_Suite1
measurement_name	string	Measurement name	Vdd
save_at	timestamp	Without time zone	1748940987950776

Reading Customer Data

All data uploaded by users through the DFF-Data-Writing feature can be retrieved during data reading.

Data Reading Interfaces

Data queries are executed using OneAPI query interfaces. For details, refer to the relevant OneAPI sections.

For C++ interfaces, see sections 2.3.3.7 (QueryResponse) and 2.3.3.8 (DFFData)

For Python interfaces, see sections 2.4.3.7 (QueryResponse) and 2.4.3.8 (DFFData)

2.13.4 Data Writing – Customer Data from Test Program

NexusTPI provides a standard interface enabling test programs to upload real-time data to the ACS Unified Server.

NOTE: Nexus v2.3.0 supports only Java version for SmarTest 8.

2.13.4.1 Load and Register Schema

Register a schema template file in JSON format for data verification.

Code Example

```
import nexus.tpi.NexusTPI;

int result = NexusTPI.DFF.importSchema("./RTDI/schema.json");
if (result == NexusTPI.SUCCEED) {
     //Schema import successful"
}
if (result == NexusTPI.SCHEMA_LOAD_FAIL) {
     //Schema file failed to load, maybe it's not in JSON format
}
if (result == NexusTPI.SCHEMA_REGIST_FAIL) {
     //Schema registration failed, maybe server is down
}
if (result == NexusTPI.SCHEMA_SAME) {
     //Schema with same $id has been already registered by this Program
}
```

2.13.4.2 Set Properties

Configure key-value pairs for subsequent data uploads. These key-value pairs will override the default values defined in the schema template.

Code Example

```
import nexus.tpi.NexusTPI;

NexusTPI.DFF.set("ecid", String.format("DATA_IS_ECID_%d_x1_y1", 1))
            .set("filename","abcde.dff.json")
            .set("data_type", "DFF")
            .set("timestamp", "yyyy-MM-dd HH:mm:ss")
            .set("lot_id", "ABCDE.23")
            .set("program_name", "A5.A0.CP1.E.S.25C.1A.02")
            .set("site_num", 1)
            .set("test_stage", "FT")
            .set("schema_version", "1.0.0");
```

2.13.4.3 Upload Data

Upload data asynchronously to the ACS Unified Server.

NOTE: Java test programs do not need to handle asynchronous processing when calling the NexusTPI layer. The NexusTPI implementation handles asynchronous data transmission internally.

Code Example

```
import nexus.tpi.NexusTPI;

  int result = NexusTPI.DFF.upload("dffdata = xxxxxxx");
  if (result == NexusTPI.DFF_CONNECTED) {
        //Properties are valid, and start to upload, it doesn't mean upload completed
  }
  if(result == NexusTPI.DFF_CONNECT_FAIL) {
        //Failed to establish data-upload connection
  }
  if(result == NexusTPI.DFF_DATA_EMPTY) {
        //Data is empty
  }
  if(result == NexusTPI.DFF_PROPERTY_INVALID) {
        //Properties invalid to schema
  }
```

2.13.4.4 Register User-Defined Callback Function

Register a user-defined callback function to receive upload results.

NOTE: There are three distinct methods to register a callback function.

Code Example

```
import nexus.tpi.NexusTPI;
import nexus.tpi.Callback;

static class DFF_Writing_Suite{
    public static void onUploadComplete(int result,
                                        String properties,
                                        String data) {
    }
}
NexusTPI.DFF.regUploadCallback(DFF_Writing_Suite::onUploadComplete);//Register
Callback Method 1

NexusTPI.DFF.regUploadCallback(new Callback() {
    @Override
    public void call(int result, String properties, String data) {
        //Register Callback Method 2
    }
});

NexusTPI.DFF.regUploadCallback((result0, properties, data) -> {
    //Register Callback Method 3
});
```

2.13.5 Data Reading – by Test Program

NexusTPI provides a standard interface that allows the test program to query DFF data form the ACS Unified Server.

NOTE: Nexus v2.3.0 supports only Java version for Smartest8.

2.13.5.1 Query Job-ID

Define custom query conditions to retrieve the Job ID.

Code Example

```
import nexus.tpi.NexusTPI;

NexusTPI.Result result = NexusTPI.DFF.query("query condition");
if (result.code == NexusTPI.SUCCEED) {
    String jobID = result.body[0];
} else {
    String errMsg = result.body[0];
}
```

2.13.5.2 Get Job Result

Use the Job ID obtained in the previous step to fetch the corresponding job result.

Code Example

```
import nexus.tpi.NexusTPI;

NexusTPI.Result result = NexusTPI.DFF.getJobResult("jobID", "JSON or CSV");
if (result.code == NexusTPI.JOB_COMPLETED) {
  boolean hasError = !result.body[0].equals("");
  if (!hasError) {
    String dffdata = result.body[1];
  }
}
if (result.code == NexusTPI.JOB_FAILED) {//job fail}
if (result.code == NexusTPI.JOB_RUNNING) {//please wait job completed}
```

2.14 Configurable Nexus Services

Nexus services and support can be enabled or disabled by modifying the acs_nexus.ini configuration file, which is located at `/opt/acs/nexus/conf/acs_nexus.ini`.

The Nexus Services contents of the file are as follows:

```
[Nexus_Services]
All_Services=true
Monitoring_Service=true
```

Nexus_Services.All_Services

The default is *true*, which enables all services. Can optionally be set as false to disable all services.

Nexus_Services.Monitoring_Service

The default is *true*, which enables monitoring service. Can optionally be set as false to disable monitoring service.

ACS Nexus must be restarted for any changes to the acs_nexus.ini file to take effect. ACS Nexus should be stopped and started manually.

2.15 Nexus License Check Modes

Nexus supports different license check modes. The license check mode is defined in the Nexus configuration file (acs_nexus.ini), located at `/opt/acs/nexus/conf/acs_nexus.ini`.

To change the license check mode, modify the [Mode] entry in the Nexus configuration file. See Table 2-28 below for license mode details.

NOTE: If the license check mode is set to any value than 1 or 0, Nexus will not check the license and all Nexus features will be disabled.

Table 2-28. License Check Modes

Mode	Description	Enabled Features	Disabled Features
1 (default)	When the Smartest is ready, Nexus checks the license. Once the license check succeeds, all features will be enabled. When Smartest terminates, Nexus will release license.	All features enabled	None
0	When Smartest is ready, Nexus will NOT check the license. License validation is triggered only when TPI is called. The TPI will be enabled if the license is valid, while other features remain disabled. When Smartest terminates, Nexus will release license.	• Custom Data Feed Forword • Bi-directional communication	• Lifecycle management for Edge APP (HC APP) • Steaming Data • Test Cell Control • Standard Data Feed Forward • Fast API

3. ACS Container Hub

The ACS Container Hub is the artifact repository and distribution hub for Advantest Cloud workloads such as ACS Edge Server container images. It is used to manage private and published App artifacts. The ACS Container Hub provides a fully OCI-compliant Container Registry and implements Docker Registry HTTP API v2. It also offers a web GUI for convenient management of artifact projects.

3.1 ACS Container Hub Registration

This section provides step-by-step instructions for the registration process that must be completed for access to the ACS Container Hub.

NOTE: The registration process should be completed prior to RTDI installation, as access to the ACS Container Hub is required to complete the installation.

3.1.1 Prerequisite

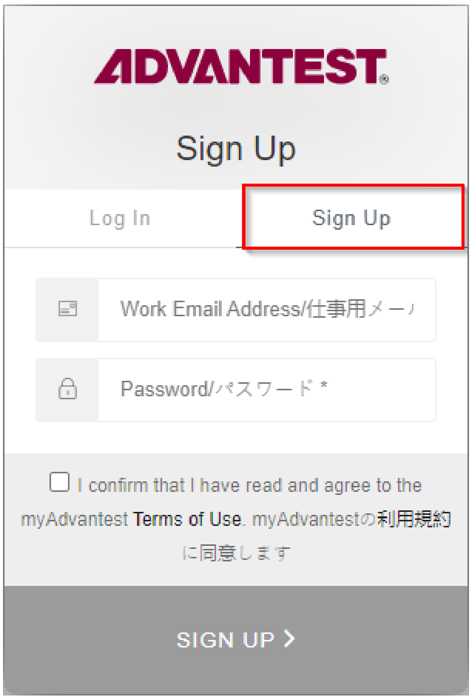
Prior to registering for ACS Container Hub access, an order must have been placed and signed between the customer and Advantest. A customer representative must also be designated to server as the Entitlement Owner. This is the person who will be granted access to an Advantest IT system (myAdvantest Entitlement System) where the order and the respective ACS Container Hub Entitlement can be viewed and managed. The Entitlement Owner is responsible for granting end-users access to the ACS Container Hub.

3.1.2 Registering myAdvantest Accounts

The designated Entitlement Owner (and any additional users) must have an active myAdvantest account to log into the ACS Container Hub (https://my.advantest.com). This process can be started during RTDI installation, but for a smoother installation experience, it is best to begin registration prior to installation. To complete registration, the Entitlement Owner must do the following:

1. Open a web browser and visit https://my.advantest.com.

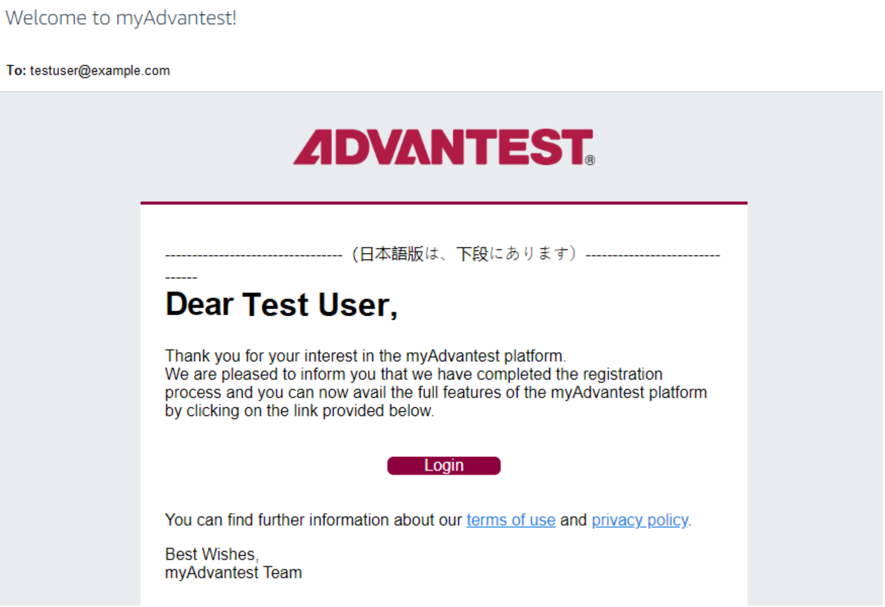
2. Click on **Sign Up**. Enter the required email address, choose a password, agree with the myAdvantest Terms of Use, then click on **SIGN UP** at the bottom.



□ I confirm that I have read and agree to the myAdvantest Terms of Use. myAdvantestの利用規約に同意します

Figure 3-1. Log In / Sign Up Dialog

3. Follow and complete the registration process. The account is verified and activated within a few days, at which point a welcome email is sent, as illustrated in the example below.



Welcome to myAdvantest!

To: testuser@example.com

-------------------------------- （日本語版は、下段にあります） -------------------------
------

Dear Test User,

Thank you for your interest in the myAdvantest platform.
We are pleased to inform you that we have completed the registration process and you can now avail the full features of the myAdvantest platform by clicking on the link provided below.

You can find further information about our terms of use and privacy policy.

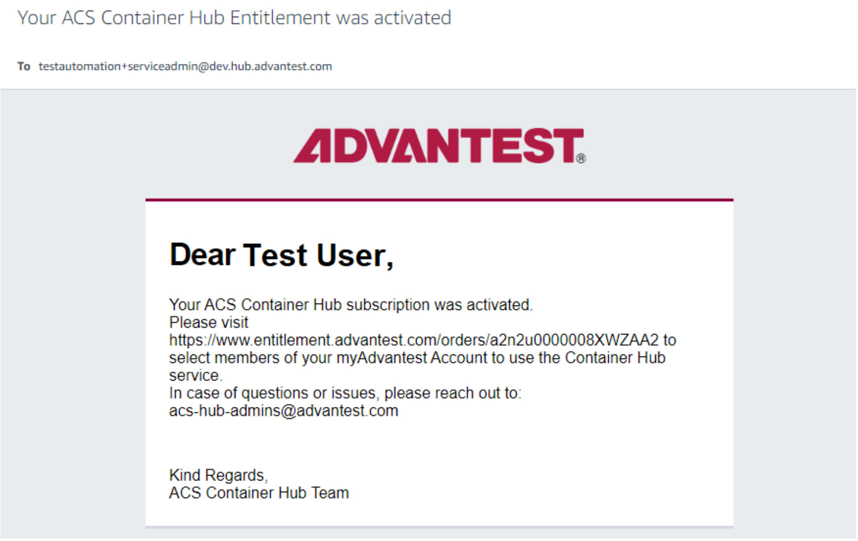
Best Wishes,
myAdvantest Team

Figure 3-2. Welcome Email Example

4. After receiving the Welcome email, the Entitlement Owner must contact the Advantest project owner (via email) to request service Entitlement.

   The request should include the Entitlement Owners email address (highlighted) among an initial list of additional user email addresses.

5. Within a few days, the Entitlement is created and activated. A notification email is then sent to the Entitlement Owner, as shown in the example below.



Your ACS Container Hub Entitlement was activated

To testautomation+serviceadmin@dev.hub.advantest.com

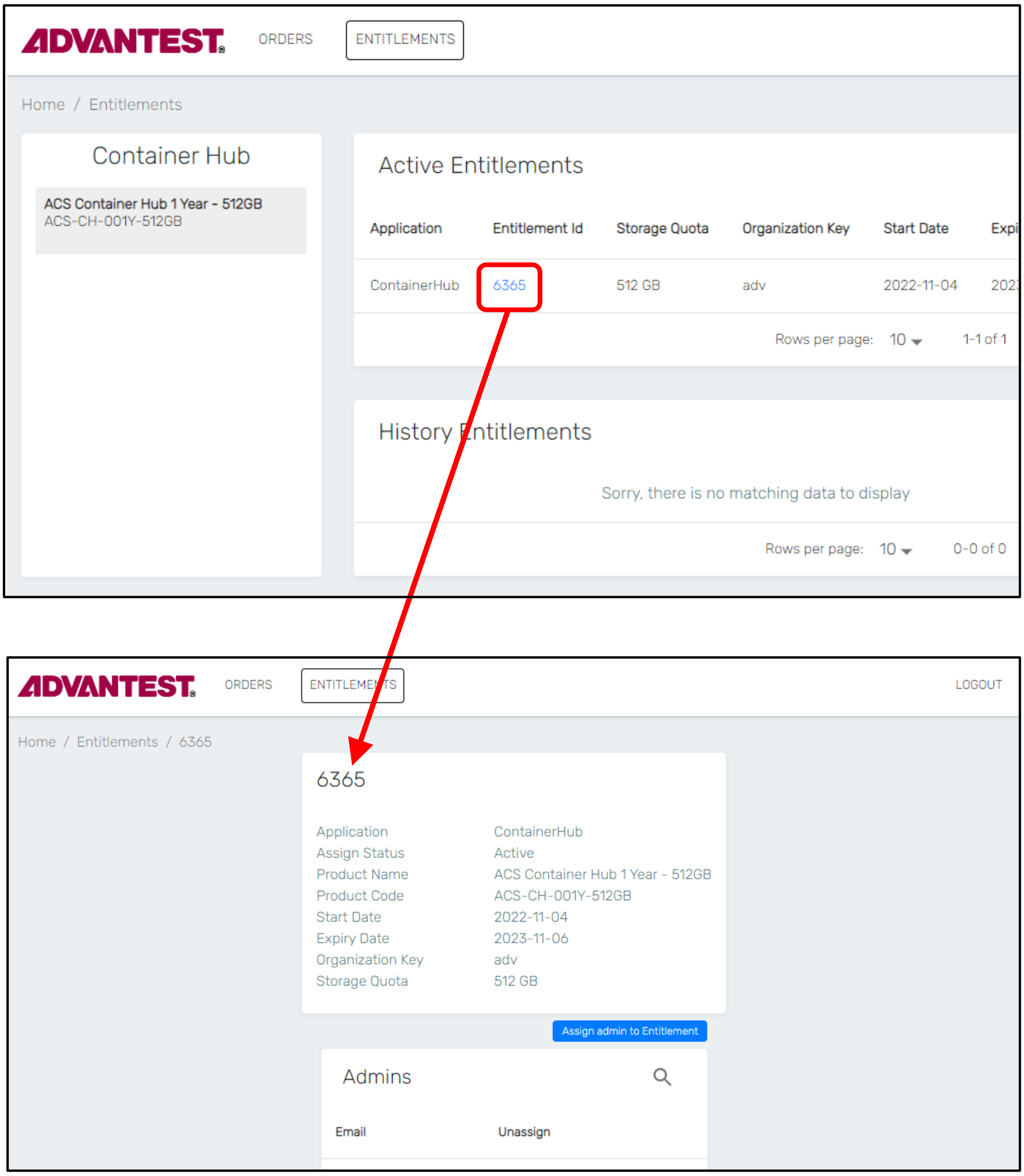
Dear Test User,

Your ACS Container Hub subscription was activated.
Please visit https://www.entitlement.advantest.com/orders/a2n2u0000008XWZAA2 to select members of your myAdvantest Account to use the Container Hub service.
In case of questions or issues, please reach out to: acs-hub-admins@advantest.com

Kind Regards,
ACS Container Hub Team

Figure 3-3. Entitlement Email Example

6. After receiving the Entitlement activation email, the Entitlement Owner must visit https://www.entitlement.advantest.com, log in, and navigate to the details page of the Active Entitlement for the Container Hub application, as shown in the examples below.


ADVANTEST
ORDERS
ENTITLEMENTS
Home / Entitlements
Container Hub
ACS Container Hub 1 Year - 512GB
ACS-CH-001Y-512GB
LOGOUT
Home / Entitlements / 6365
Assign admin to Entitlement

Active Entitlements

Application	Entitlement Id	Storage Quota	Organization Key	Start Date	Expi
ContainerHub	6365	512 GB	adv	2022-11-04	202

Rows per page: 10 1-1 of 1

History Entitlements

Sorry, there is no matching data to display

Rows per page: 10 0-0 of 0

6365

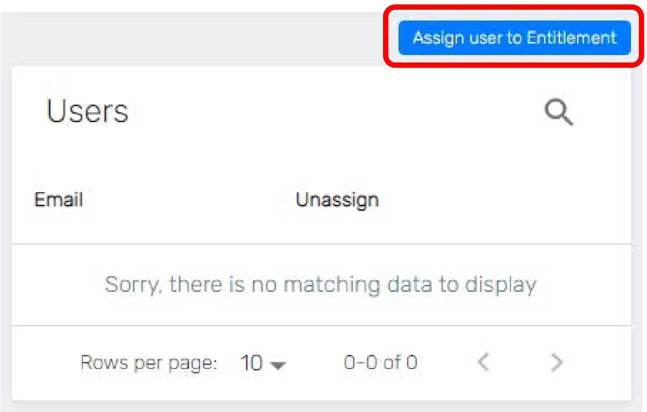
Application	ContainerHub
Assign Status	Active
Product Name	ACS Container Hub 1 Year - 512GB
Product Code	ACS-CH-001Y-512GB
Start Date	2022-11-04
Expiry Date	2023-11-06
Organization Key	adv
Storage Quota	512 GB

Admins

Email	Unassign

Figure 3-4. Active Entitlement Details Page

7. On the Active Entitlement details page, the Entitlement Owner is already assigned to the Admin role. However, the Entitlement Owner must click the **Assign user to Entitlement** button to assign any additional desired myAdvantest users from the customer organization to the User role.



Sorry, there is no matching data to display

Figure 3-5. Assign User Button

8. Every user that was added to the Entitlement will receive another email notification that access to ACS Container Hub has been granted. After receiving that email, the user can visit https://registry.advantest.com in a web browser and log in with their myAdvantest credentials.

3.2 Connecting ACS Container Hub to an ACS Unified Server

For every RTDI installation, it is required to set up replication from the ACS Container Hub to a local Container Registry running on the Unified Server. The replication ensures that all Container Hub contents that are supposed to run on RTDI are made available on the local RTDI installation.

Replication requires a Client Credential with permissions to pull contents from all projects that shall be used with the RTDI installation. The Client Credential must be created by the end customer and handed over to the Advantest Field Service Engineer who is configuring the Container Registry for the ACS Unified Server during a new RTDI installation.

NOTE: This process must be completed prior to installing a Unified Server during a new RTDI installation.

3.2.1 Creating Client Credential for Replication

NOTE: As a prerequisite, you must have at least one project that contains container images and shall be replicated. See Create a New Project for details on how to create new projects.

Credential Management Strategies

Depending on your technical and security requirements, you can choose between different strategies for managing Client Credentials for replication.

Strategy	Description	Pros	Cons
Single Credential	A single Client Credential which is used for all Unified Servers and RTDI installations.	Easier to manage. Only one credential needs to be changed when new projects are added.	If a credential gets compromised, deactivating/deleting it will affect all RTDI installations.
Credential per Unified Server	One Client Credential per Unified Server and RTDI installation.	Greater security control. If a credential gets compromised, deactivating/deleting it will affect only one RTDI installation. Replicated projects can be different per RTDI installation.	Higher management effort New projects must be added to each credential.

Steps to Create Client Credential for Replication

NOTE: Make sure to update this client credential and add any new projects to it that are required by the RTDI installation. Otherwise, project contents cannot be replicated, and ACS container starts will fail.

1. Choose a name that matches your management strategy and makes it easy to recognize the purpose of the credential in the future. You may use the description field to leave additional information about how/where the credential is used.

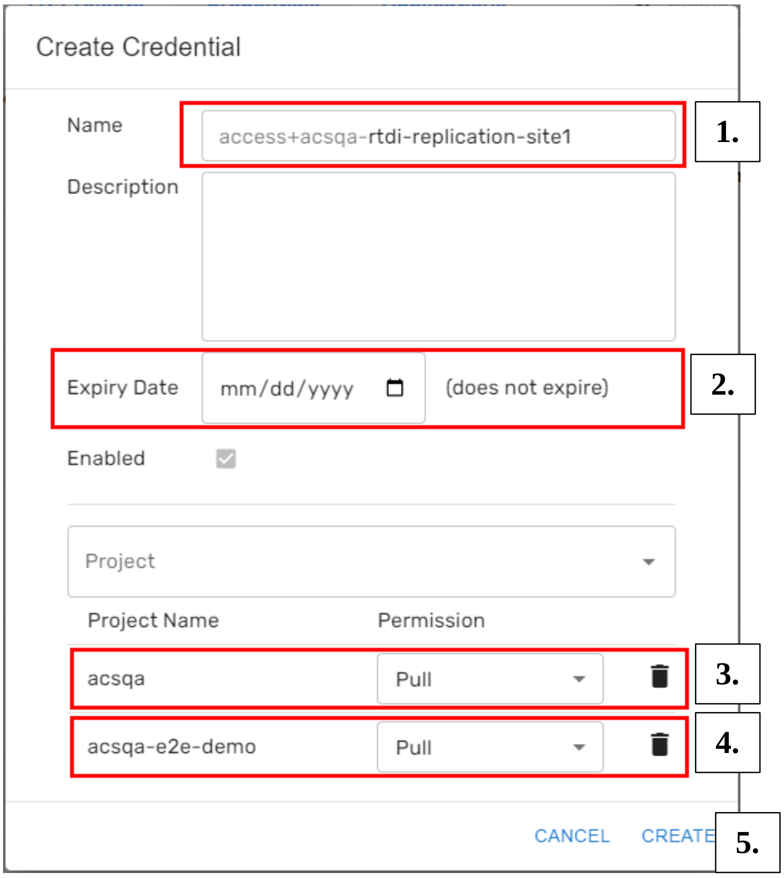
2. Make sure the credential does not expire, even if you have good reason to set an expiration date.

3. Make sure to add the system project that stores Application Descriptors with pull permissions.

4. Make sure to add any other project that stores container images which shall be replicated.

5. Click **CREATE**.

6. In the result popup, click the download icon to store the Client Credential as a JSON file. This JSON file must be provided to the Advantest Field Service engineer that sets up the Unified Server for the targeted RTDI installation.



Create Credential

Name: access+acsqa-rtdi-replication-site1

Description:

Expiry Date: mm/dd/yyyy (does not expire)

Enabled ☑

Project

Project Name	Permission
acsqa	Pull
acsqa-e2e-demo	Pull

Figure 3-6. Create Client Credential Dialog

Application Descriptor

The Application Descriptor system project is the project equal to the ID of your Container Hub Organization. You can find your Organization ID in the Customer Portal user menu:

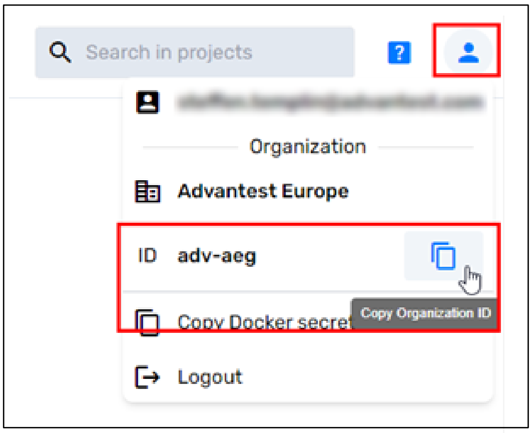


Figure 3-7. Application Descriptor - Customer Portal User Menu

3.3 ACS Container Hub User Interface

This section provides detailed information on how to navigate and use the ACS Container Hub UI.

3.3.1 Log In to ACS Container Hub

To access the ACS Container Hub, login information is required. Prior to logging in, both of the following conditions must be met:

- User must have ACS Container Hub service access
- User must be assigned to a valid entitlement (administrator or user)

If ACS Container Hub service access or entitlement assignment has not been acquired, contact your Advantest representative. If both of the above conditions are satisfied, log in to the ACS Container Hub by opening a browser and going to https://registry.advantest.com/.

In the ACS Container Hub Login screen, enter valid myAdvantest credentials and click **LOG IN >**.

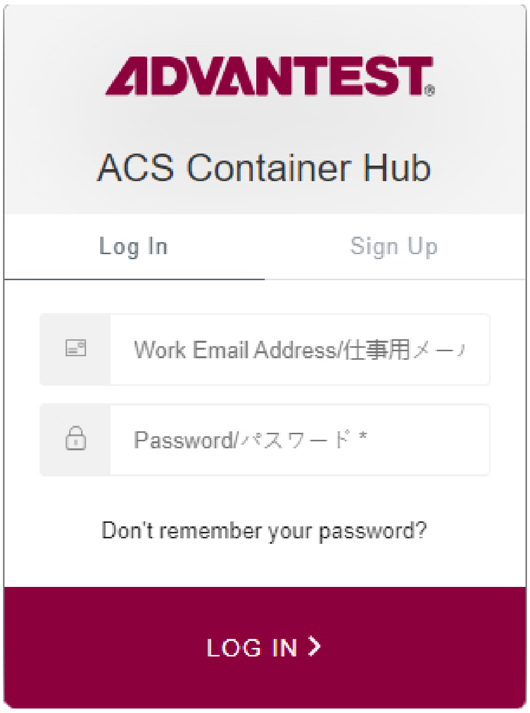
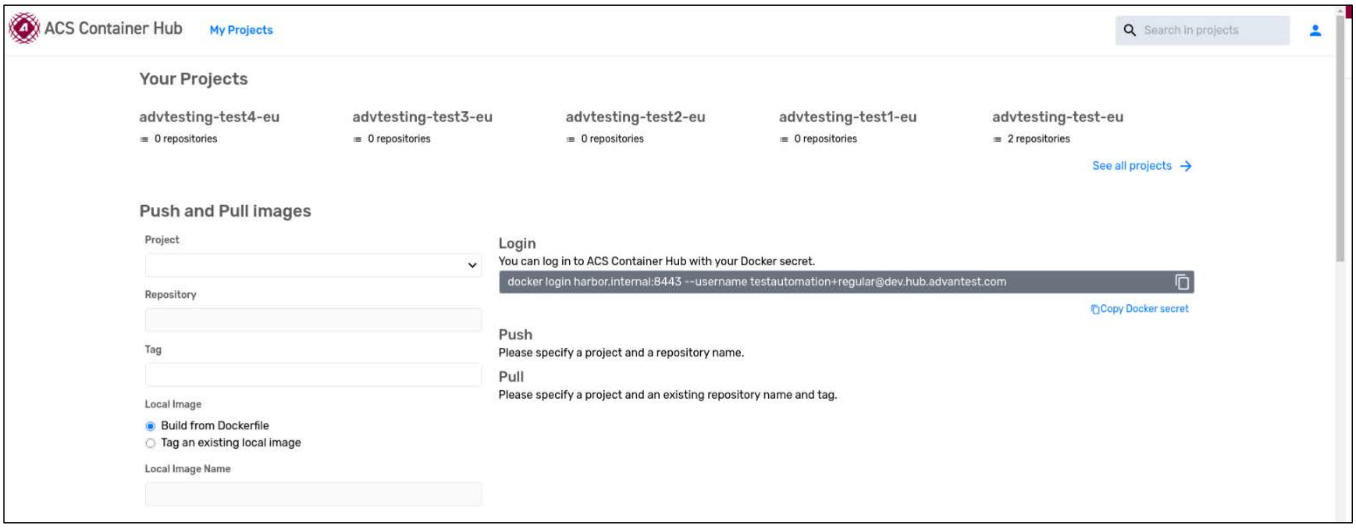


Figure 3-8. ACS Container Hub Login

After logging in, the ACS Container Hub homepage displays as shown below. The projects listed under "Your Projects" will vary based on which projects users can access and which projects have been accessed recently.



ACS Container Hub   My Projects   Search in projects

Your Projects

advtesting-test4-eu
0 repositories

advtesting-test3-eu
0 repositories

advtesting-test2-eu
0 repositories

advtesting-test1-eu
0 repositories

advtesting-test-eu
2 repositories

See all projects →

Push and Pull images

Project

Repository

Tag

Local Image
● Build from Dockerfile
○ Tag an existing local image

Local Image Name

Login
You can log in to ACS Container Hub with your Docker secret.

```
docker login harbor.internal:8443 --username testautomation+regular@dev.hub.advantest.com
```

Copy Docker secret

Push
Please specify a project and a repository name.

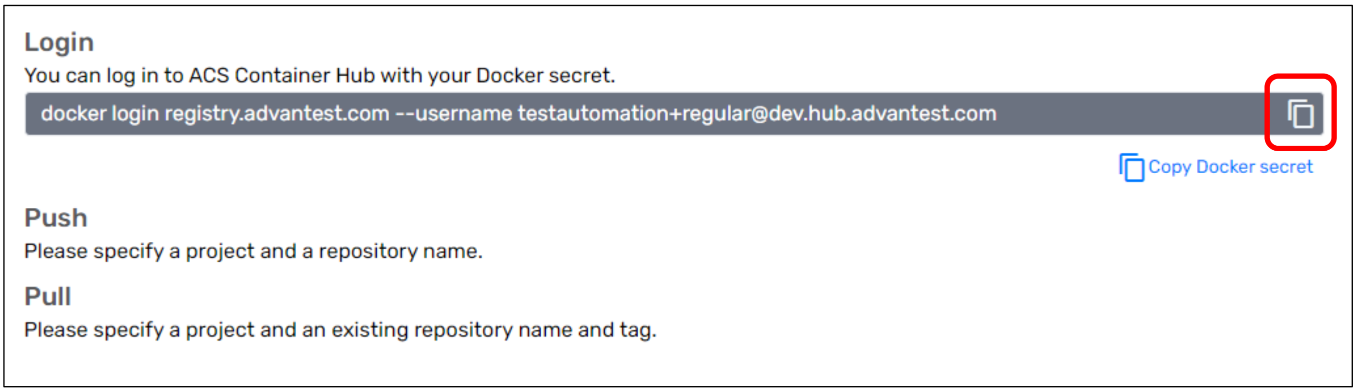
Pull
Please specify a project and an existing repository name and tag.

Figure 3-9. ACS Container Hub Homepage

3.3.2 Log in to Docker Command Line

To push and pull images to/from the ACS Container Hub, docker command line login is required. A docker wizard is available on the ACS Container Hub homepage to facilitate easy copy of docker tag, build, login, and push/pull commands. To log in to the docker command line, do the following:

1. Copy the login command using the copy icon, as shown below.



Login
You can log in to ACS Container Hub with your Docker secret.

```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```

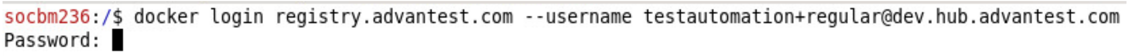
Copy Docker secret

Push
Please specify a project and a repository name.

Pull
Please specify a project and an existing repository name and tag.

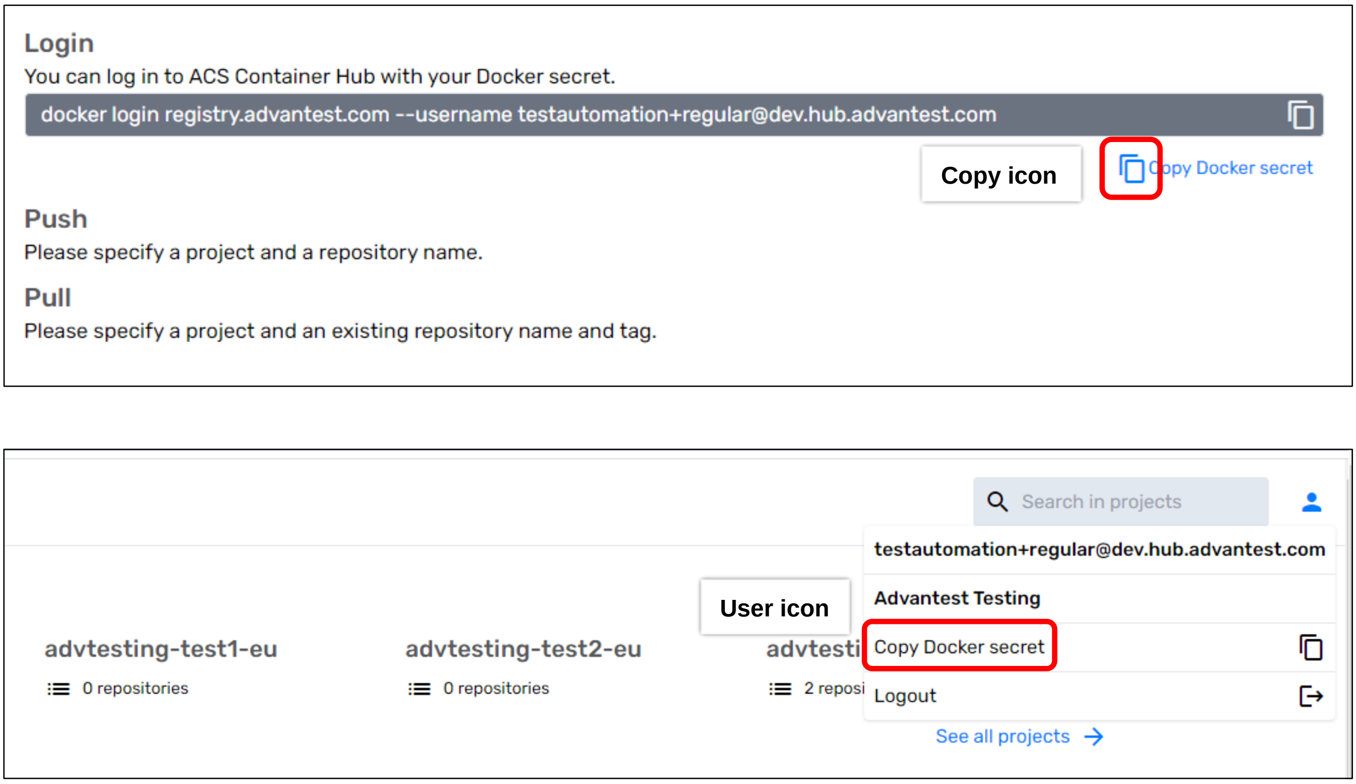
Figure 3-10. Copy Login Command

2. Open a command terminal on the Host workstation and paste the command copied in the previous step, then press Enter. A password is requested, as shown below.



```
socbm236:/$ docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
Password: █
```

3. Copy the docker password using the copy icon beside "Copy Docker secret." The docker password can alternatively be copied by clicking the User Icon and clicking "Copy Docker secret."



Login

You can log in to ACS Container Hub with your Docker secret.

```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```

Push

Please specify a project and a repository name.

Pull

Please specify a project and an existing repository name and tag.

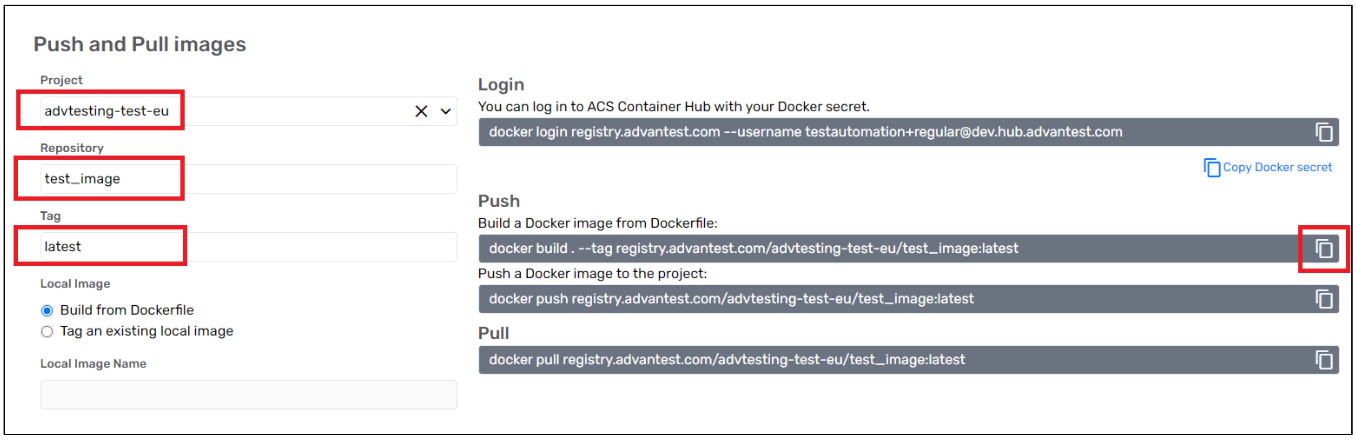
Figure 3-11. Copy Password

4. In the command line, paste the password copied in previous step and press Enter, as shown in the example below.

```
socbm236:/$ docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
Password:
WARNING! Your password will be stored unencrypted in /home/desharma/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
socbm236:/$ █
```

3.3.3 Docker Build and Push

1. To build the docker image with the desired tag, use the docker wizard and select the Project, Repository, and Tag.



Push and Pull images

Project
advtesting-test-eu

Repository
test_image

Tag
latest

Local Image
● Build from Dockerfile
○ Tag an existing local image

Local Image Name

Login
You can log in to ACS Container Hub with your Docker secret.
```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```
Copy Docker secret

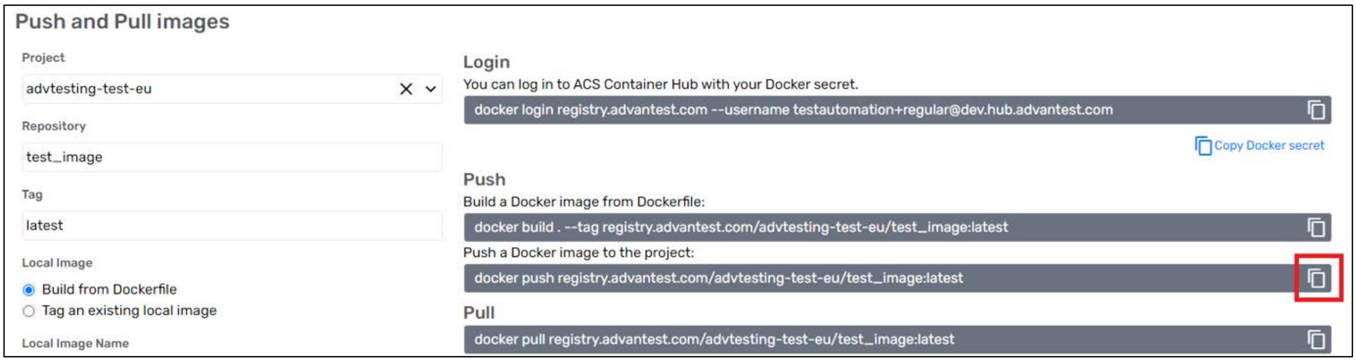
Push
Build a Docker image from Dockerfile:
```
docker build . --tag registry.advantest.com/advtesting-test-eu/test_image:latest
```
Push a Docker image to the project:
```
docker push registry.advantest.com/advtesting-test-eu/test_image:latest
```
Pull
```
docker pull registry.advantest.com/advtesting-test-eu/test_image:latest
```

Figure 3-12. Building from Dockerfile

2. Copy the build command using the copy icon for the *Build a Docker image from Docker file* (as shown above), then build the docker image by executing the copied build command.

```
Successfully built c0fac4eb48b0
Successfully tagged registry.advantest.com/advtesting-test-eu/test image:latest
```

3. After successful build, copy the docker push command from the wizard.



Push and Pull images

Project
advtesting-test-eu

Repository
test_image

Tag
latest

Local Image
● Build from Dockerfile
○ Tag an existing local image

Local Image Name

Login
You can log in to ACS Container Hub with your Docker secret.
```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```
Copy Docker secret

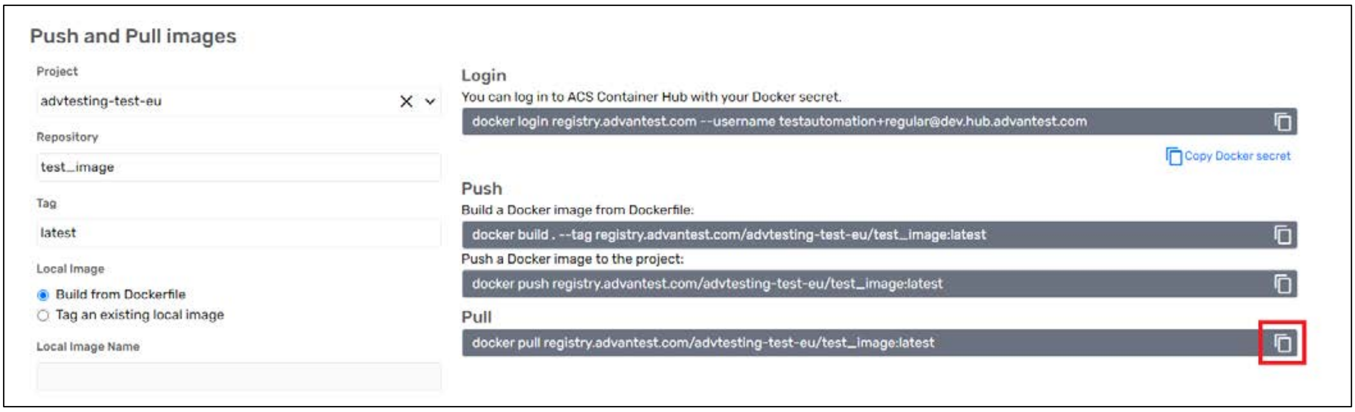
Push
Build a Docker image from Dockerfile:
```
docker build . --tag registry.advantest.com/advtesting-test-eu/test_image:latest
```
Push a Docker image to the project:
```
docker push registry.advantest.com/advtesting-test-eu/test_image:latest
```
Pull
```
docker pull registry.advantest.com/advtesting-test-eu/test_image:latest
```

Figure 3-13. Copy Docker Push Command

4. Execute the command and it should push the image to the Container Hub.

3.3.4 Docker Pull

1. To pull the docker image with the desired tag, use the docker wizard and select the Project, Repository, and Tag. Once selected, copy the docker pull command from the wizard.



Push and Pull images

Project: advtesting-test-eu

Repository: test_image

Tag: latest

Local Image
● Build from Dockerfile
○ Tag an existing local image

Local Image Name:

Login

You can log in to ACS Container Hub with your Docker secret.

```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```

Push

Build a Docker image from Dockerfile:

```
docker build . --tag registry.advantest.com/advtesting-test-eu/test_image:latest
```

Push a Docker image to the project:

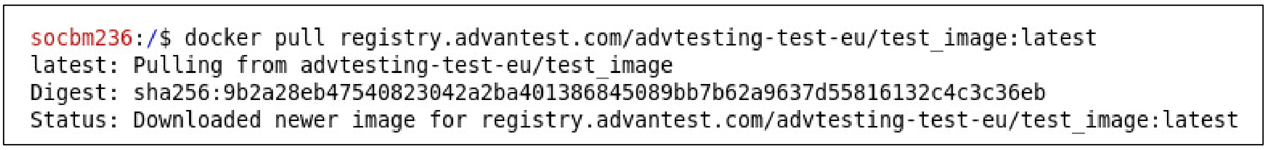
```
docker push registry.advantest.com/advtesting-test-eu/test_image:latest
```

Pull

```
docker pull registry.advantest.com/advtesting-test-eu/test_image:latest
```

Figure 3-14. Pulling from Dockerfile

2. Execute the command and it should pull the image from the ACS Container Hub.



```
socbm236:/$ docker pull registry.advantest.com/advtesting-test-eu/test_image:latest
latest: Pulling from advtesting-test-eu/test_image
Digest: sha256:9b2a28eb47540823042a2ba401386845089bb7b62a9637d55816132c4c3c36eb
Status: Downloaded newer image for registry.advantest.com/advtesting-test-eu/test_image:latest
```

3.3.5 Tag Existing Local Image

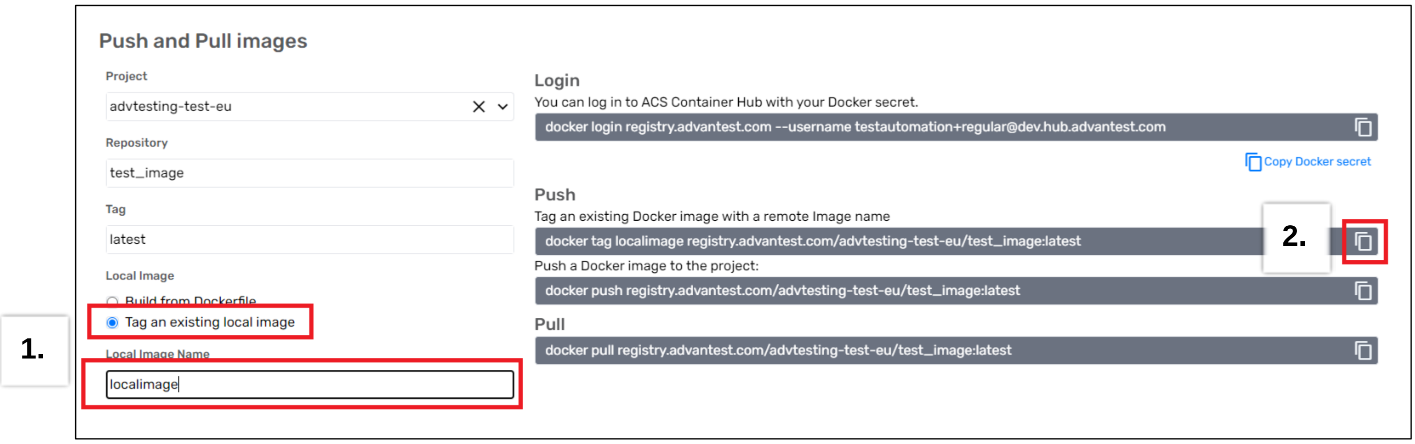
To push a locally stored image onto the ACS Container Hub for downstream distribution, the image must first be tagged with its final location inside the Container Hub Container Registry. To do this, a new tag for the image must be set that matches this full-qualified name (registry.advantest.com/project-name/repo-name:tag).

To tag the existing docker image, which is present on the Host workstation, use the docker wizard to do the following:

1. Select the option 'Tag an existing local image" and input the name of local image.

2. Copy the command using the copy icon for *Tag an existing Docker image with a remote Image name*.

3. Execute the command on docker command line and it will tag the local image with the correct host and tag.



Push and Pull images

Project: advtesting-test-eu

Repository: test_image

Tag: latest

Local Image

○ Build from Dockerfile

● Tag an existing local image

Local Image Name: localimage

Login

You can log in to ACS Container Hub with your Docker secret.

```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```

Copy Docker secret

Push

Tag an existing Docker image with a remote Image name

```
docker tag localimage registry.advantest.com/advtesting-test-eu/test_image:latest
```

Push a Docker image to the project:

```
docker push registry.advantest.com/advtesting-test-eu/test_image:latest
```

Pull

```
docker pull registry.advantest.com/advtesting-test-eu/test_image:latest
```

Figure 3-15. Tagging a Local Image

3.3.6 Project Search

1. To search for a project, click on the link for either **My Projects** or **See All Projects**.

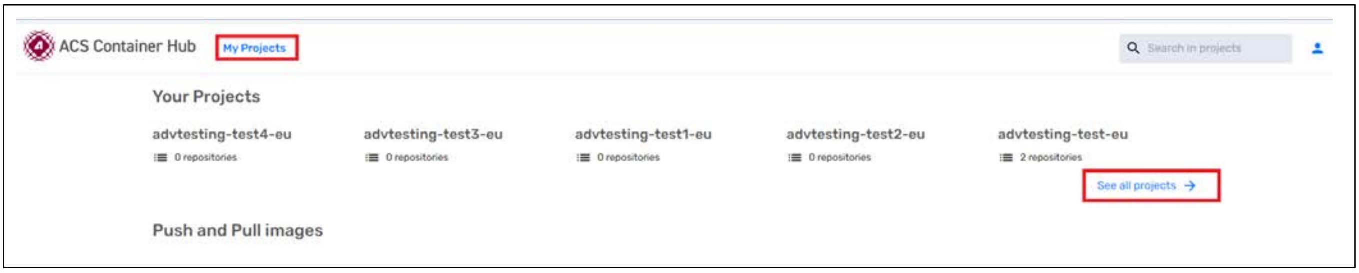


Figure 3-16. Project Search Options

2. Clicking any of the projects links will display a list of projects which you can access.

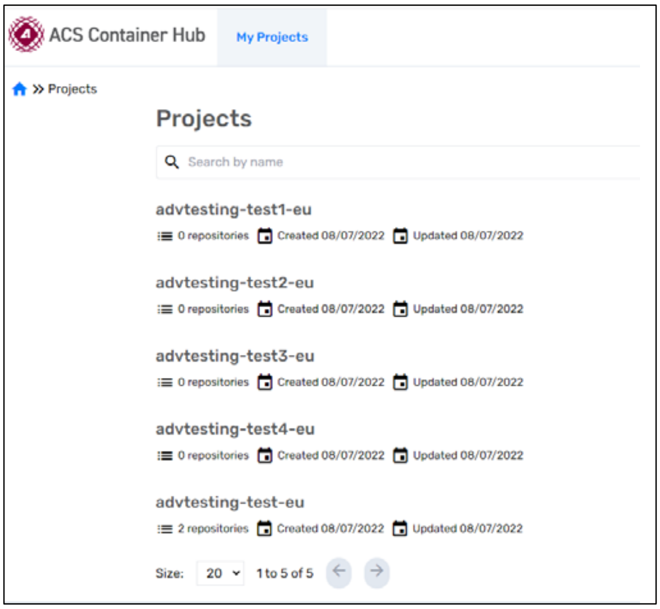

ACS Container Hub
My Projects
» Projects
Projects
Search by name
advtesting-test1-eu
0 repositories Created 08/07/2022 Updated 08/07/2022
advtesting-test2-eu
0 repositories Created 08/07/2022 Updated 08/07/2022
advtesting-test3-eu
0 repositories Created 08/07/2022 Updated 08/07/2022
advtesting-test4-eu
0 repositories Created 08/07/2022 Updated 08/07/2022
advtesting-test-eu
2 repositories Created 08/07/2022 Updated 08/07/2022
Size: 20
1 to 5 of 5

Figure 3-17. Project List

3. In the search field textbox, enter the project search criteria to filter the result(s).

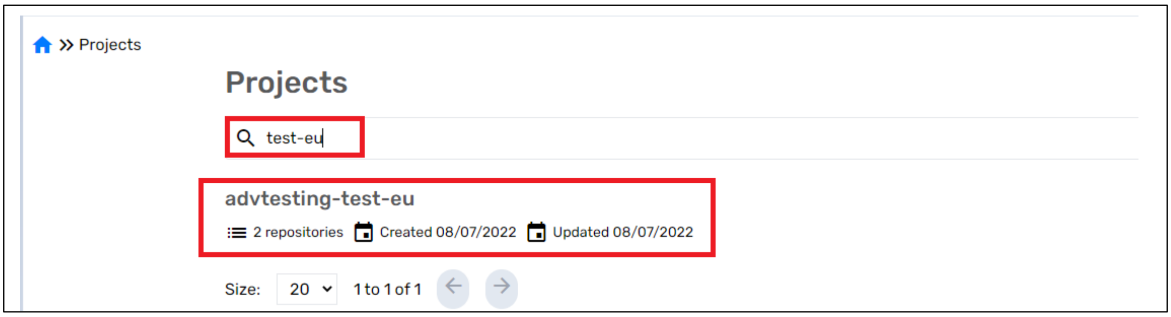
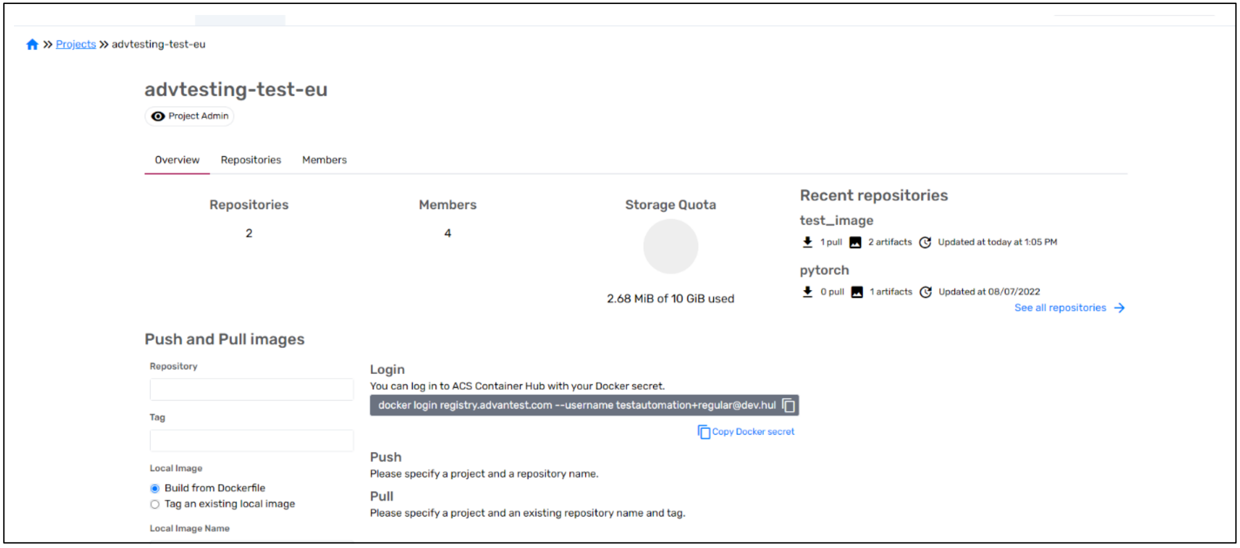


Figure 3-18. Search Criteria and Result

4. Click on any of the results to be redirected to the project details page. From this page, the docker wizard is also available for convenience of copying docker commands.



You can log in to ACS Container Hub with your Docker secret.

```
docker login registry.advantest.com --username testautomation+regular@dev.hul
```

Please specify a project and a repository name.

Please specify a project and an existing repository name and tag.

Figure 3-19. Project Details

3.3.7 Create a New Project

NOTE: Only Organization Administrators (Entitlement Owners) can create projects. Regular users must contact an Organization Administrators to request creating a new project.

1. To create a new project, first log in with Organization Administrator credentials.
2. Select either **My Projects** or **See All Projects**.

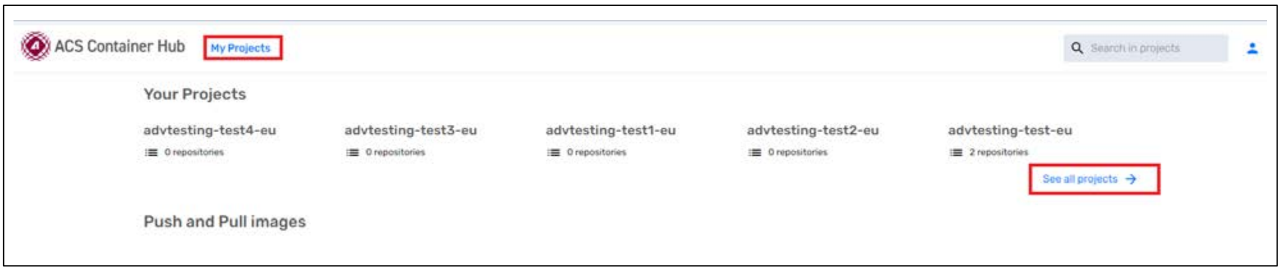


Figure 3-20. Navigate to Projects

3. Click the **New Project** button.

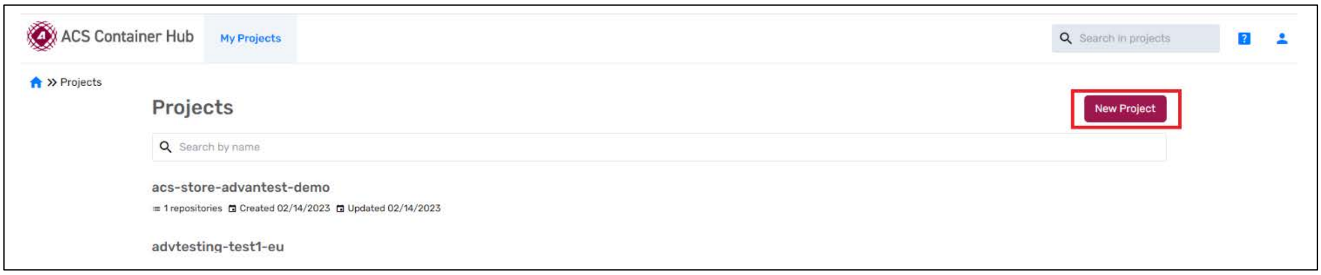
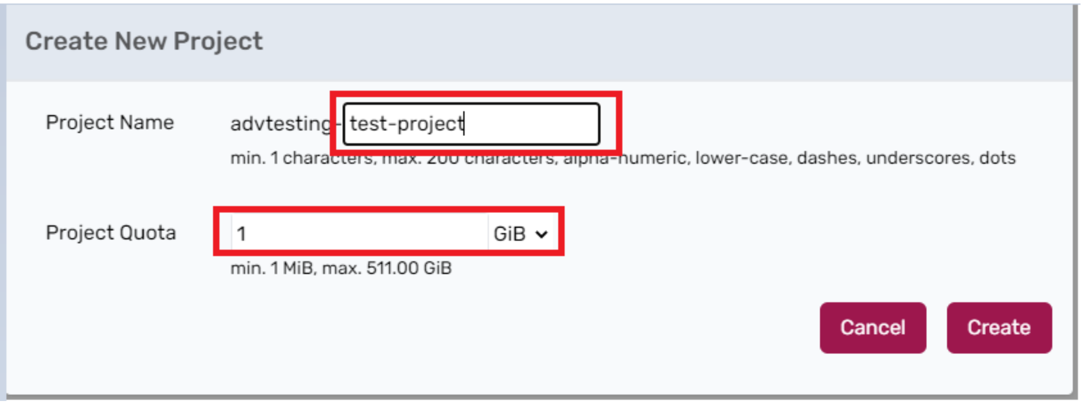


Figure 3-21. New Project Button

4. A *Create New Project* dialog opens. Enter the Project Name and Project Quota. Note that the Organization name appears as a prefix to the Project Name.


Create New Project

Project Name: advtesting-test-project

min. 1 characters, max. 200 characters, alpha-numeric, lower-case, dashes, underscores, dots

Project Quota: 1 GiB

min. 1 MiB, max. 511.00 GiB

Figure 3-22. Project Name and Quota Fields

5. Click the **Create** button to create the new project and be redirected to the details page of the newly created project.

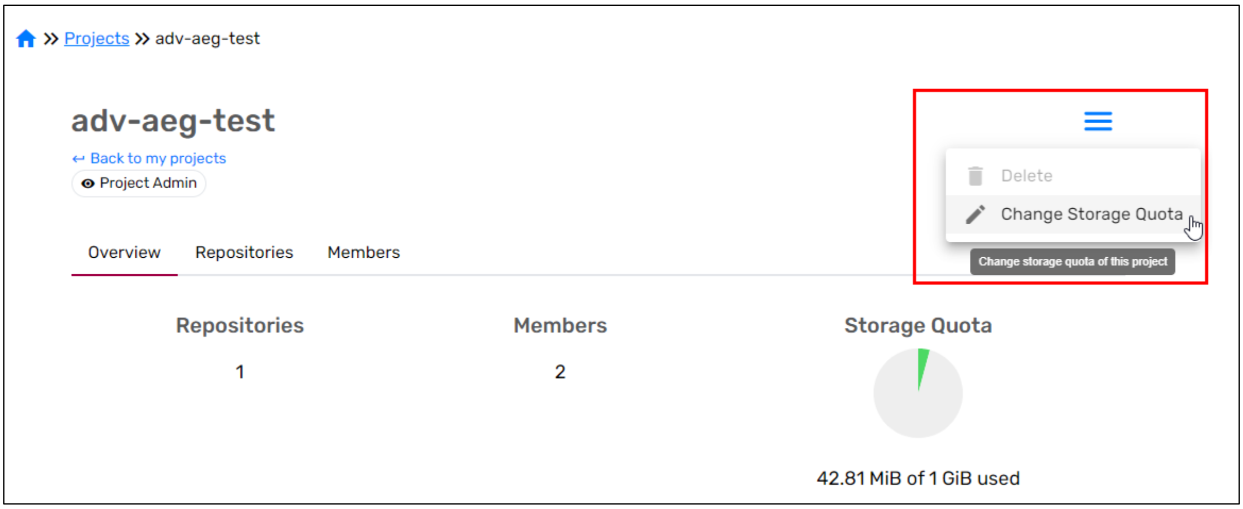
3.3.8 Change Project Storage Quota

NOTE: Only Organization Administrators (Entitlement Owners) can change the storage quota for a project.

Changing the quota helps to better balance the available Organization Storage Quota among projects. The quota value must be positive and always (at least) the amount that the project currently consumes.

1. To change the project storage quota, first navigate into the target project.

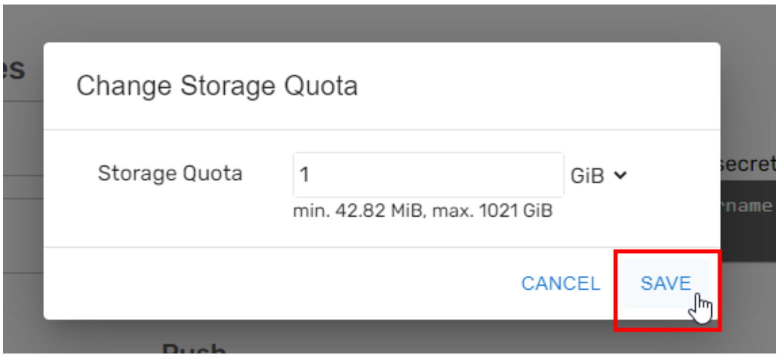
2. Click on the **More** Menu, then select **Change Storage Quota**.



Repositories	Members	Storage Quota
1	2	42.81 MiB of 1 GiB used

Figure 3-23. Change Storage Quota

3. In the *Chang Storage Quota* dialog, adjust the value as desired and click **SAVE**.



Change Storage Quota

Storage Quota: 1 GiB

min. 42.82 MiB, max. 1021 GiB

Figure 3-24. Change Storage Quota Dialog

3.3.9 Delete a Project

NOTE: Only Organization Administrators (Entitlement Owners) and Project Administrators can delete a project. Non-empty projects cannot be deleted. All contained repositories must first be deleted before a project can be deleted.

1. Navigate into the project to be deleted.

2. Click on the **More** Menu, then select **Delete**.

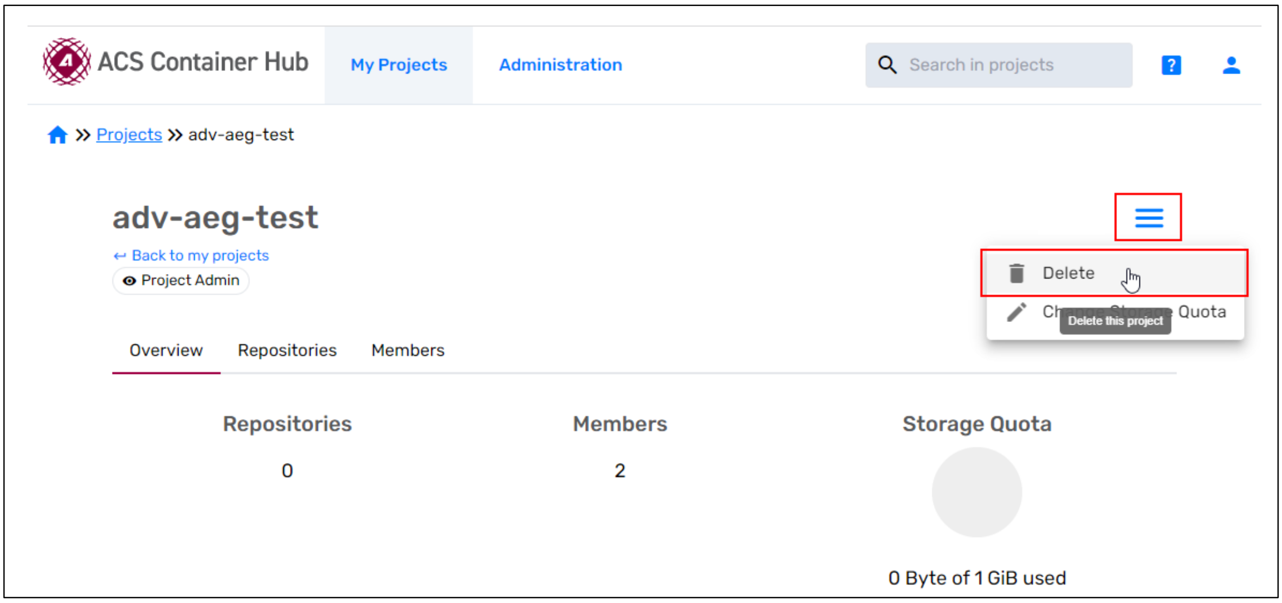
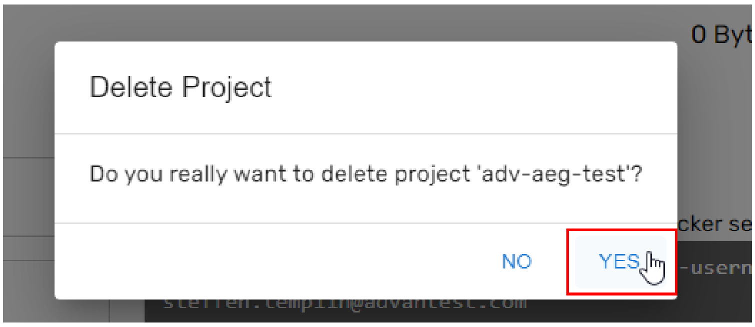


Figure 3-25. Delete Project

3. A *Delete Project* dialog displays. Click **YES** to confirm the project deletion.

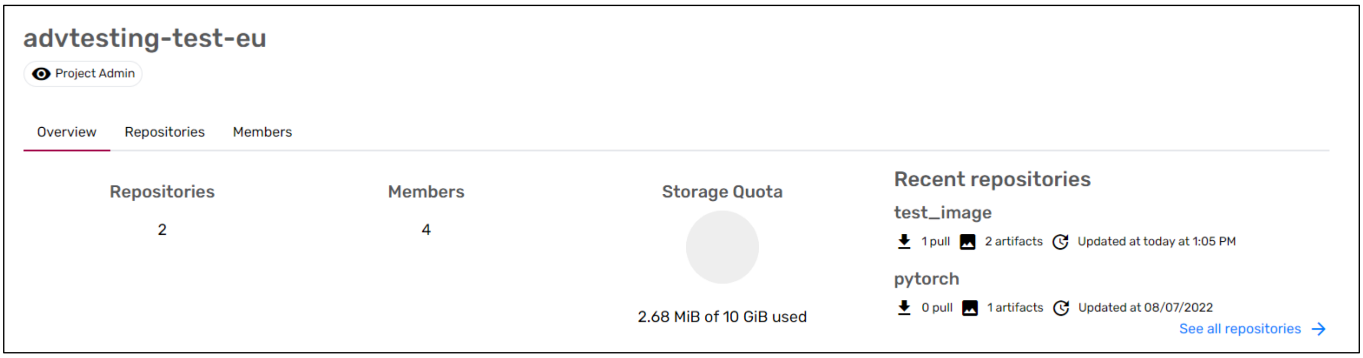


Do you really want to delete project 'adv-aeg-test'?

Figure 3-26. Delete Project Dialog

3.3.10 Project Repository and Artifacts

When you select a project selected and navigate to the project details, the details page displays the number of repositories that the project contains, the number of members, the storage quota, and recent repositories that have been accessed.



advtesting-test-eu
Project Admin

Repositories
2

Members
4

Storage Quota
2.68 MiB of 10 GiB used

Recent repositories

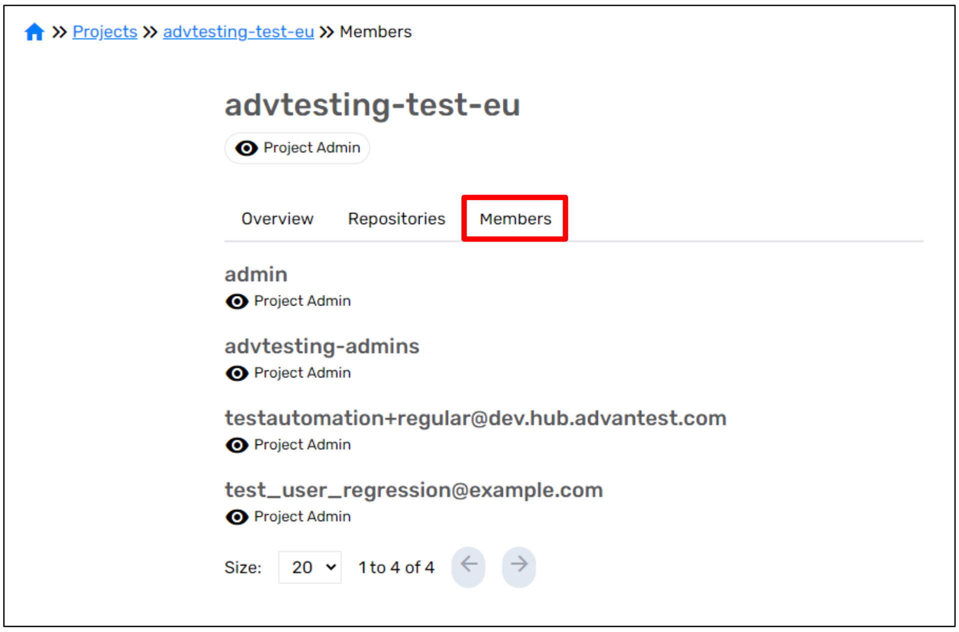
test_image
1 pull 2 artifacts Updated at today at 1:05 PM

pytorch
0 pull 1 artifacts Updated at 08/07/2022

Figure 3-27. Project Details Page

Click on *Members* to see all the users and user groups which have access to Projects.

NOTE: Limited Guest roles will not have access to this tab.



advtesting-test-eu
Project Admin

admin
Project Admin

advtesting-admins
Project Admin

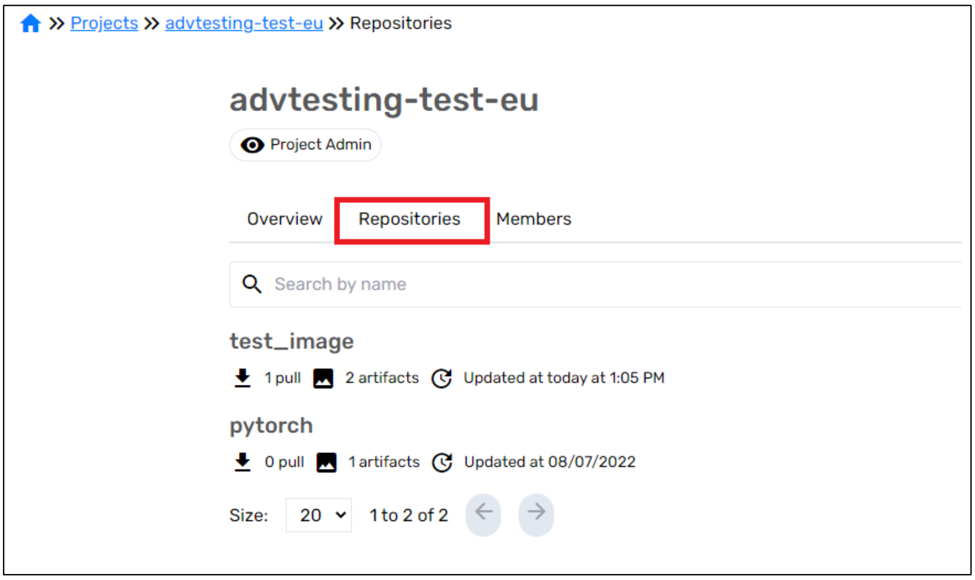
testautomation+regular@dev.hub.advantest.com
Project Admin

test_user_regression@example.com
Project Admin

Size: 20 1 to 4 of 4

Figure 3-28. Members

Click on *Repositories* to display the repository details, such as name of repository, number of pulls, number of artifacts, and when the repository was updated.



advtesting-test-eu

Project Admin

test_image

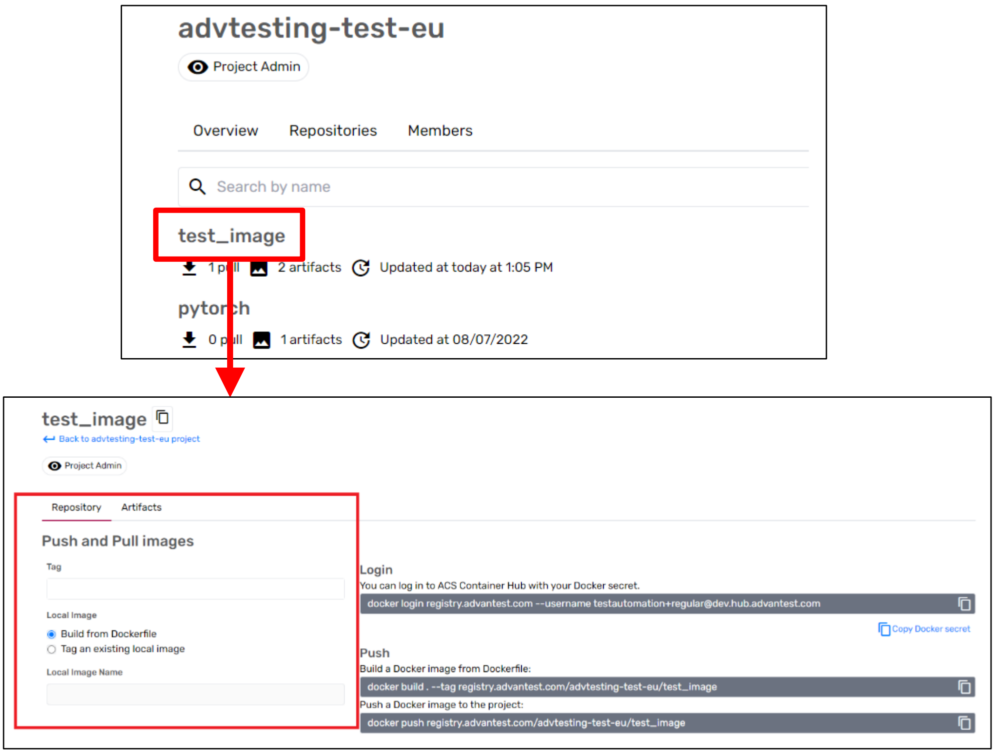
1 pull 2 artifacts Updated at today at 1:05 PM

pytorch

0 pull 1 artifacts Updated at 08/07/2022

Figure 3-29. Repositories

Click on one of the repositories to be redirected to the repository details page. Docker wizard is also available on this page to conveniently copy the docker commands.



advtesting-test-eu

Project Admin

test_image

1 pull 2 artifacts Updated at today at 1:05 PM

pytorch

0 pull 1 artifacts Updated at 08/07/2022

test_image

Project Admin

Push and Pull images

Tag

Local Image

● Build from Dockerfile
○ Tag an existing local image

Local Image Name

Login

You can log in to ACS Container Hub with your Docker secret.

```
docker login registry.advantest.com --username testautomation+regular@dev.hub.advantest.com
```

Push

Build a Docker image from Dockerfile:

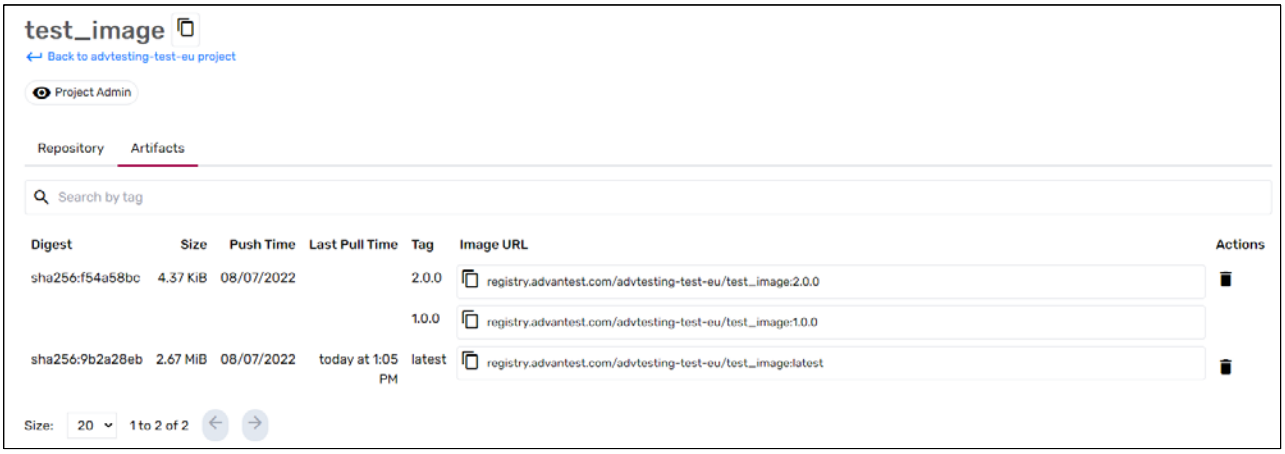
```
docker build . --tag registry.advantest.com/advtesting-test-eu/test_image
```

Push a Docker image to the project:

```
docker push registry.advantest.com/advtesting-test-eu/test_image
```

Figure 3-30. Repository Details

Click on *Artifacts* to show artifacts/images details, such as Digest, size of the image, push time, pull time, tag, and image URL.



test_image

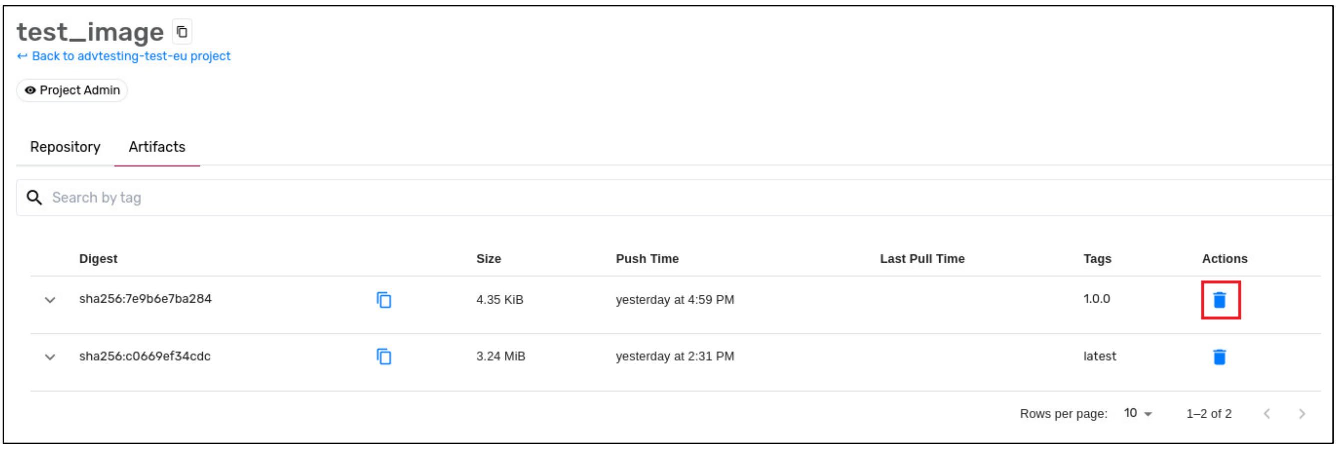
Digest	Size	Push Time	Last Pull Time	Tag	Image URL	Actions
sha256:f54a58bc	4.37 KiB	08/07/2022		2.0.0	registry.advantest.com/advtesting-test-eu/test_image:2.0.0	🗑
				1.0.0	registry.advantest.com/advtesting-test-eu/test_image:1.0.0	
sha256:9b2a28eb	2.67 MiB	08/07/2022	today at 1:05 PM	latest	registry.advantest.com/advtesting-test-eu/test_image:latest	🗑

Figure 3-31. Project Artifacts

3.3.11 Delete Artifact

NOTE: Only those with the role of Maintainer, Project Administrators, or Organization Administrators can delete artifacts.

1. In the *Artifacts* tab of a repository, click on the trash bin icon next to the image that you want to delete. (See section 3.3.10 for information on navigating to artifacts.)



test_image

↩ Back to advtesting-test-eu project

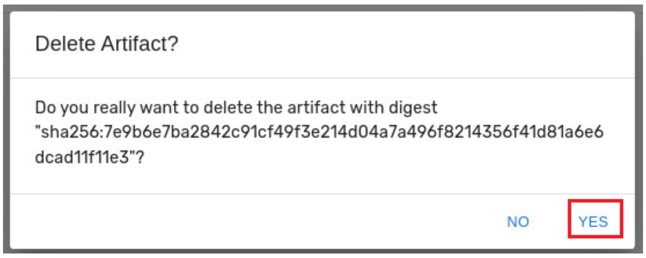
Project Admin

	Digest		Size	Push Time	Last Pull Time	Tags	Actions
⌄	sha256:7e9b6e7ba284		4.35 KiB	yesterday at 4:59 PM		1.0.0	🗑
⌄	sha256:c0669ef34cdc		3.24 MiB	yesterday at 2:31 PM		latest	🗑

Rows per page: 10   1–2 of 2

Figure 3-32. Trash Bin Icon

2. A dialog displays to confirm the delete action. Select **YES** to confirm.



Delete Artifact?

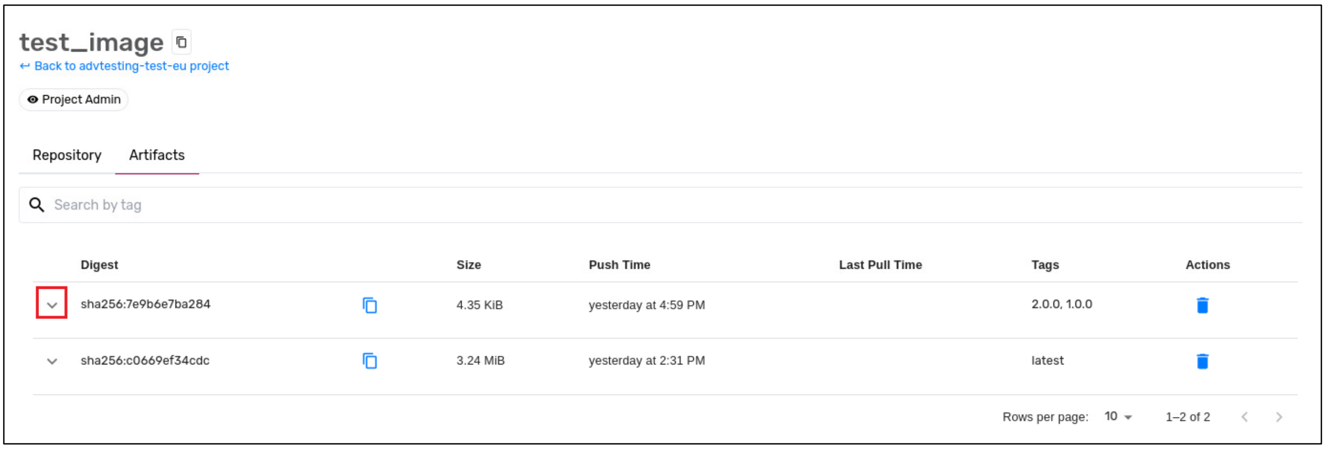
Do you really want to delete the artifact with digest "sha256:7e9b6e7ba2842c91cf49f3e214d04a7a496f8214356f41d81a6e6dcad11f11e3"?

Figure 3-33. Confirm Artifact Delete

3.3.12 Delete Image Tag

NOTE: Only those with the role of Maintainer, Project Administrator, or Organization Administrator can delete an image tag.

1. In the *Artifacts* tab of a repository, click on the arrow icon to the left of the image digest to expand the row. (See section 3.3.10 for information on navigating to artifacts.)



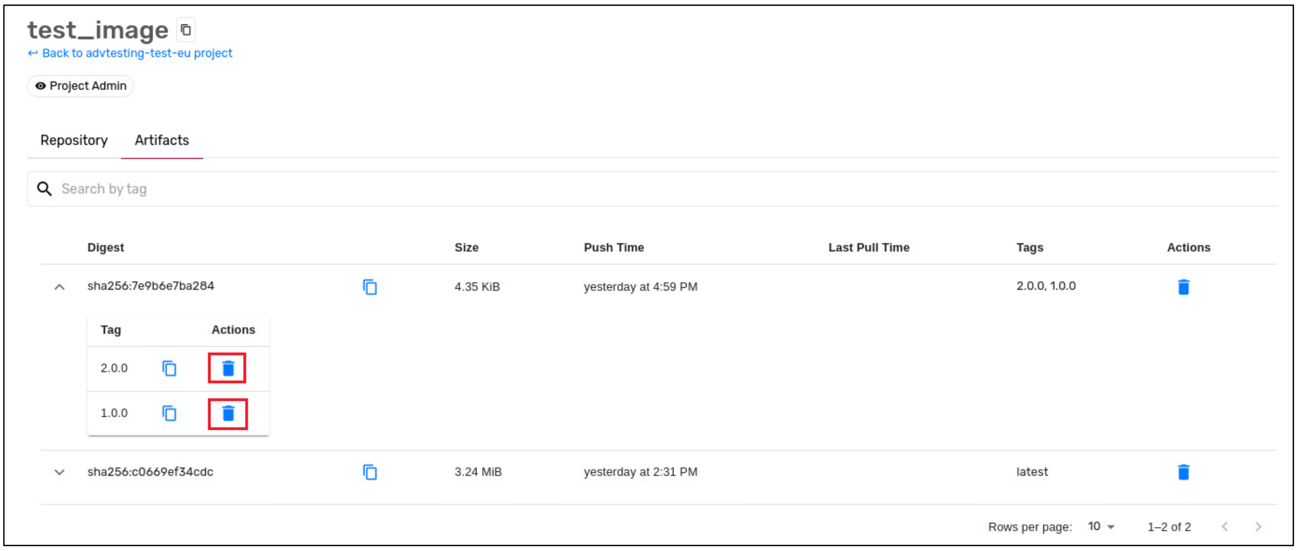
test_image
↩ Back to advtesting-test-eu project
Project Admin

	Digest		Size	Push Time	Last Pull Time	Tags	Actions
⌄	sha256:7e9b6e7ba284		4.35 KiB	yesterday at 4:59 PM		2.0.0, 1.0.0	
⌄	sha256:c0669ef34cdc		3.24 MiB	yesterday at 2:31 PM		latest	

Rows per page: 10   1–2 of 2

Figure 3-34. Expand Image Row

2. Click on the delete button for the tag(s) you wish to delete.



test_image
↩ Back to advtesting-test-eu project
Project Admin

	Digest		Size	Push Time	Last Pull Time	Tags	Actions
⌃	sha256:7e9b6e7ba284		4.35 KiB	yesterday at 4:59 PM		2.0.0, 1.0.0	
⌄	sha256:c0669ef34cdc		3.24 MiB	yesterday at 2:31 PM		latest	

Tag	Actions
2.0.0	
1.0.0	

Rows per page: 10   1–2 of 2

Figure 3-35. Delete Tags

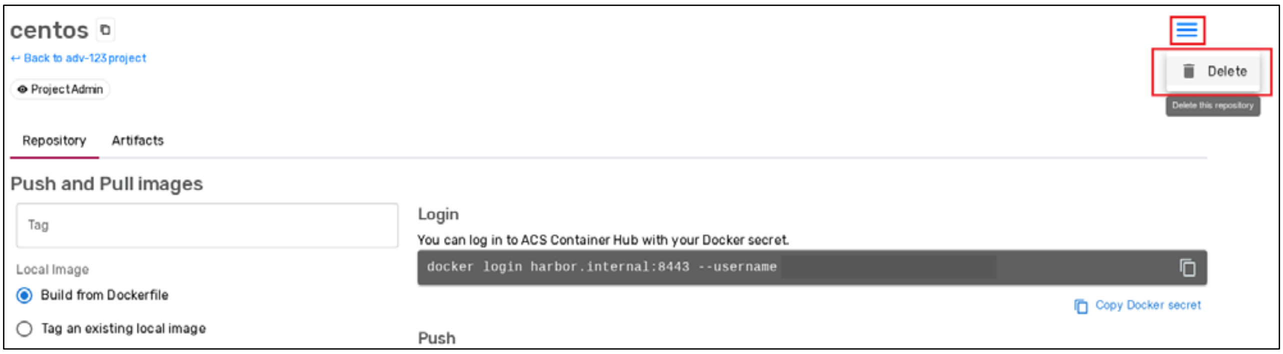
3. A dialog displays to confirm the delete action. Select **YES** to confirm.

3.3.13 Delete Repository

NOTE: Only those with the role of Maintainer or Project Administrator can delete repositories. Deleting a repository will also delete all contained artifacts.

1. Navigate to the repository you want to delete. (See section 3.3.10 for information on navigating to repositories.)

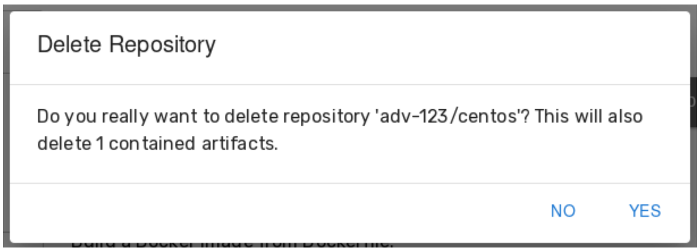
2. In the repository, click the More menu (top-right) and select **Delete**.



```
docker login harbor.internal:8443 --username
```

Figure 3-36. Delete Repository

3. A confirmation dialog displays. Click on **YES** to confirm the delete action.



Do you really want to delete repository 'adv-123/centos'? This will also delete 1 contained artifacts.

Figure 3-37. Confirm Repository Delete

3.3.14 Create Client Credentials

NOTE: Only Organization Administrators have permission to create Client Credentials.

For machine-to-machine communication between the ACS Container Hub and a Docker client, Client Credentials are required. Follow the steps below to create and use Client Credentials:

1. Log in to the ACS Container Hub with Organization Administrator credentials, then click **Credentials**.

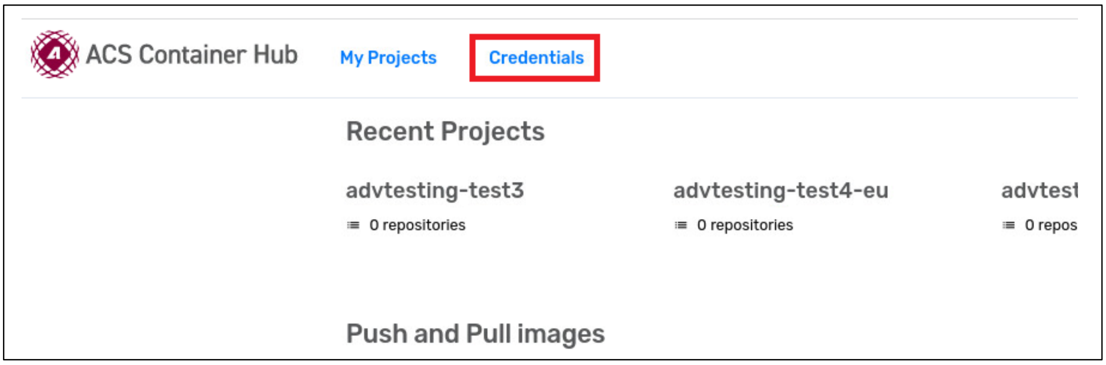
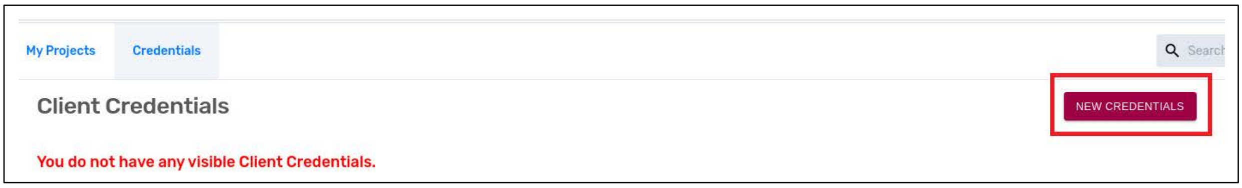


Figure 3-38. Click Credentials

2. The *Client Credentials* screen displays. Click the **New Credentials** button.

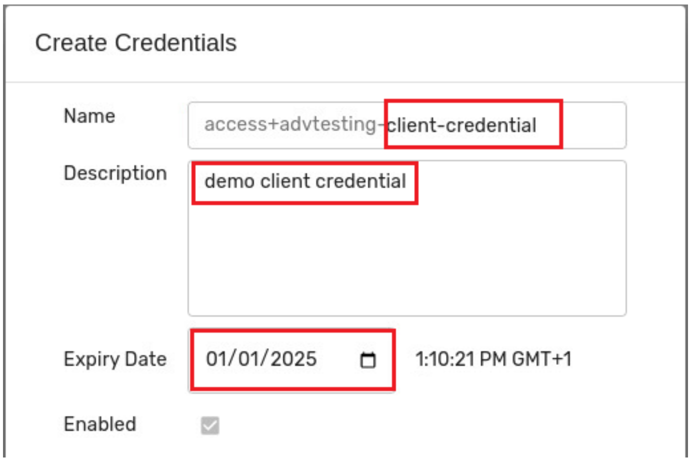


You do not have any visible Client Credentials.

Figure 3-39. New Credentials Button

3. A *Create Credentials* dialog displays. Enter a meaningful name and description for the client credential. Also select an Expiry date, if desired (by default, credentials do not expire).

   NOTE: An "access+[organization name]" prefix will automatically precede this name.



Create Credentials

Name: access+advtesting-client-credential

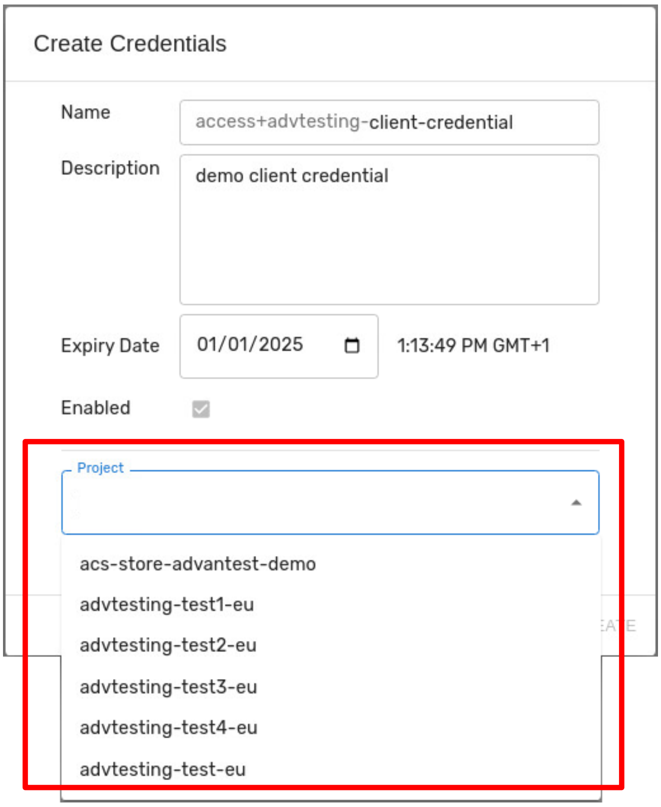
Description: demo client credential

Expiry Date: 01/01/2025 1:10:21 PM GMT+1

Enabled ☑

Figure 3-40. Project Name and Description

4. Click on *Project* textbox to display a drop-down list containing all the projects for the organization.



Create Credentials

Name: access+advtesting-client-credential

Description: demo client credential

Expiry Date: 01/01/2025 1:13:49 PM GMT+1

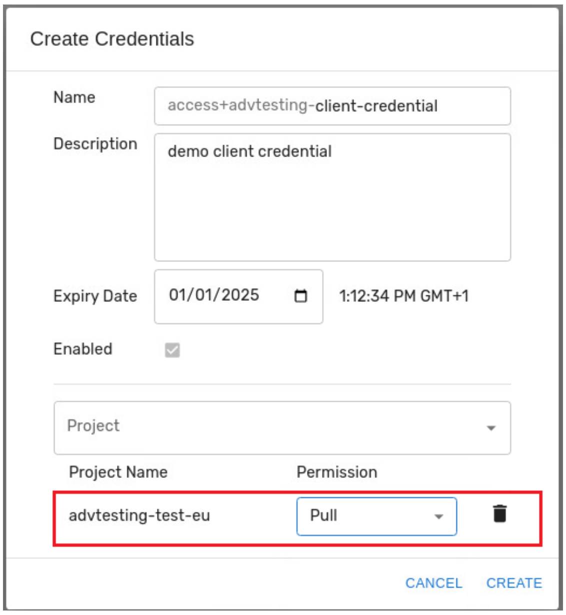
Enabled ☑

Project

acs-store-advantest-demo
advtesting-test1-eu
advtesting-test2-eu
advtesting-test3-eu
advtesting-test4-eu
advtesting-test-eu

Figure 3-41. Project Drop-Down List

5. Select the desired project from the drop-down list and select a permission level.



Create Credentials

Name: access+advtesting-client-credential

Description: demo client credential

Expiry Date: 01/01/2025 1:12:34 PM GMT+1

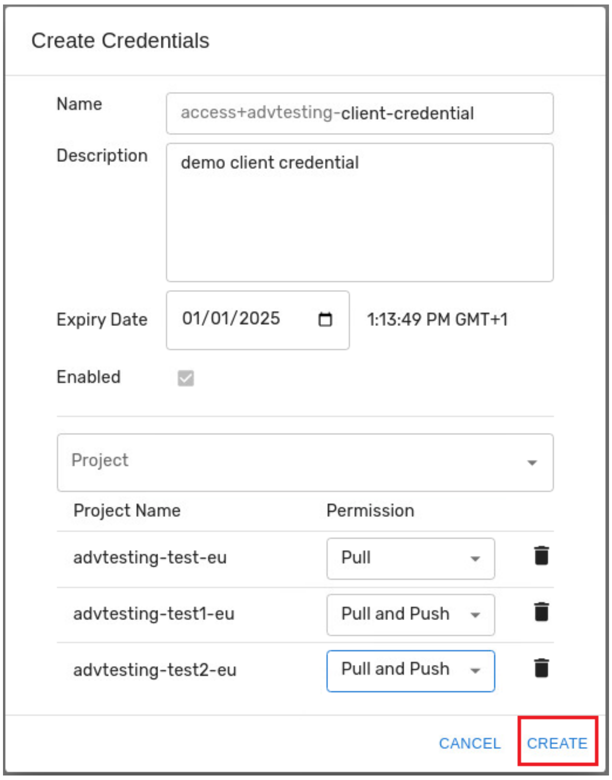
Enabled: ☑

Project

Project Name	Permission
advtesting-test-eu	Pull

Figure 3-42. Project and Permission Selection

6. If desired, more than one project can be selected, as shown below. When finished selecting the project(s), click **CREATE**.



Create Credentials

Name: access+advtesting-client-credential

Description: demo client credential

Expiry Date: 01/01/2025 1:13:49 PM GMT+1

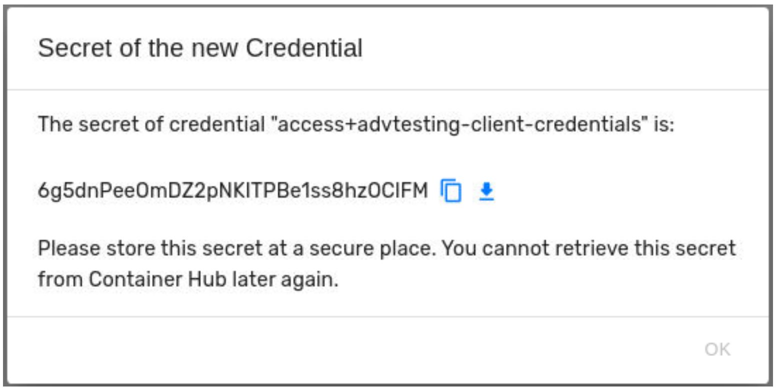
Enabled: ☑

Project

Project Name	Permission
advtesting-test-eu	Pull
advtesting-test1-eu	Pull and Push
advtesting-test2-eu	Pull and Push

Figure 3-43. Multiple Projects Selected

7. Once the client credentials are created, the secret (password) of the new credential displays. Make sure to **copy the secret or download the JSON file and keep it safe**. This is the only opportunity to view the secret details. Click OK when ready to close this dialog.



Secret of the new Credential

The secret of credential "access+advtesting-client-credentials" is:

6g5dnPee0mDZ2pNKITPBe1ss8hz0ClFM

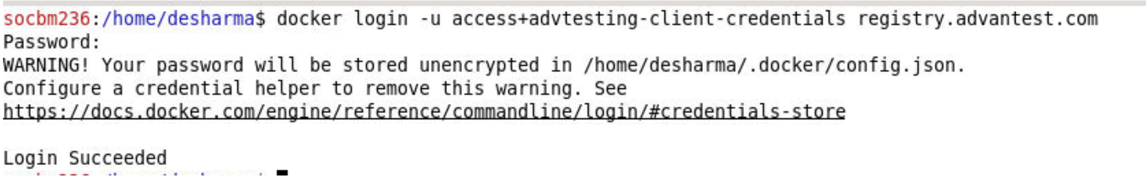
Please store this secret at a secure place. You cannot retrieve this secret from Container Hub later again.

Figure 3-44. Client Credential Secret

The newly-created client credentials are now displayed on the homepage. The JSON file naming convention is "CredentialName.json" (e.g. client-credentials.json). The content of the file will be in the following format, where "name" is the username for the docker login, and "secret" is the password:

```
{"name":"access+advtesting-client-credentials","secret":"a6qcHu5QTrELEzh0L2VWClHVgy7ObIkd"}
```

8. Open a command terminal on the Host workstation and log in using the username and password.



```
socbm236:/home/desharma$ docker login -u access+advtesting-client-credentials registry.advantest.com
Password:
WARNING! Your password will be stored unencrypted in /home/desharma/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

3.3.15 Update Client Credentials

NOTE: Only Organization Administrators have permission to update Client Credentials.

1. Log in to the ACS Container Hub with Organization Administrator credentials, then click **Credentials**.

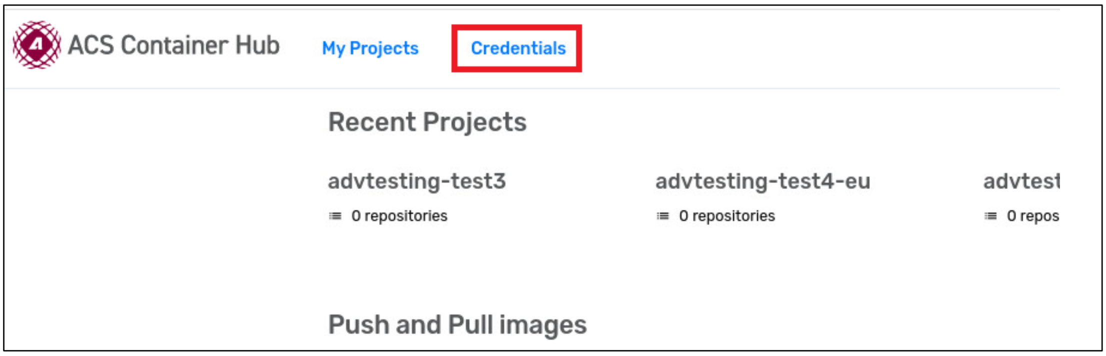
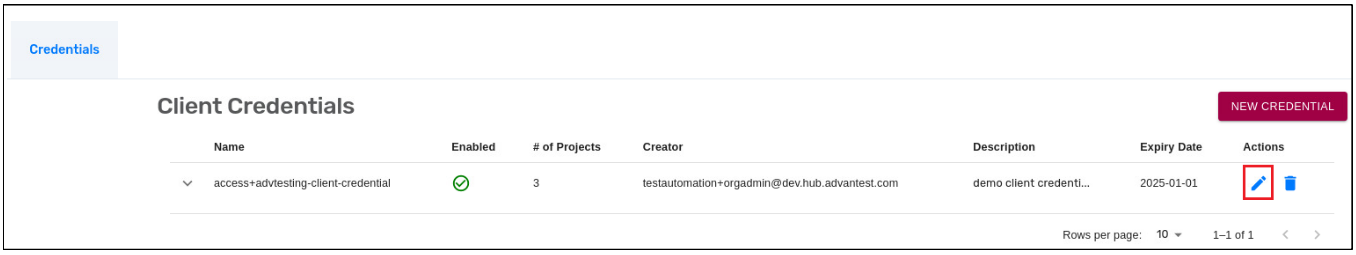


Figure 3-45. Click Credentials

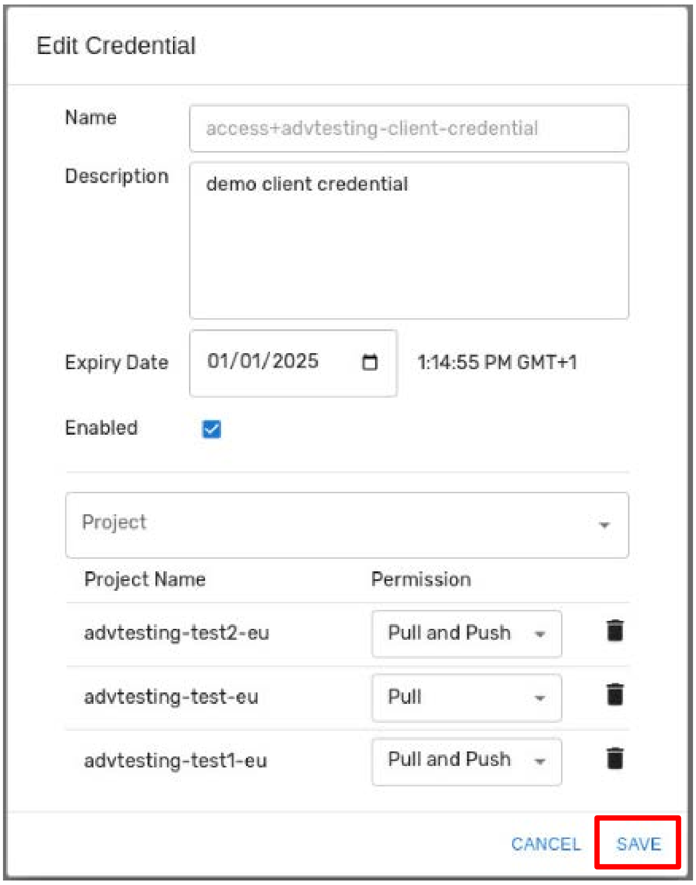
2. The *Client Credentials* screen displays. Click the Edit icon for the credential you want to update.



	Name	Enabled	# of Projects	Creator	Description	Expiry Date	Actions
⌄	access+advtesting-client-credential	✓	3	testautomation+orgadmin@dev.hub.advantest.com	demo client credenti...	2025-01-01	✎ 🗑

Figure 3-46. Update Credentials Icon

3. The client credential opens in edit mode. With the exception of the Name field, every field can be edited in the Edit Credential dialog. After editing is completed, click SAVE to confirm the changes and exit the dialog.



Edit Credential

Name: access+advtesting-client-credential

Description: demo client credential

Expiry Date: 01/01/2025 1:14:55 PM GMT+1

Enabled ☑

Project

Project Name	Permission
advtesting-test2-eu	Pull and Push
advtesting-test-eu	Pull
advtesting-test1-eu	Pull and Push

Figure 3-47. Edit Credential Dialog

3.3.16 Delete Client Credential

NOTE: Only Organization Administrators have permission to delete Client Credentials.

1. Log in with Organization Administrator credentials and click on **Credentials**.

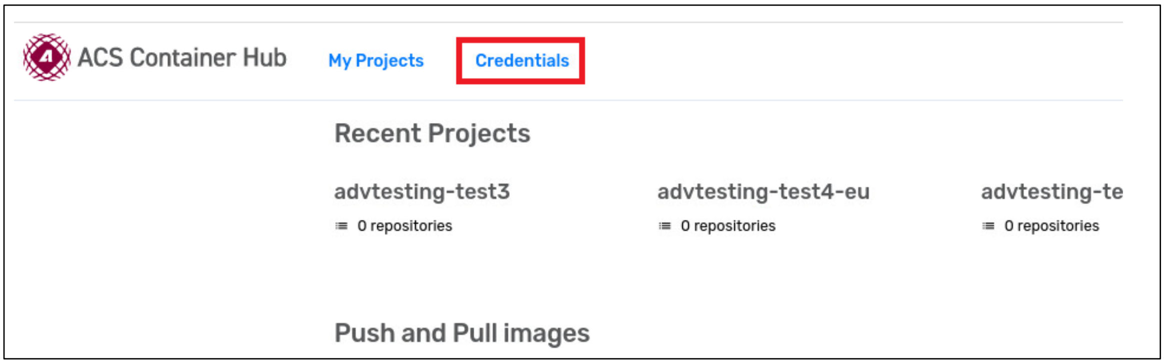
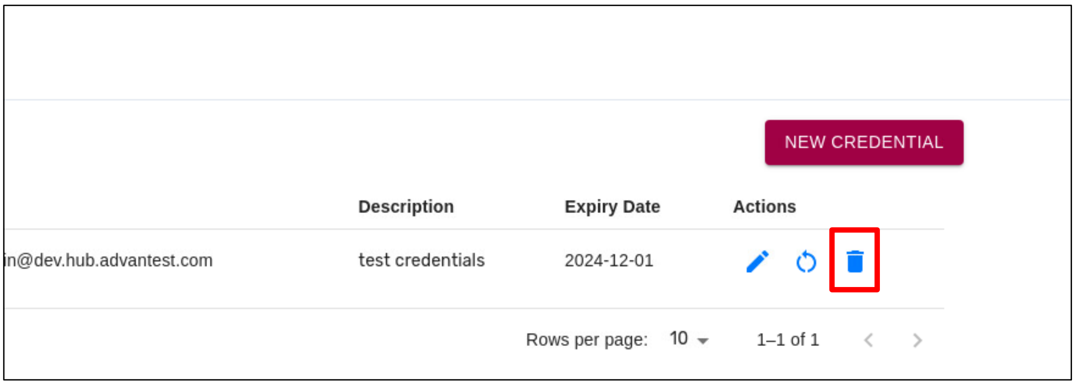


Figure 3-48. Client Credentials

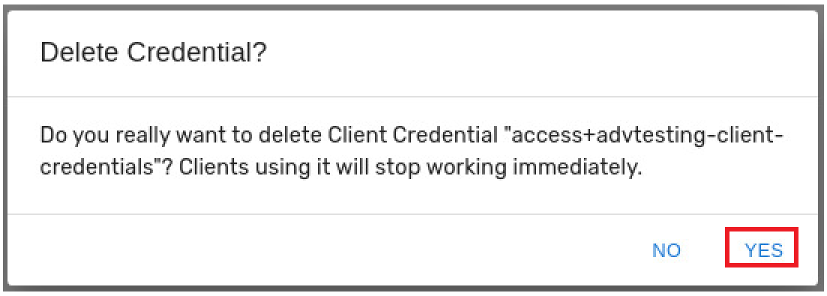
2. The *Credentials* screen displays. Click the Delete icon for the Client Credential you want to delete.



	Description	Expiry Date	Actions
in@dev.hub.advantest.com	test credentials	2024-12-01	

Figure 3-49. Delete Icon

3. A confirmation dialog displays. Click on **YES** to confirm the delete action.


Delete Credential?

Do you really want to delete Client Credential "access+advtesting-client-credentials"? Clients using it will stop working immediately.

Figure 3-50. Confirm Credential Delete

3.3.17 Reset Client Credential

NOTE: Only Organization Administrators have permission to reset Client Credentials. The Reset operation will change the Client credential. Therefore, clients using this credential will stop working.

1. Login with Organization Administrator credentials and click on **Credentials**.

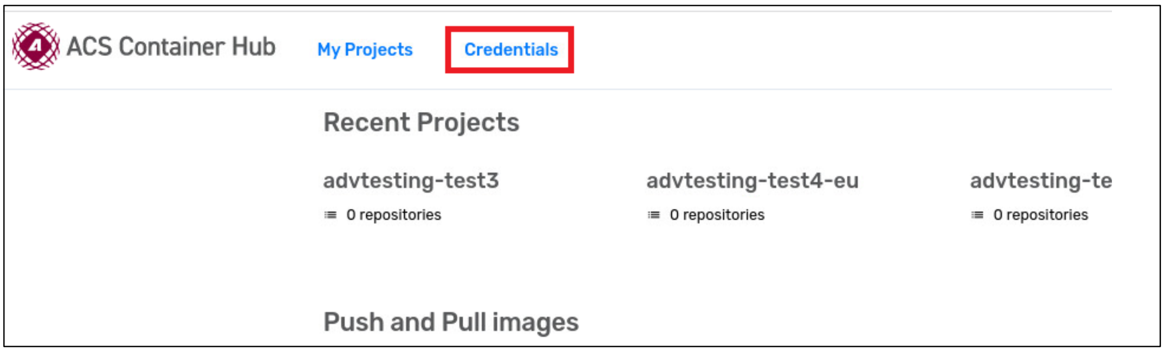
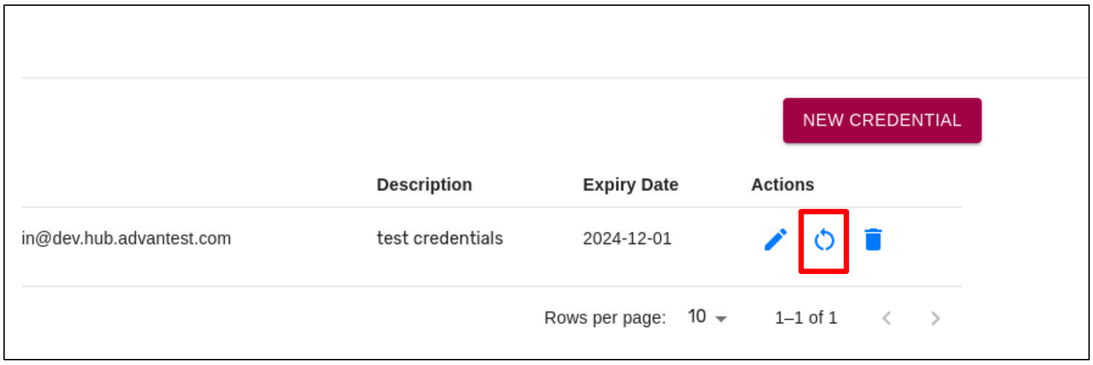


Figure 3-51. Client Credentials

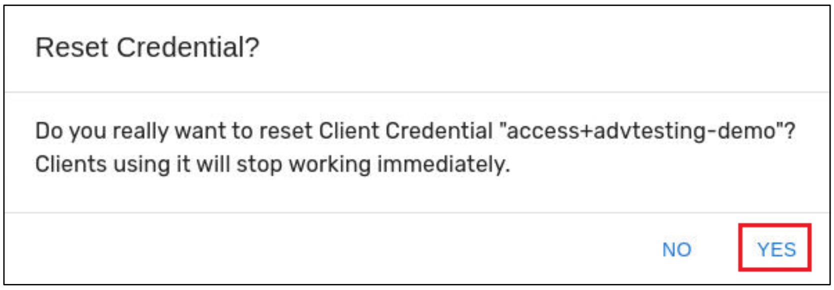
2. The Credentials screen displays. Click the Reset icon for the Client Credential you want to reset.



	Description	Expiry Date	Actions
in@dev.hub.advantest.com	test credentials	2024-12-01	✎ ⟲ 🗑

Figure 3-52. Reset Icon

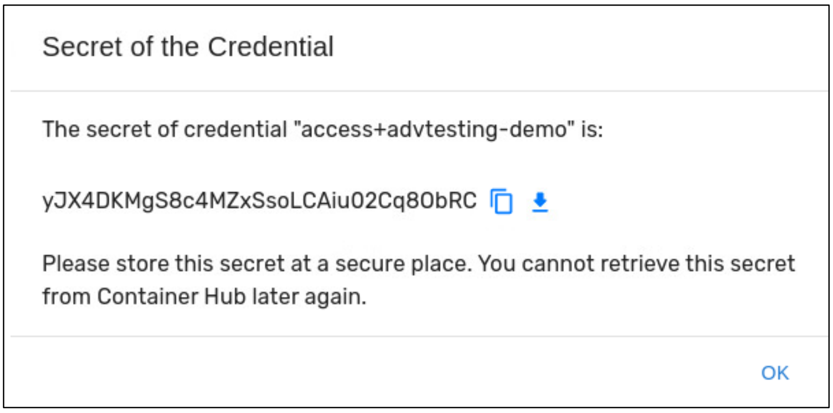
3. A confirmation dialog displays. Click on **YES** to confirm the reset action.


Reset Credential?

Do you really want to reset Client Credential "access+advtesting-demo"? Clients using it will stop working immediately.

Figure 3-53. Confirm Credential Reset

4. Upon successful reset, the credential password (secret) displays. Copy the secret or download the json file and keep it safe. This is the only time the secret details will be displayed. Click **OK** once the secret is safely stored.



Secret of the Credential

The secret of credential "access+advtesting-demo" is:

yJX4DKMgS8c4MZxSsoLCAiu02Cq8ObRC

Please store this secret at a secure place. You cannot retrieve this secret from Container Hub later again.

Figure 3-54. Credential Secret

3.3.18 Create 1:1 Application Descriptor

NOTE: Only Organization Administrators have permission to create Application Descriptors.

1. Log in with Organization Administrator credentials and click on **Applications** > **1:1 Application Descriptors**.

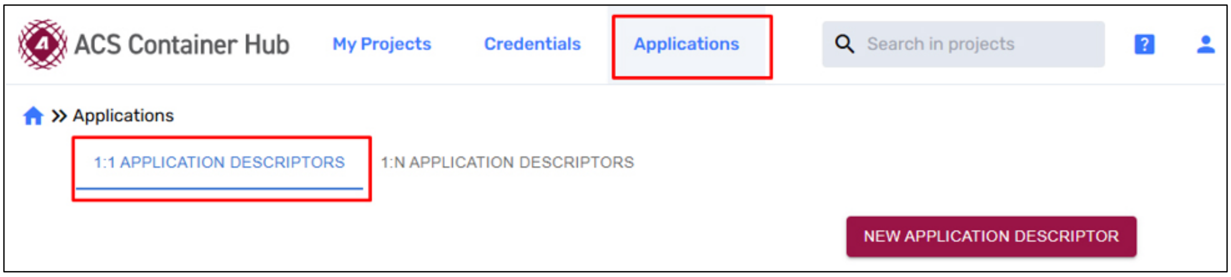


Figure 3-55. 1:1 Application Descriptors

2. Click the **NEW APPLICATION DESCRIPTOR** button to open the *Create Application Descriptor* dialog.

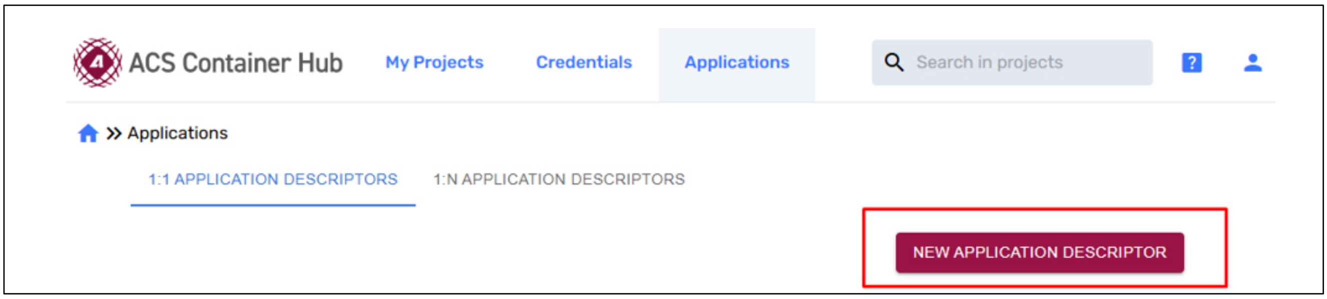


Figure 3-56. NEW APPLICATION DESCRIPTOR Button

3. In the Application Descriptor dialog, enter the requested information (see Figure 3-57 for example image):

CAUTION: Make sure that your released Test Program and Application Descriptor make a **perfect match**. Otherwise, it is technically possible to create application descriptors with different selectors that lead to ambiguous matches during application descriptor selection in production. Take the following example for 2 application descriptors:

Descriptor #1 Selector	Descriptor #2 Selector
Device name: my_device	Device name: my_device
Product family: my_product	Product family: ANY
Test program name: TestProgram_1	Test program name: TestProgram_1
Test program revision: ANY	Test program revision: v1

In the above scenario, the test program in production would emit the following attributes:

- Device name: my_device
- Product family: my_product
- Test program name: TestProgram_1
- Test program revision: v1

In this example scenario, both descriptors match, because each selector matches the device name and 2 other attributes. This is an error condition in production, and a proper descriptor selection cannot be made. To avoid this type of condition, specify as many selector attributes as possible in the application descriptor. If a descriptor shall be useable for multiple products and/or test programs, make sure to not create an ambiguous situation as presented in the above example.

The Application Descriptor dialog consists of three sections (GENERAL, VOLUMES, and CONTAINERS) where descriptor details should be entered. The fields of each section are described below.

GENERAL Tab

- In the **Info** section, enter the name of the Application Descriptor. This is a required attribute.
- In the **Selector** section, *Device name* is a required attribute. The other attributes in this section (Test program name, Product family, and Test program revision) are optional.
    - If the optional attributes are left empty, they will match any input value originating from the test program during test operation. This is indicated as "ANY" in the dialog text fields. If a value is given, the test program must generate the exact value during test operation to match this descriptor.
    - Every selector attribute combination can only exist once among all application descriptors.

VOLUMES Tab

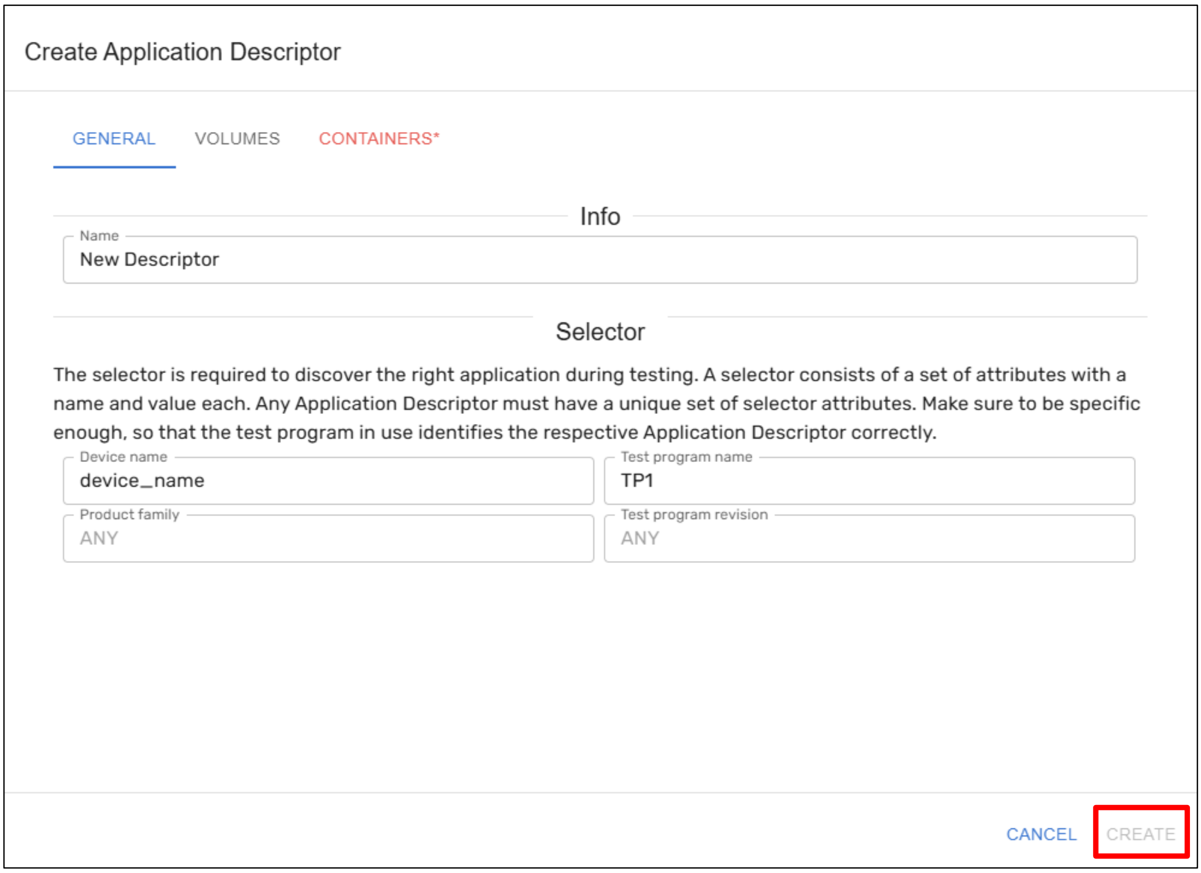
- In the **Info** section, enter the name of the Volume. This is a required attribute.
- In the **Image** section, enter the required Image details.

CONTAINERS Tab

- In the **Info** section, enter the name of the Container. This is a required attribute.
- In the **Image** section, enter the required Image details.
- For **Volumes**, add the associated volumes.
- For **Info ACS Edge Requirements**, use the checkbox to enable the GPU, or leave it blank to disable the GPU.
- **Exposed ports** is an optional attribute. If entered, the same port number would be mapped from the host to the container.
- **Mapped ports** is an optional attribute. Enter a port number to explicitly specify host-to-container port mapping, along with the protocol. Enter this information in the format `host_port:container_port/protocol`. For example, `8200:8200/tcp.`
- Use the **Monitoring** checkbox to enable or disable monitoring for a container. If enabled, enter the port and path for scraping metrics. You can also indicate the scrape interval (time interval between consecutive scrapes) and the scrape timeout for a particular scrape.

  NOTE: If monitoring metrics are implemented into your Application and deviate from the standard monitoring endpoint, you can optionally override the port and path for metric scraping. ACS Nexus will consider these overridden settings in production.
- **Environment** variables can optionally be specified if you need to configure the application container during runtime. These are passed into the container as defined in this section.

4. When finished entering the Application Descriptor information, click **CREATE** in the bottom-right corner of the dialog.



Create Application Descriptor

Info

Name: New Descriptor

Selector

The selector is required to discover the right application during testing. A selector consists of a set of attributes with a name and value each. Any Application Descriptor must have a unique set of selector attributes. Make sure to be specific enough, so that the test program in use identifies the respective Application Descriptor correctly.

Device name: device_name

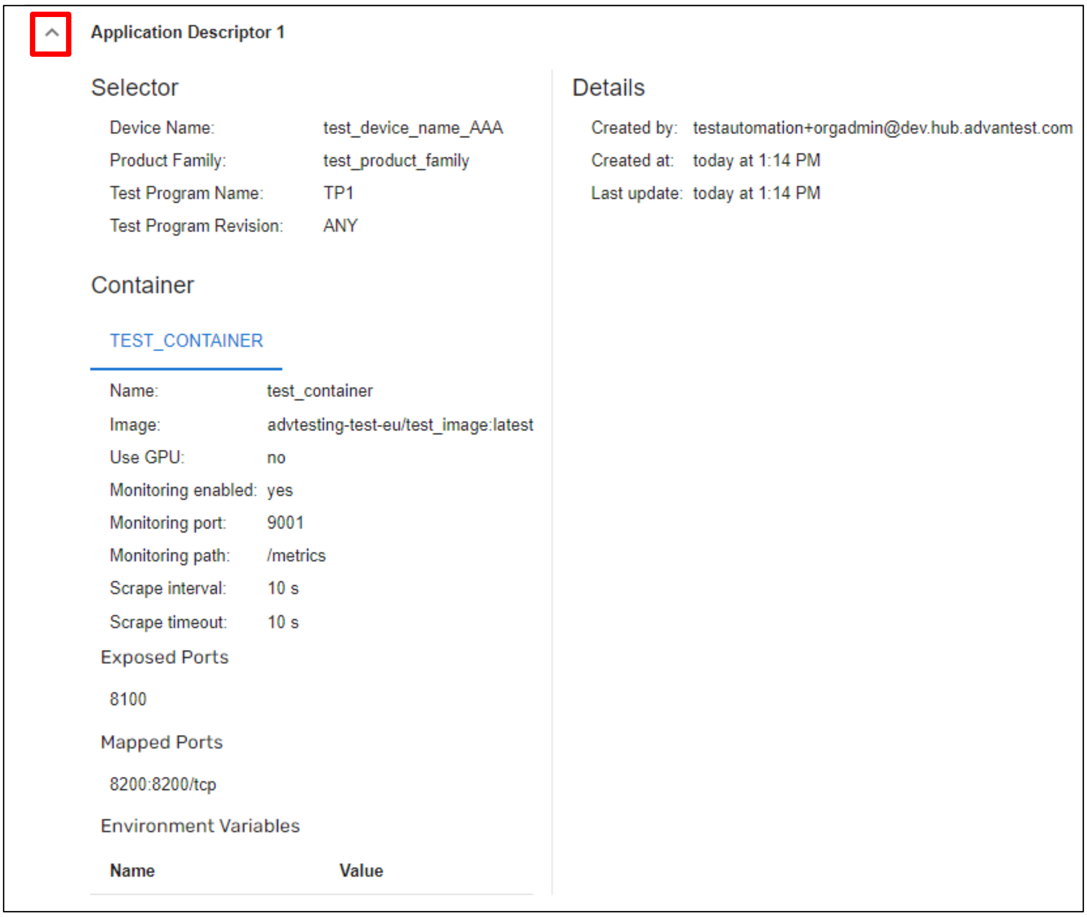
Test program name: TP1

Product family: ANY

Test program revision: ANY

Figure 3-57. Create Application Descriptor Dialog

5. To see the Application Descriptor details, click on the arrow button to the left of the Application Descriptor name.



Application Descriptor 1

Selector

Device Name:	test_device_name_AAA
Product Family:	test_product_family
Test Program Name:	TP1
Test Program Revision:	ANY

Container

TEST_CONTAINER

Name:	test_container
Image:	advtesting-test-eu/test_image:latest
Use GPU:	no
Monitoring enabled:	yes
Monitoring port:	9001
Monitoring path:	/metrics
Scrape interval:	10 s
Scrape timeout:	10 s

Exposed Ports

8100

Mapped Ports

8200:8200/tcp

Environment Variables

Name	Value

Details

Created by:	testautomation+orgadmin@dev.hub.advantest.com
Created at:	today at 1:14 PM
Last update:	today at 1:14 PM

Figure 3-58. Application Descriptor Details

3.3.19 Update 1:1 Application Descriptor

NOTE: Only Organization Administrators have permission to update Application Descriptors.

1. Log in with Organization Administrator credentials and click on **Applications** > **1:1 Application Descriptors**.

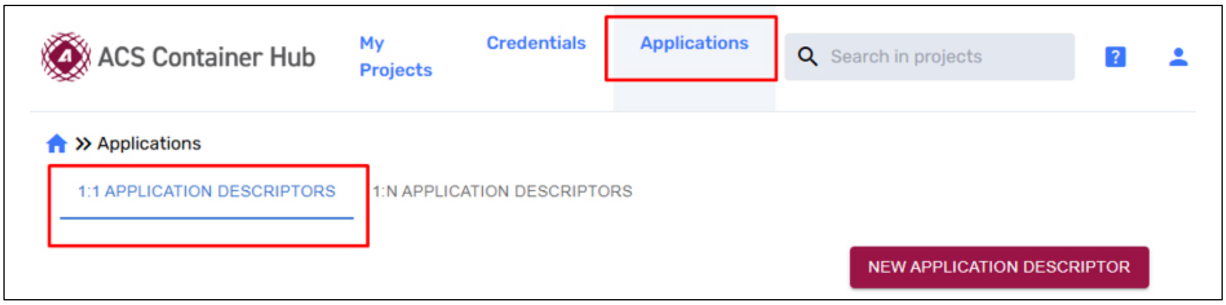
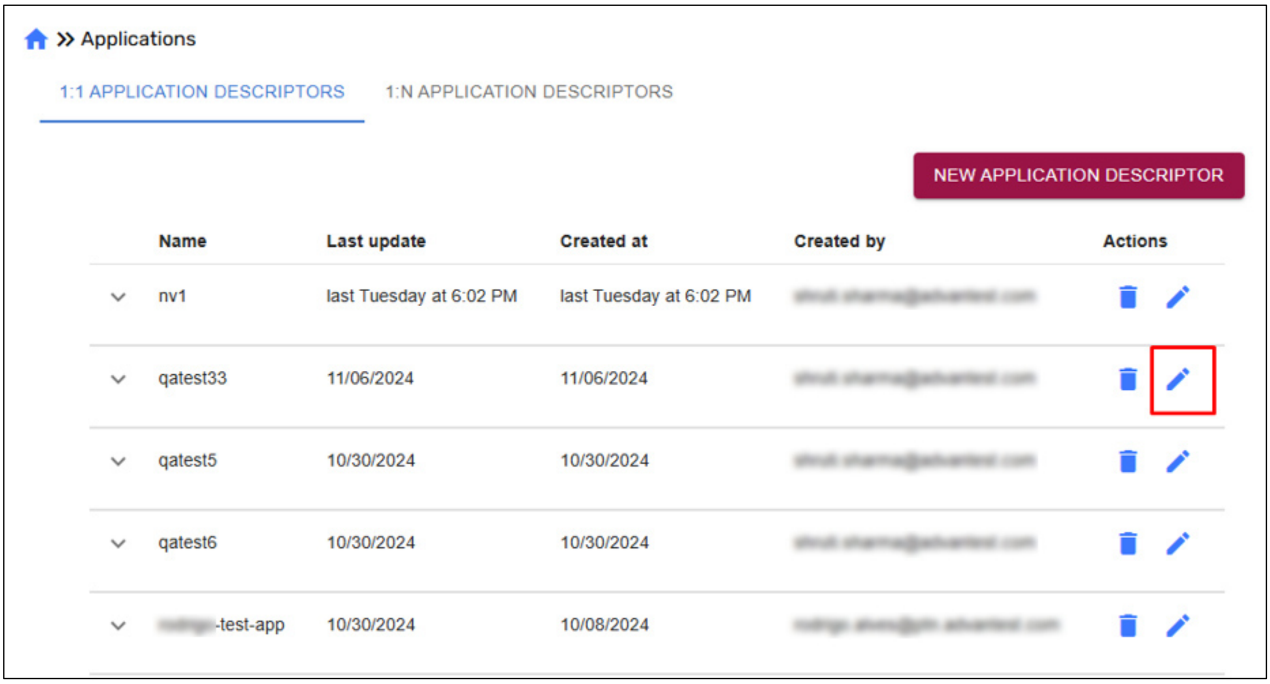


Figure 3-59. 1:1 Application Descriptors

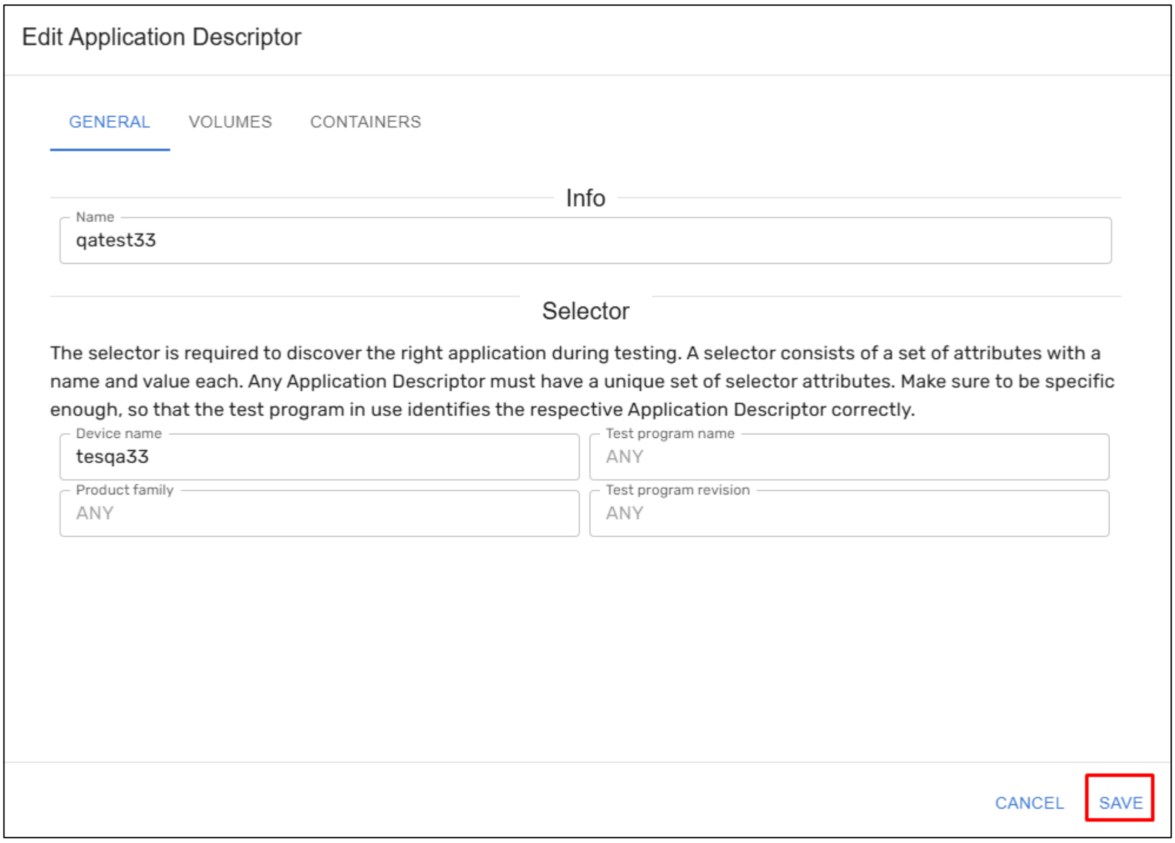
2. Click the Edit icon that is associated with the desired Application Descriptor.


Applications
1:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

Name	Last update	Created at	Created by	Actions
nv1	last Tuesday at 6:02 PM	last Tuesday at 6:02 PM	[redacted]	
qatest33	11/06/2024	11/06/2024	[redacted]	
qatest5	10/30/2024	10/30/2024	[redacted]	
qatest6	10/30/2024	10/30/2024	[redacted]	
[redacted]-test-app	10/30/2024	10/08/2024	[redacted]	

Figure 3-60. Application Descriptor Edit Button

3. Update the desired details and click **SAVE** to save the edits when completed.



Edit Application Descriptor

Info

Name: qatest33

Selector

The selector is required to discover the right application during testing. A selector consists of a set of attributes with a name and value each. Any Application Descriptor must have a unique set of selector attributes. Make sure to be specific enough, so that the test program in use identifies the respective Application Descriptor correctly.

Device name: tesqa33

Test program name: ANY

Product family: ANY

Test program revision: ANY

Figure 3-61. Save Application Descriptor Changes

3.3.20 Delete 1:1 Application Descriptor

NOTE: Only Organization Administrators have permission to create Application Descriptors.

1. Log in with Organization Administrator credentials and click on **Applications** > **1:1 Application Descriptors**.

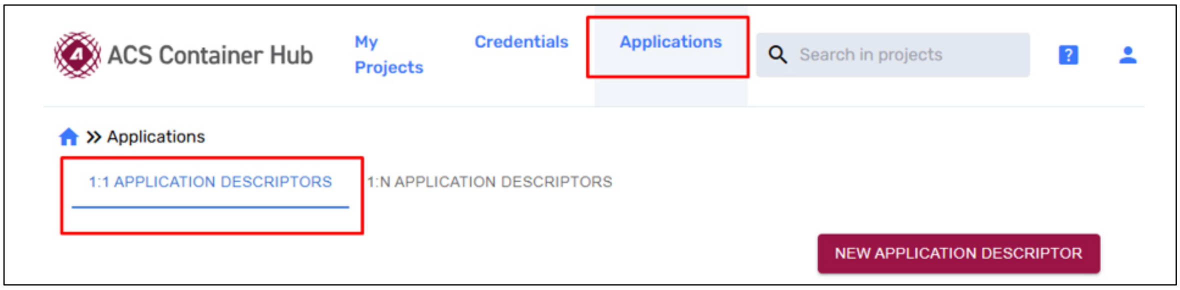
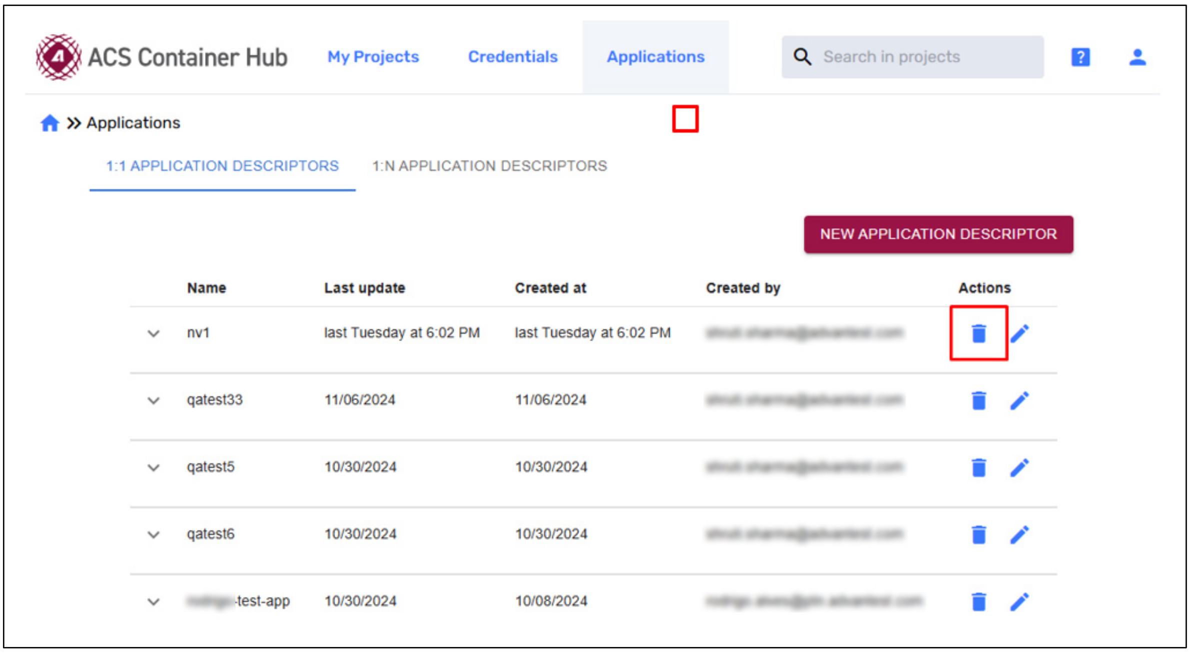


Figure 3-62. 1:1 Application Descriptors

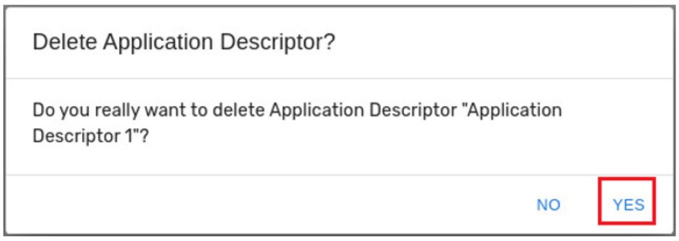
2. Click the Delete icon that is associated with the Application Descriptor.



	Name	Last update	Created at	Created by	Actions
⌄	nv1	last Tuesday at 6:02 PM	last Tuesday at 6:02 PM	[redacted]	🗑 ✎
⌄	qatest33	11/06/2024	11/06/2024	[redacted]	🗑 ✎
⌄	qatest5	10/30/2024	10/30/2024	[redacted]	🗑 ✎
⌄	qatest6	10/30/2024	10/30/2024	[redacted]	🗑 ✎
⌄	[redacted]-test-app	10/30/2024	10/08/2024	[redacted]	🗑 ✎

Figure 3-63. Application Descriptor Delete Icon

3. A confirmation dialog displays. Click **Yes** to confirm the delete action.


Delete Application Descriptor?

Do you really want to delete Application Descriptor "Application Descriptor 1"?

Figure 3-64. Confirm Application Descriptor Delete

3.3.21 Multi-User Permissions for Managing App Descriptors or Organization Project

Organization Administrators have the capability to provide multi-user permissions that allow Service users to manage Application Descriptors or an Organization Project. Only one of the following roles may be assigned:

- **Developer**
  Provides permissions for user to access and manage Application Descriptors
- **Project Administrator**
  Provides permissions for user to access and manage the Organization Project

1. Log in with Organization Administrator user credentials, then navigate to the Organization Project.

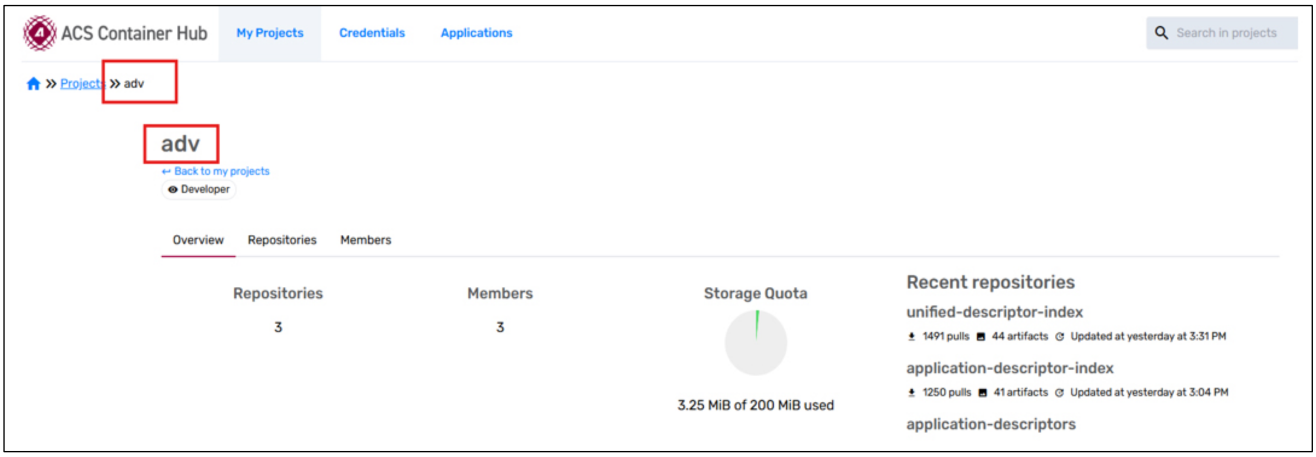


Figure 3-65. Navigate to Organization Project

2. Click on the *Members* tab and then click the **Add Members** button.

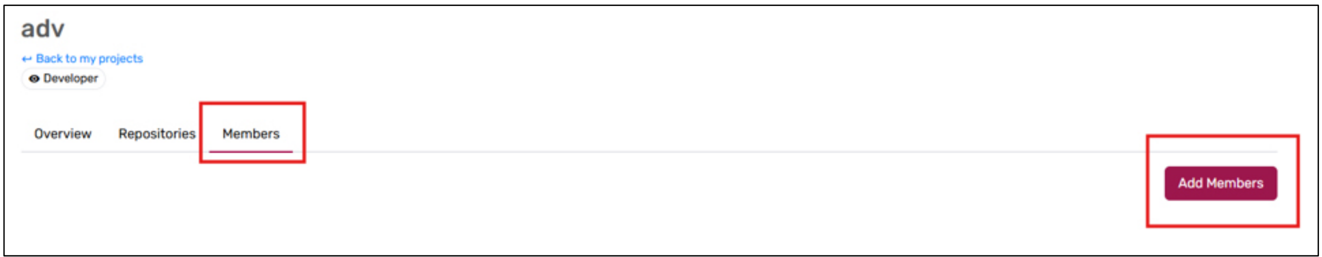


Figure 3-66. Members Tab

3. In the *Add Members To Project* dialog, select a user from the User dropdown.

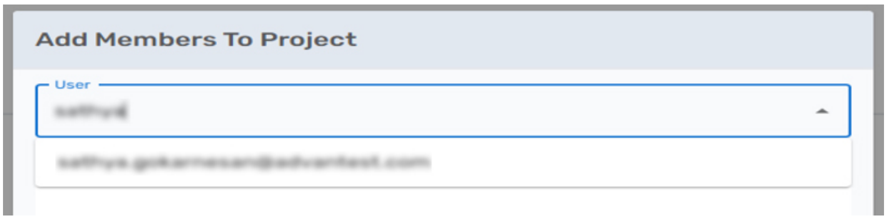
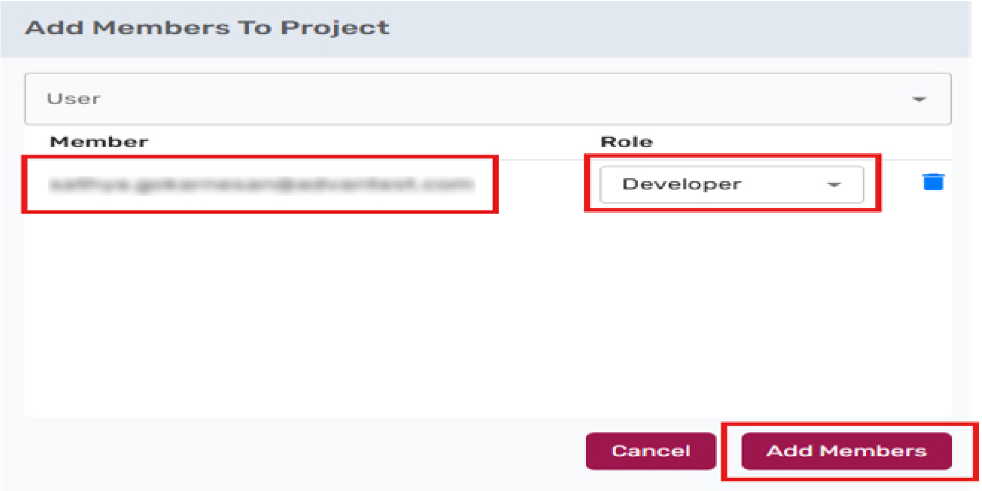


Figure 3-67. Select a User

4. To provide Service users with permissions to manage Application Descriptors, select **Developer** from the Role dropdown and click the **Add Members** button.

   To give Service users Project Administrator privilege for the Organization Project, select **Admin** from the Role dropdown and click the **Add Members** button.



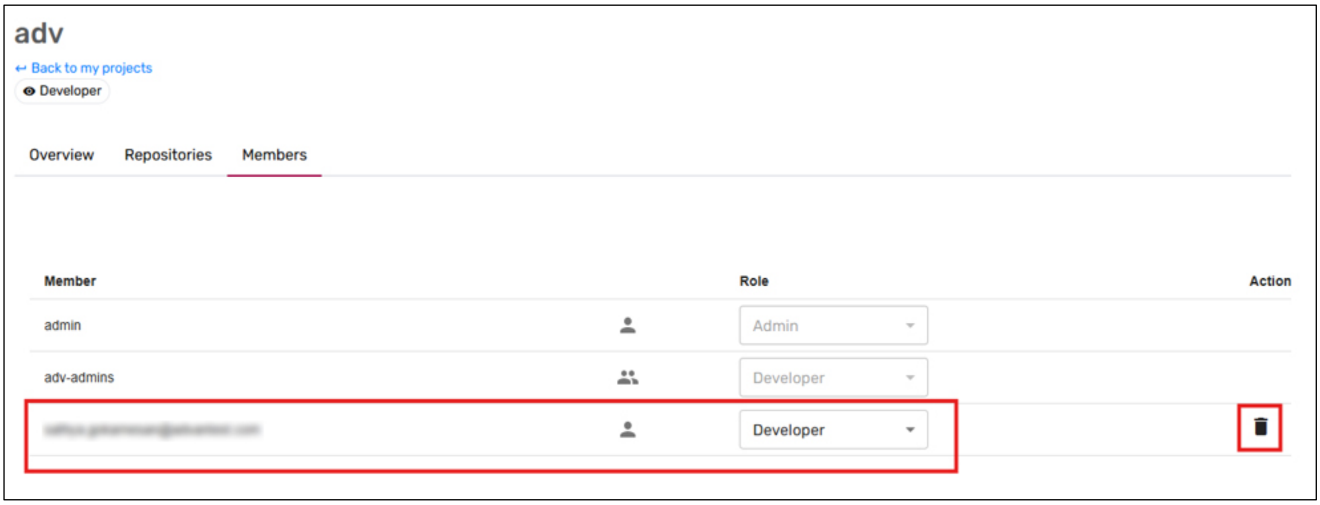
Add Members To Project

User

Member	Role
[redacted]	Developer

Figure 3-68. Assign Service User Role

5. Upon successful completion, the newly added Service user will be visible under the *Members* tab. To revoke granted permission for any user, click the Delete icon. To change the role for a user, select the appropriate option from the Role dropdown.



adv

↩ Back to my projects

Developer

Member	Role	Action
admin	Admin	
adv-admins	Developer	
[redacted]	Developer	🗑

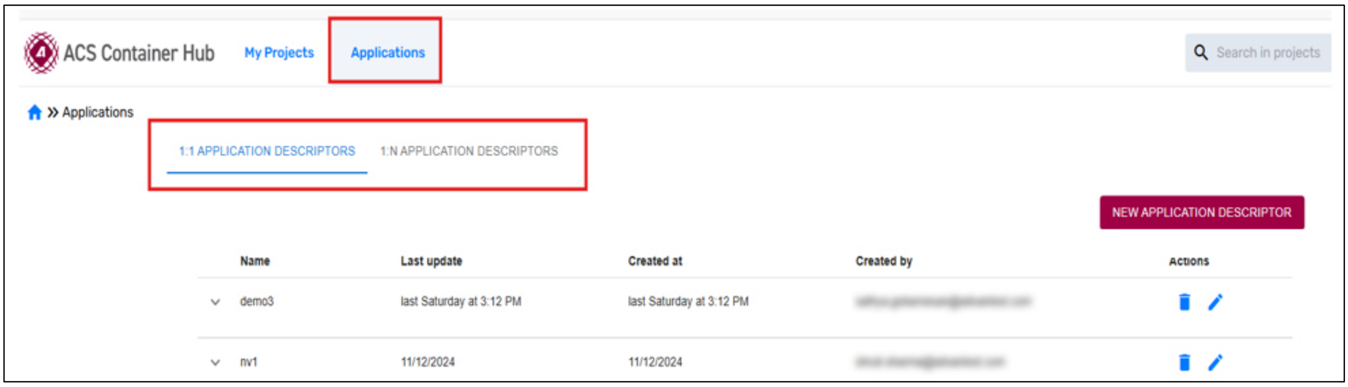
Figure 3-69. Newly Added Service User and Delete Icon

3.3.22 Using Multi-User Permissions

For Service users that have been granted permissions to manage Application Descriptors or manage the Organization Project (see page 260), this section provides guidance on how Service users can get started in the newly assigned role.

Developer (Application Descriptors)

1. Log in to the Container Hub with user credentials.
2. On successful login, click the **Applications** tab to display the *1:1 APPLICATION DESCRIPTORS* and *1:N APPLICATION DESCRIPTORS*.



ACS Container Hub
My Projects
Applications
Search in projects
Applications
1:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

	Name	Last update	Created at	Created by	Actions
⌄	demo3	last Saturday at 3:12 PM	last Saturday at 3:12 PM	[redacted]	🗑 ✎
⌄	nv1	11/12/2024	11/12/2024	[redacted]	🗑 ✎

Figure 3-70. Application Descriptors

3. See the following sections for instructions on managing Application Descriptors:
   - Create 1:1 Application Descriptor
   - Update 1:1 Application Descriptor
   - Delete 1:1 Application Descriptor
   - Support for 1:N Applications
   - Viewing 1:N Application Status

Project Admin (Organization Project)

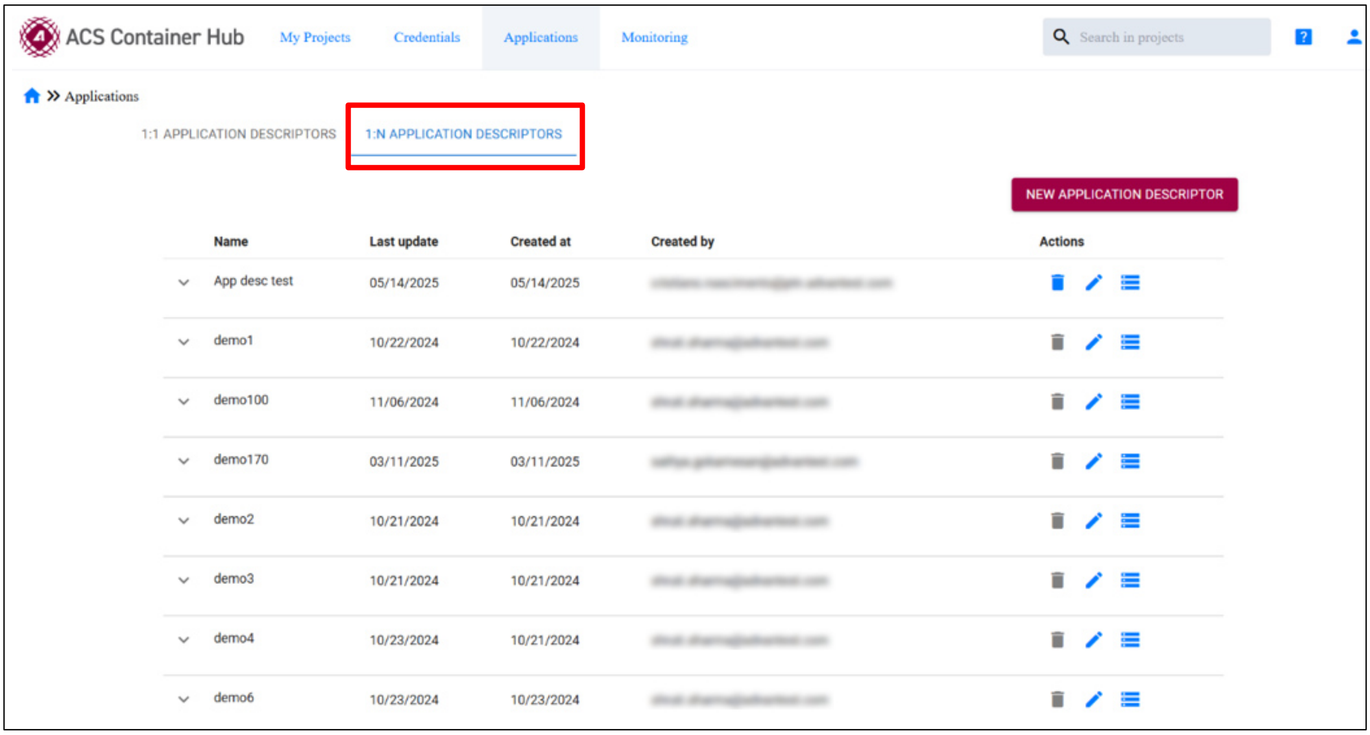
Container Hub users with Project Admin privileges can assign other users to the role of Developer or Project Admin, as described in Multi-user Permissions for Application Descriptors or Organization Project.

3.3.23 Support for 1:N Applications

The 1:N Application Descriptor displays information about applications running on the Unified Server. All information is collected automatically, so there is no need for any user input.

Click on the **1:N Application Descriptor** tab to display a table similar to Figure 3-71 below.

NOTE: The information shown is read-only. No editing is required or allowed. The delete icon (in the Actions column) is only active if all instances of the application are stopped (see the "Actions" column in Viewing 1:N Application Status).



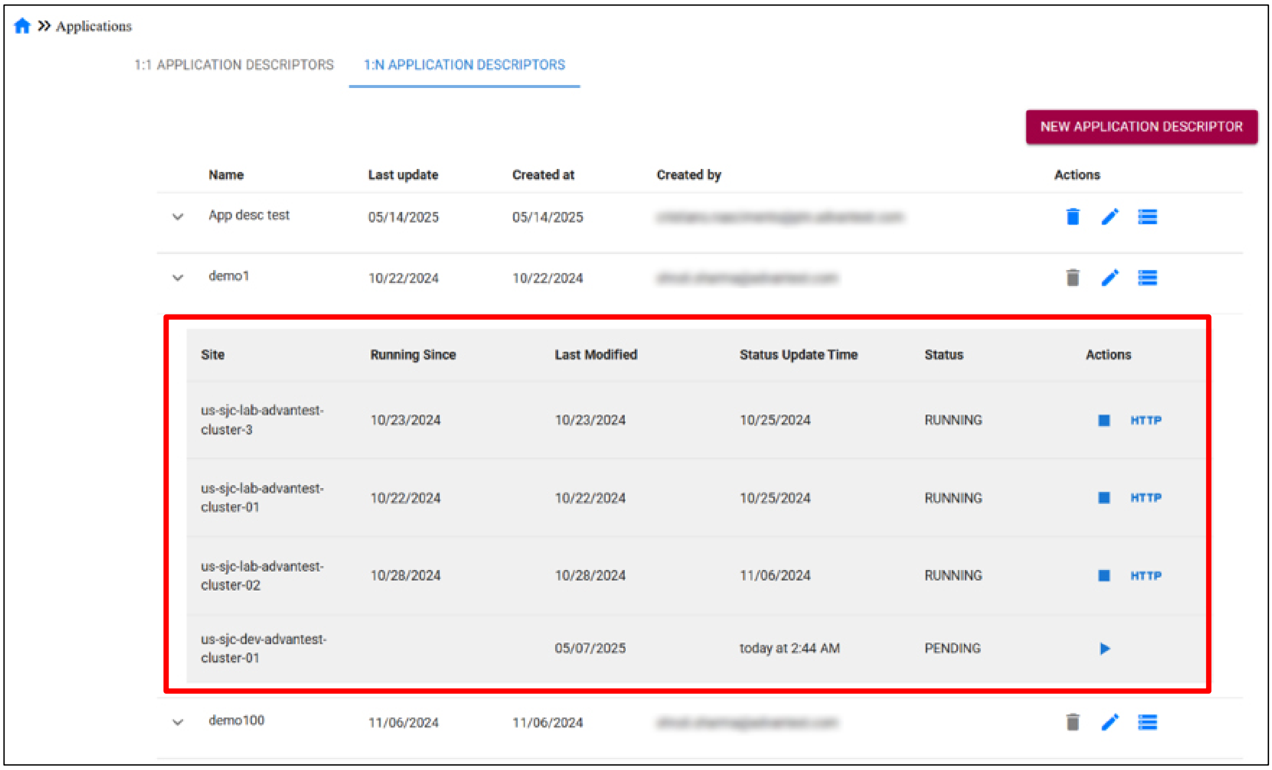
	Name	Last update	Created at	Created by	Actions
⌄	App desc test	05/14/2025	05/14/2025	[redacted]	🗑 ✎ ☰
⌄	demo1	10/22/2024	10/22/2024	[redacted]	🗑 ✎ ☰
⌄	demo100	11/06/2024	11/06/2024	[redacted]	🗑 ✎ ☰
⌄	demo170	03/11/2025	03/11/2025	[redacted]	🗑 ✎ ☰
⌄	demo2	10/21/2024	10/21/2024	[redacted]	🗑 ✎ ☰
⌄	demo3	10/21/2024	10/21/2024	[redacted]	🗑 ✎ ☰
⌄	demo4	10/23/2024	10/21/2024	[redacted]	🗑 ✎ ☰
⌄	demo6	10/23/2024	10/23/2024	[redacted]	🗑 ✎ ☰

Figure 3-71. 1:N Application Descriptors

3.3.24 Viewing 1:N Application Status

To display status information for the 1:N application running on different servers, click the server icon (in the Actions column) for the application descriptor. A table will display with information specific to that application, as show below.

NOTE: The delete icon is only active when all instances of the application are stopped.



» Applications

1:1 APPLICATION DESCRIPTORS | 1:N APPLICATION DESCRIPTORS

NEW APPLICATION DESCRIPTOR

Name	Last update	Created at	Created by	Actions
App desc test	05/14/2025	05/14/2025	[redacted]	
demo1	10/22/2024	10/22/2024	[redacted]	
demo100	11/06/2024	11/06/2024	[redacted]	

Site	Running Since	Last Modified	Status Update Time	Status	Actions
us-sjc-lab-advantest-cluster-3	10/23/2024	10/23/2024	10/25/2024	RUNNING	■ HTTP
us-sjc-lab-advantest-cluster-01	10/22/2024	10/22/2024	10/25/2024	RUNNING	■ HTTP
us-sjc-lab-advantest-cluster-02	10/28/2024	10/28/2024	11/06/2024	RUNNING	■ HTTP
us-sjc-dev-advantest-cluster-01		05/07/2025	today at 2:44 AM	PENDING	▶

Figure 3-72. 1:N Application Status

The 1:N application status information is displayed in a 6-column table. Each row in the table represents one single instance running in a specific Unified Server.

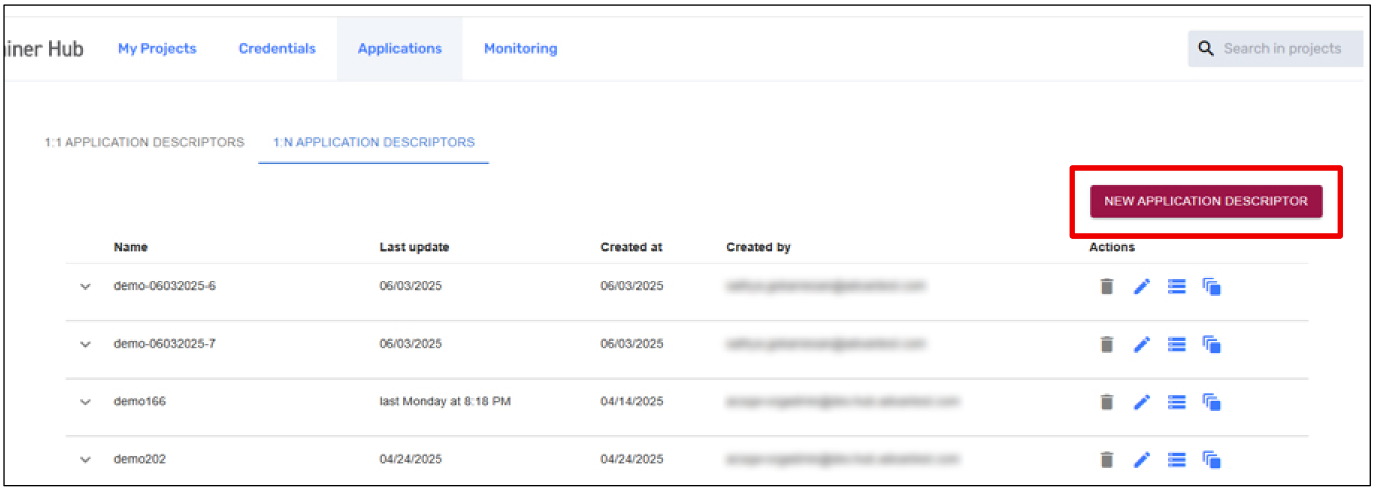
- **Unified Server ID**
  The identifier of the Unified Server running an instance of the application.
- **Running Since**
  The date when the application ran for the first time.
- **Last Modified**
  The date when the application was last deployed.
- **Status Update Time**
  The date when the status of the application was last modified
- **Status**
  The status of the application, which will either be RUNNING or STOPPED.
- **Actions**
  The actions to be performed on the application. There are two possible actions that can be performed, either START or STOP. If the current status of the application is RUNNING, a "STOP" action can be performed. If the current status of the application is STOPPED, a "START" action can be performed. When executing an action, a confirmation dialog will display to confirm the execution action. Select YES to confirm the action, or NO to cancel execution.

3.3.25 Viewing, Creating, and Editing 1:N Application Descriptors

To view, create and edit 1:N Application Descriptors, log in with Organization Administrator credentials and click on **Applications > 1:N Application Descriptors**.

Creating 1:N Application Descriptors

To create a new 1:N Application Descriptor, click the **NEW APPLICATION DESCRIPTOR** button. Refer to Create 1:1 Application Descriptor for more information.

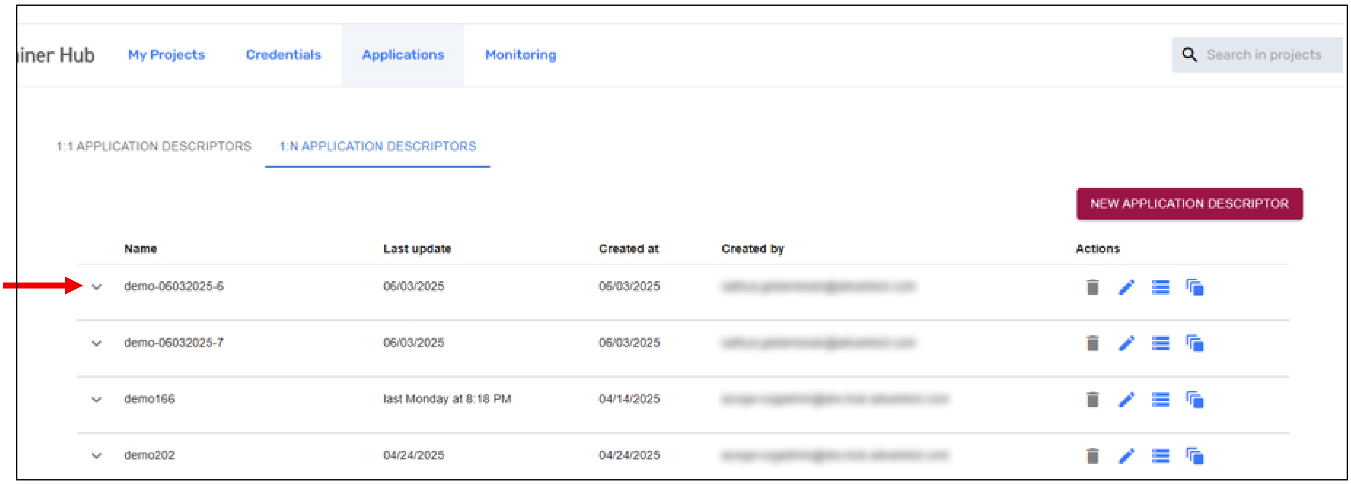

iner Hub
My Projects
Credentials
Applications
Monitoring
Search in projects
1:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

	Name	Last update	Created at	Created by	Actions
⌄	demo-06032025-6	06/03/2025	06/03/2025	[redacted]	🗑 ✎ ☰ ⧉
⌄	demo-06032025-7	06/03/2025	06/03/2025	[redacted]	🗑 ✎ ☰ ⧉
⌄	demo166	last Monday at 8:18 PM	04/14/2025	[redacted]	🗑 ✎ ☰ ⧉
⌄	demo202	04/24/2025	04/24/2025	[redacted]	🗑 ✎ ☰ ⧉

Figure 3-73. New Application Descriptor Button

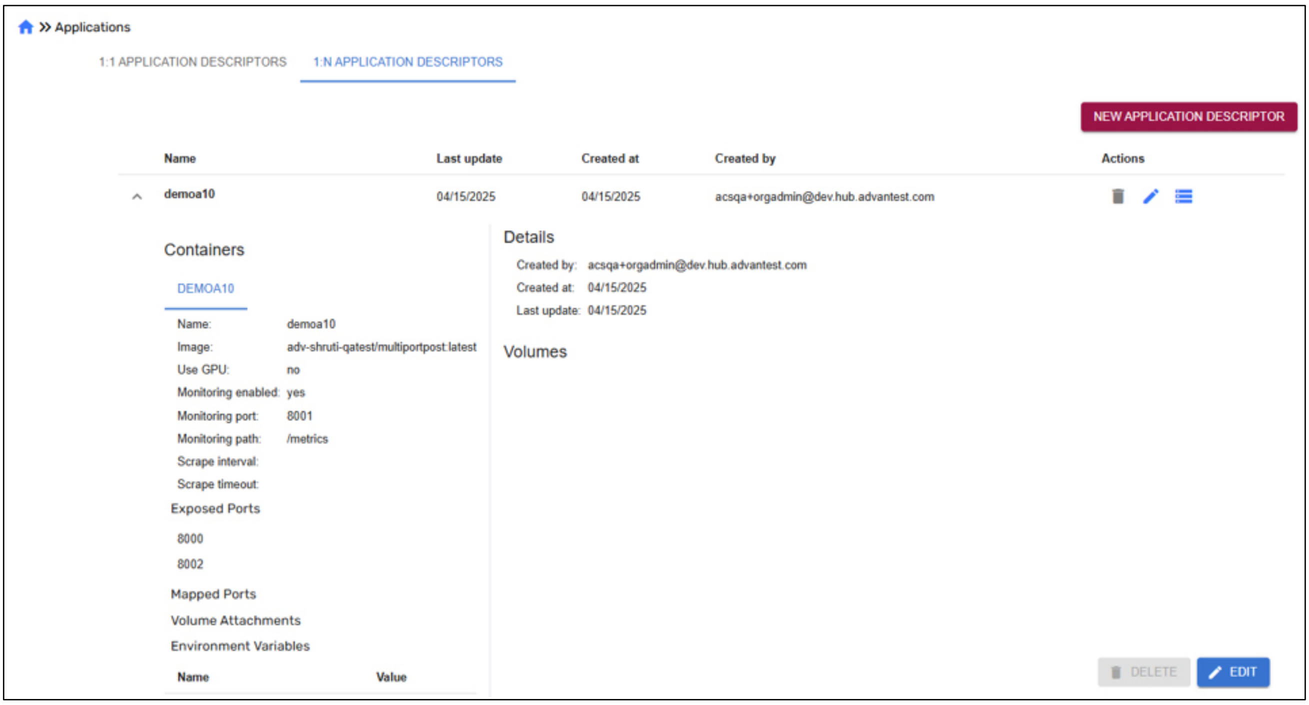
Viewing 1:N Application Descriptor Details

To view expanded details for 1:N Application Descriptors, click the expand chevron (downward arrow at the far left of the descriptor) as indicated in the figure below. Application Descriptor details then display as illustrated in Figure 3-75.


iner Hub
My Projects
Credentials
Applications
Monitoring
Search in projects
1:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

	Name	Last update	Created at	Created by	Actions
⌄	demo-06032025-6	06/03/2025	06/03/2025	[redacted]	🗑 ✎ ☰ ⧉
⌄	demo-06032025-7	06/03/2025	06/03/2025	[redacted]	🗑 ✎ ☰ ⧉
⌄	demo166	last Monday at 8:18 PM	04/14/2025	[redacted]	🗑 ✎ ☰ ⧉
⌄	demo202	04/24/2025	04/24/2025	[redacted]	🗑 ✎ ☰ ⧉

Figure 3-74. Expand Chevron


Applications
1:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

Name	Last update	Created at	Created by	Actions
demoa10	04/15/2025	04/15/2025	acsqa+orgadmin@dev.hub.advantest.com	

Containers

DEMOA10

Name: demoa10
Image: adv-shruti-qatest/multiportpost:latest
Use GPU: no
Monitoring enabled: yes
Monitoring port: 8001
Monitoring path: /metrics
Scrape interval:
Scrape timeout:

Exposed Ports

8000
8002

Mapped Ports
Volume Attachments
Environment Variables

Name	Value

Details

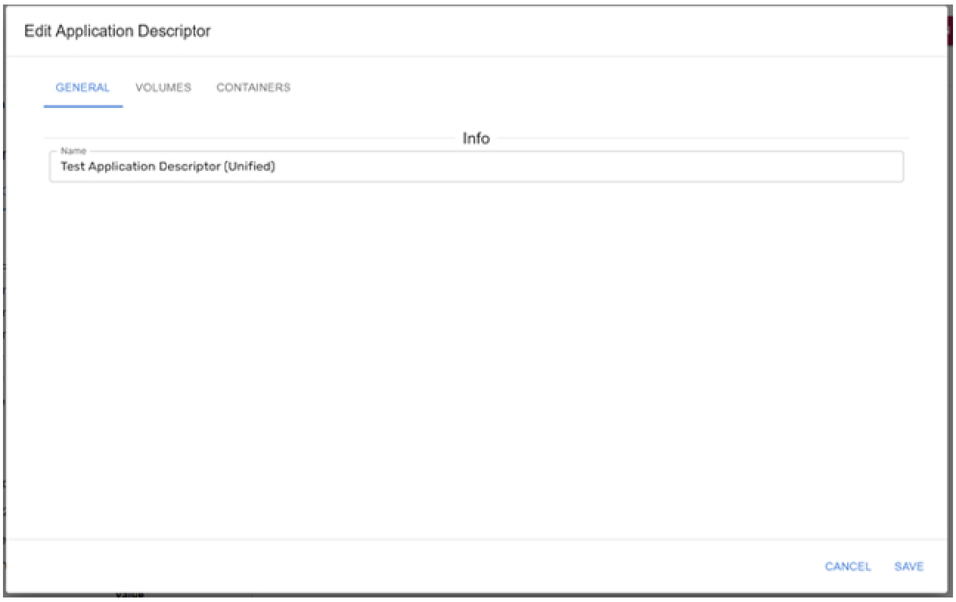
Created by: acsqa+orgadmin@dev.hub.advantest.com
Created at: 04/15/2025
Last update: 04/15/2025

Volumes

Figure 3-75. Application Descriptor Expanded View

Updating 1:N Application Descriptors

To update an existing 1:N Application Descriptor, click the edit icon at the right of the Application Descriptor to open the *Edit Application Descriptor* dialog.


Edit Application Descriptor

Info

Name: Test Application Descriptor (Unified)

Figure 3-76. Edit Application Descriptor Dialog

3.3.26 Test Applications for 1:N Application Descriptors

The Test Application feature enables users to send HTTP requests directly to docker applications running in a specific server. This can be useful for verifying application endpoints, debugging, or interacting with your service.

Availability Requirements

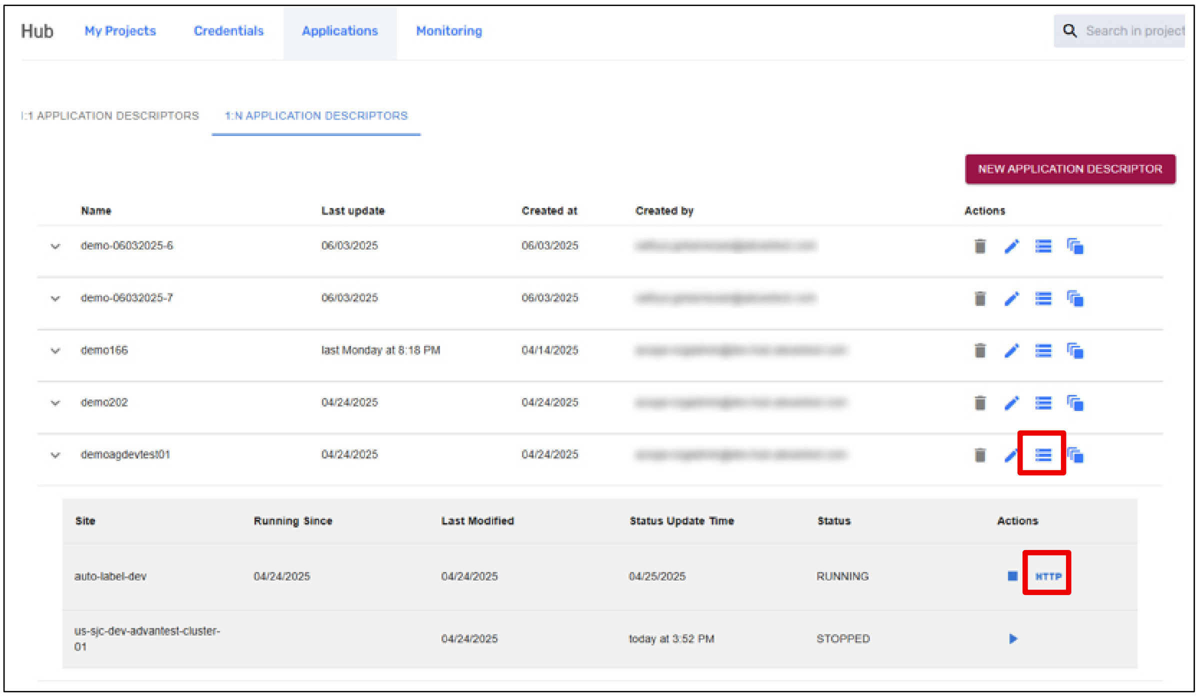
The Test App feature is only available if both of the following conditions are met:

- The application is running on the server
- The application exposes ports that are accessible via HTTP

Accessing the Test App User Interface

To open the HTTP requests UI,steps below.

1. Navigate to the **Application Descriptors** menu
2. Select the **1:N Application Descriptors** tab
3. Click the **Sites** icon to view detailed information about the application status. If the application is running and exposes public ports, the **HTTP** button will appear next to the Stop Application button.


Hub
My Projects
Credentials
Applications
Monitoring
Search in project
I:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

Name	Last update	Created at	Created by	Actions
demo-06032025-6	06/03/2025	06/03/2025	[redacted]	
demo-06032025-7	06/03/2025	06/03/2025	[redacted]	
demo166	last Monday at 8:18 PM	04/14/2025	[redacted]	
demo202	04/24/2025	04/24/2025	[redacted]	
demoagdevtest01	04/24/2025	04/24/2025	[redacted]	

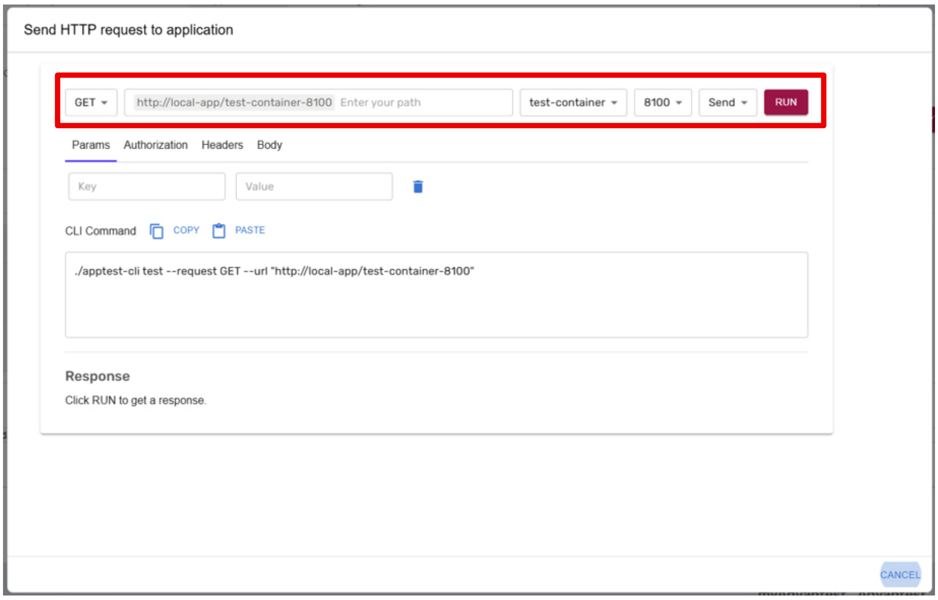
Site	Running Since	Last Modified	Status Update Time	Status	Actions
auto-label-dev	04/24/2025	04/24/2025	04/25/2025	RUNNING	■ HTTP
us-sjc-dev-advantest-cluster-01		04/24/2025	today at 3:52 PM	STOPPED	▶

Figure 3-77. Sites Icon and HTTP Button

4. Click the **HTTP** button to open the Test App UI (Figure 3-78).

Using the Test App UI

At the top of the UI, users can configure and send an HTTP request. The components are described below.



```
./apptest-cli test --request GET --url "http://local-app/test-container-8100"
```

Click RUN to get a response.

Figure 3-78. Test App UI – HTTP Request Configuration

1 HTTP Method

Select an HTTP method from the dropdown. Selection options include GET, POST, PUT, DELETE, and PATCH.

2 Endpoint Configuration

- **Hostname:** This is the name of the server where the app is running. It is pre-filled and read-only.
- **Container-Port Path:** The first part of the path is auto-generated based on the selected container and port in the format:

  `<container-name>-<port>`
- **Additional Path:** A text input where users can add custom path segments after the container-port string.

3 Container and Port Selection

If multiple containers are available, a **Container** dropdown will be shown.

If a container exposes multiple ports, a **Port** dropdown will also be available.

4 Action Selection

A dropdown to choose one of the following actions:

`Send:` Sends the HTTP request

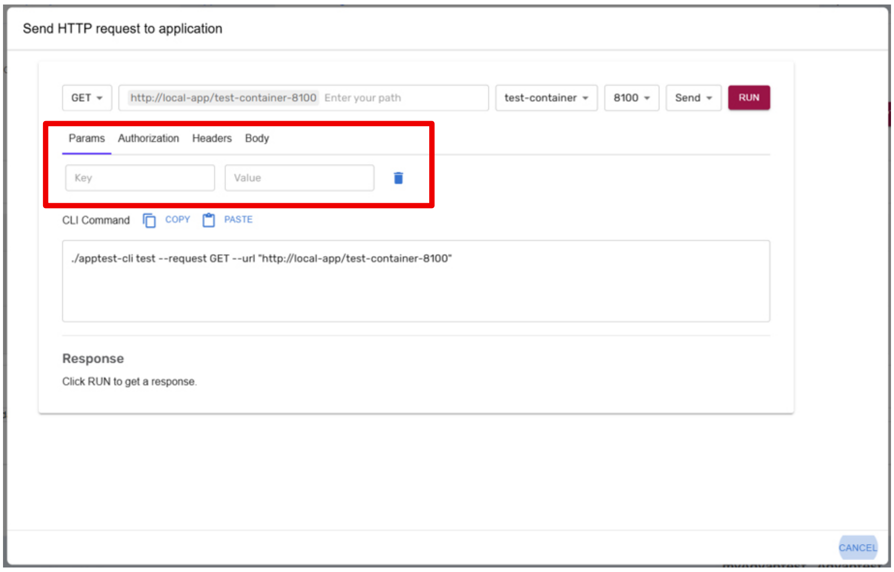
`Clear Form:` Resets the form to its initial state

5 RUN button

Executes the selected action (`Send` or `Clear Form`).

Configuring HTTP Request Details

Below the top controls, you’ll find a set of tabs to configure various aspects of the HTTP request. These options allow you to fully customize the HTTP request to match the behavior and requirements of your application.



```
./apptest-cli test --request GET --url "http://local-app/test-container-8100"
```

Click RUN to get a response.

Figure 3-79. Test App UI – Configuring HTTP Request Details

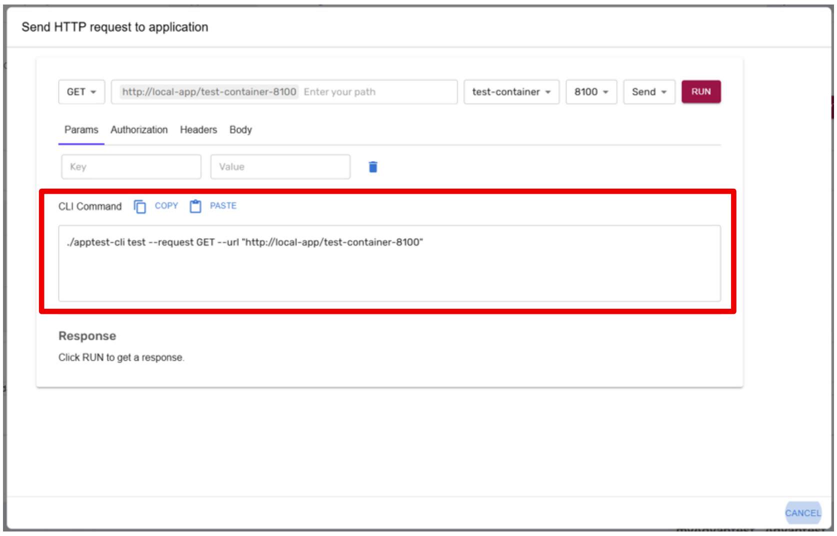
- **Params:** Add query parameters to be included in the request URL. You can enter multiple Key-Value pairs. Additional rows can be added as needed.
- **Authorization:** Configure authentication credentials:
  - No Auth
  - Basic Auth (username and password)
  - Token (e.g., Bearer token)
- **Headers:** Define custom HTTP headers to include in the request. You can enter multiple Key-Value pairs.
- **Body:** Enter raw data to be sent in the request body, typically in JSON format (available for methods like POST, PUT, PATCH).

CLI Command

This area displays the command in CLI format. As the user inserts the values on the forms at the top of the screen, this area will be updated automatically

The command can be copied using the COPY button. A command can also be pasted in this field using the PASTE button. When a valid command is pasted in this field, the values in the upper section of the Test App UI will be updated based on the values from the command.

A command can also be directly entered into this field. As with the PASTE button, the values in the upper section of the Test App UI will be updated based on the values directly entered in this field, after a 2-second delay.



```
./apptest-cli test --request GET --url "http://local-app/test-container-8100"
```

Figure 3-80. Test App UI – CLI Command

Example Command

`./apptest-cli test-application --request PUT --url "http://us-sjc-lab-advantest-cluster-01/demoa10-8000/test" --header "Content-Type: JavaScript" --data '{"name": "PutTest"}'`

Below is a description of the above example command:

`./apptest-cli` The entry point. This is the command-line interface (CLI) tool script or executable file you're invoking.

`test-application` A subcommand or action the CLI tool will perform. In this case, it likely triggers a test run for an application endpoint.

`--request PUT` Specifies the HTTP request method to use. Here, it's PUT, which is generally used to update a resource.

`--url` `Error! Hyperlink reference not valid.` The target **URL** to which the request will be sent. This is the endpoint of the application under test.

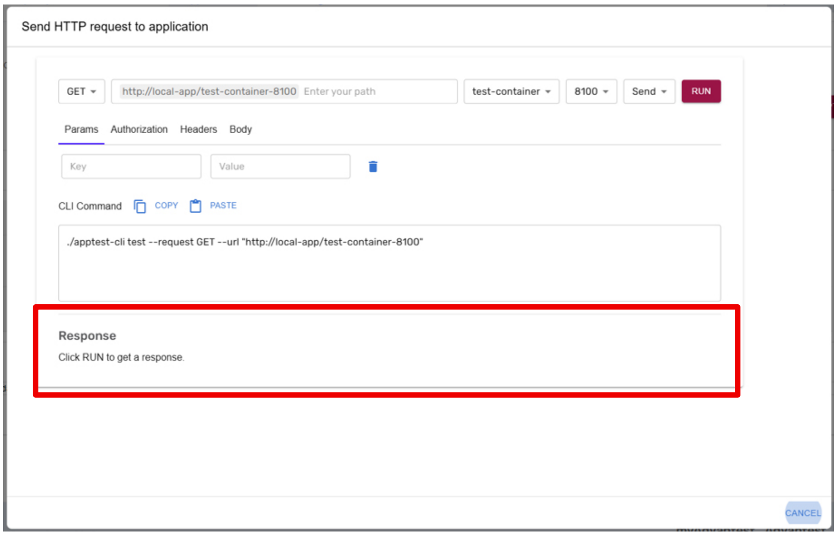
`--header "Content-Type: JavaScript"` Adds a **custom HTTP header** to the request. In this example, it sets Content-Type to JavaScript. This is likely a placeholder—typically you'd use application/json.

`--data '{"name": "PutTest"}'` The body of the HTTP request. This sends a JSON payload, commonly used in PUT and POST requests. Here it updates the name field with the value "PutTest"

Response

After clicking the RUN button (at the top-right of the Test App UI) and a response is received from the API, all the following information will be displayed in this area:

- The response status code
- The duration the application took to process the request (in milliseconds)
- The body of the response



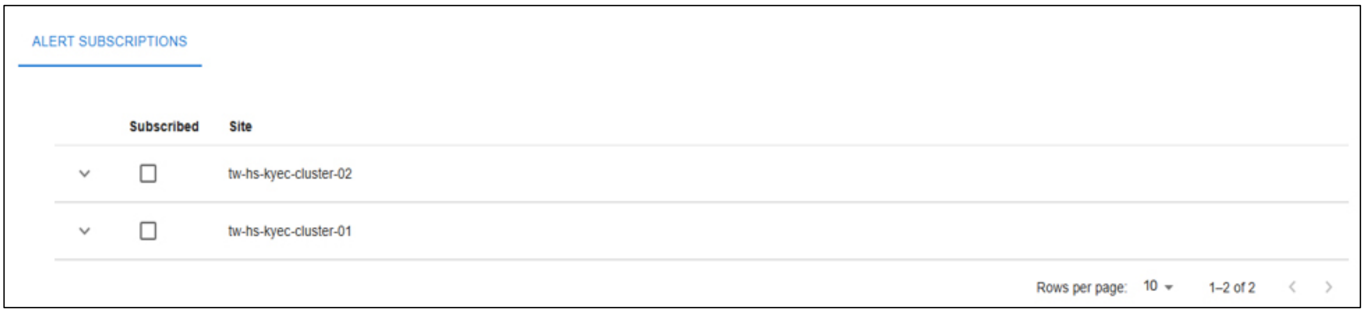
```
./apptest-cli test --request GET --url "http://local-app/test-container-8100"
```

Click RUN to get a response.

Figure 3-81. Test App UI – Response

3.3.27 Monitoring – Alert Subscriptions

The Monitoring tab allows you to view and configure Alert Subscriptions. From the Alert Subscriptions section, you can view a list of available sites for which alerts are configured, as well view and modify the subscription status for individual alerts within each respective site.



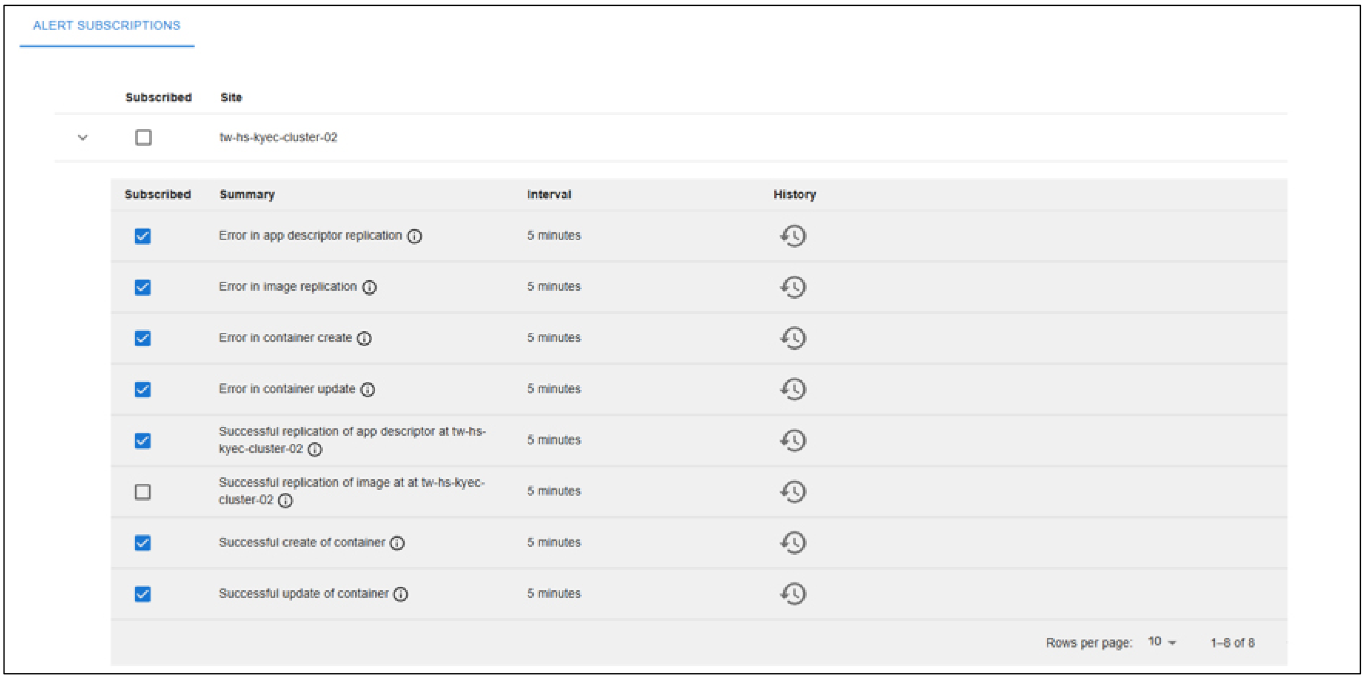
ALERT SUBSCRIPTIONS

	Subscribed	Site
⌄	☐	tw-hs-kyec-cluster-02
⌄	☐	tw-hs-kyec-cluster-01

Rows per page: 10 ▾ 1–2 of 2 ‹ ›

Figure 3-82. Alert Subscriptions

Click on a site to expand the view and see the individual alert details for that site.



ALERT SUBSCRIPTIONS

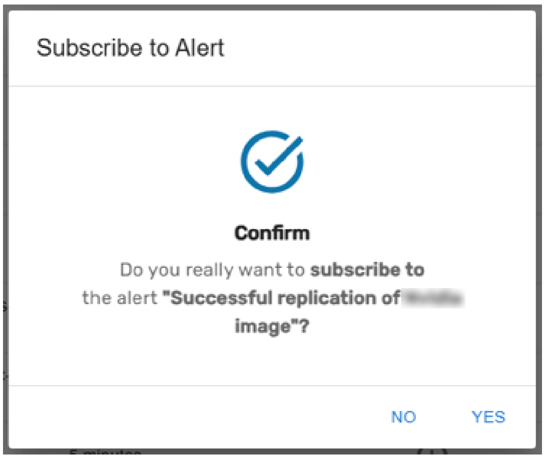
	Subscribed	Site
⌄	☐	tw-hs-kyec-cluster-02

Subscribed	Summary	Interval	History
☑	Error in app descriptor replication ⓘ	5 minutes	⟲
☑	Error in image replication ⓘ	5 minutes	⟲
☑	Error in container create ⓘ	5 minutes	⟲
☑	Error in container update ⓘ	5 minutes	⟲
☑	Successful replication of app descriptor at tw-hs-kyec-cluster-02 ⓘ	5 minutes	⟲
☐	Successful replication of image at at tw-hs-kyec-cluster-02 ⓘ	5 minutes	⟲
☑	Successful create of container ⓘ	5 minutes	⟲
☑	Successful update of container ⓘ	5 minutes	⟲

Rows per page: 10 ▾ 1–8 of 8

Figure 3-83. Alert Subscriptions – Expanded Site

In the site expanded view, you can subscribe to (or unsubscribe from) specific alerts by clicking the checkbox corresponding to that alert. Upon clicking the checkbox, a confirmation dialog displays, from which you can confirm or decline the action.



Subscribe to Alert

Confirm

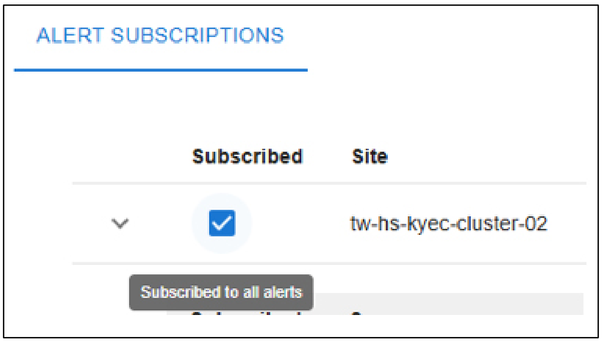
Do you really want to **subscribe to** the alert **"Successful replication of [redacted] image"?**

Figure 3-84. Confirmation Dialog

The expanded details for a site includes the following information:

- **Subscribed**
  This shows the subscription status for the alert. A check mark in the checkbox indicates that the user is subscribed to this alert. If unchecked, the user is not subscribed.

  NOTE: If a user is subscribed to all the alerts in the site's expanded view, the Subscribed checkbox for that site is enabled, as shown in the figure below. If a user is unsubscribed to any one of the alerts for that site, the Subscribed checkbox for that site is disabled.

  ALERT SUBSCRIPTIONS

	Subscribed	Site
⌄	☑	tw-hs-kyec-cluster-02

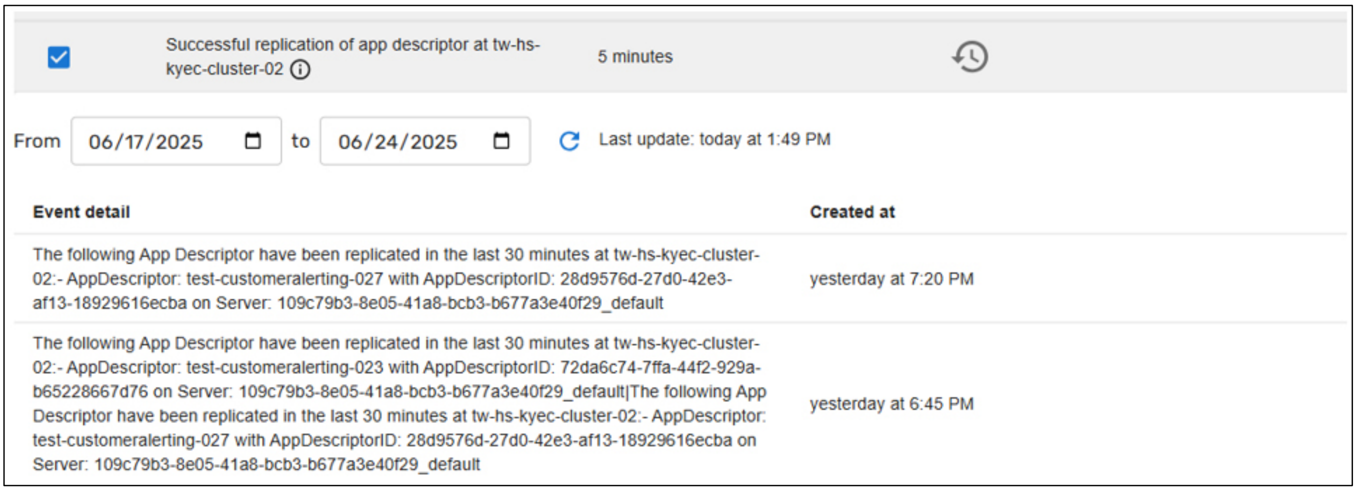
  Subscribed to all alerts

  For all subscribed alerts, users will receive an email from Advantest Alerts (alerts@advantest.com) that includes information about the alert event, as shown in . The level provided event information (Summary, Description, and Details) in the email varies for each alert.

- **Summary**
  Provides a summary of the alert. Hover over the information icon at the right of the summary to display a full description of the alert.

- **Interval**
  Indicates the time interval for the alert.

- **History**
  Expands the alert section to display the history for the alert, as illustrated in Figure 3-85.



☑ Successful replication of app descriptor at tw-hs-kyec-cluster-02 5 minutes

From 06/17/2025 to 06/24/2025 Last update: today at 1:49 PM

Event detail	Created at
The following App Descriptor have been replicated in the last 30 minutes at tw-hs-kyec-cluster-02:- AppDescriptor: test-customeralerting-027 with AppDescriptorID: 28d9576d-27d0-42e3-af13-18929616ecba on Server: 109c79b3-8e05-41a8-bcb3-b677a3e40f29_default	yesterday at 7:20 PM
The following App Descriptor have been replicated in the last 30 minutes at tw-hs-kyec-cluster-02:- AppDescriptor: test-customeralerting-023 with AppDescriptorID: 72da6c74-7ffa-44f2-929a-b65228667d76 on Server: 109c79b3-8e05-41a8-bcb3-b677a3e40f29_default|The following App Descriptor have been replicated in the last 30 minutes at tw-hs-kyec-cluster-02:- AppDescriptor: test-customeralerting-027 with AppDescriptorID: 28d9576d-27d0-42e3-af13-18929616ecba on Server: 109c79b3-8e05-41a8-bcb3-b677a3e40f29_default	yesterday at 6:45 PM

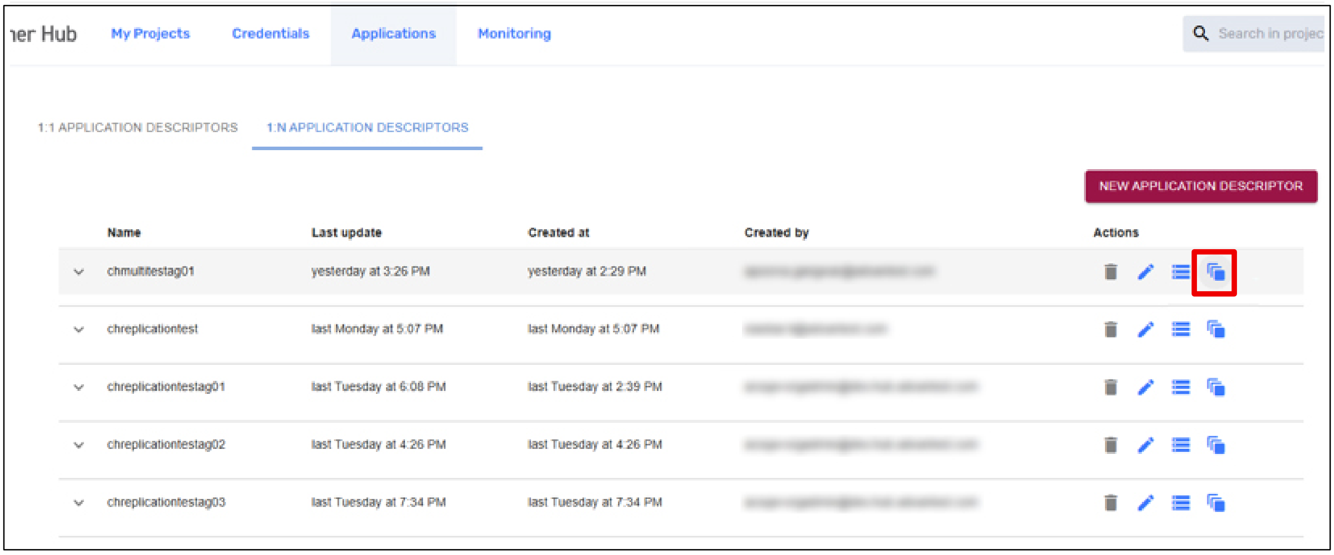
Figure 3-85. Alert History

The **Event detail** provides information about the replicated image and other replication information for that alert. This information provided varies for each alert.

By default, the most current information is displayed. However, the *From* and *to* date filter fields can be used to display historical information available within the selected date range. After selecting a date range, click the refresh button (to the right of the date fields) to see the alert events filtered for the specified date range.

3.3.28 Replication

Replication allows the user to replicate the application descriptor and related image across multiple sites. To use the Replication button, navigate to **Applications** menu and select **1:N Application Descriptors**.


ner Hub
My Projects
Credentials
Applications
Monitoring
Search in projec
1:1 APPLICATION DESCRIPTORS
1:N APPLICATION DESCRIPTORS
NEW APPLICATION DESCRIPTOR

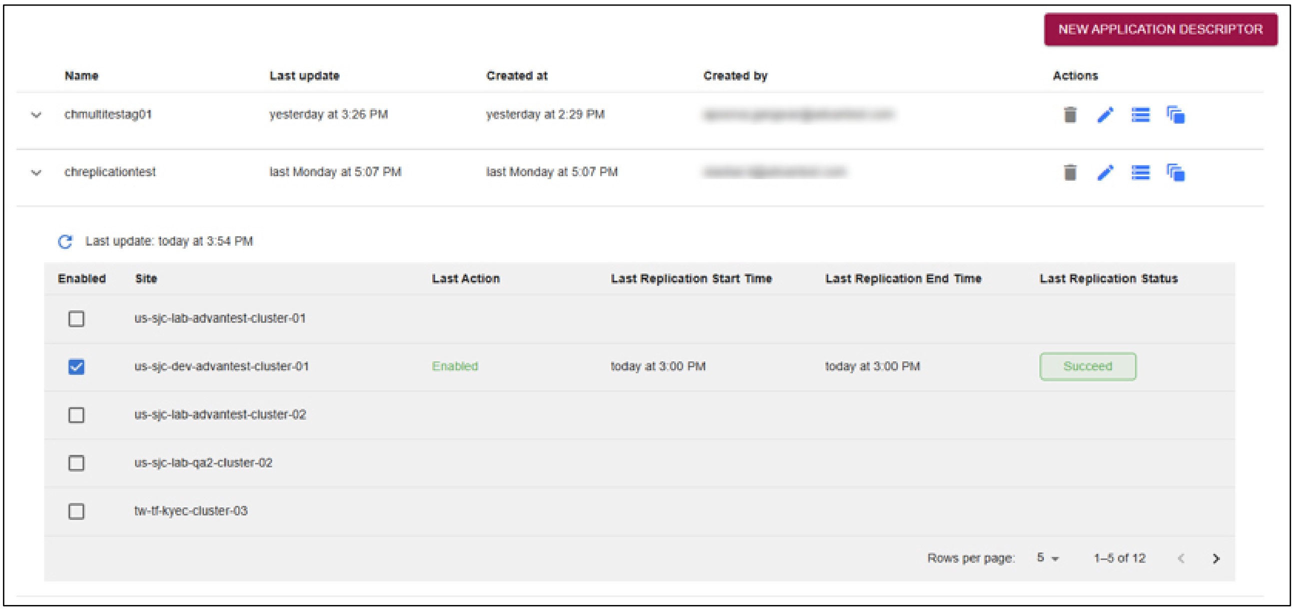
Name	Last update	Created at	Created by	Actions
chmultitestag01	yesterday at 3:26 PM	yesterday at 2:29 PM	[redacted]	
chreplicationtest	last Monday at 5:07 PM	last Monday at 5:07 PM	[redacted]	
chreplicationtestag01	last Tuesday at 6:08 PM	last Tuesday at 2:39 PM	[redacted]	
chreplicationtestag02	last Tuesday at 4:26 PM	last Tuesday at 4:26 PM	[redacted]	
chreplicationtestag03	last Tuesday at 7:34 PM	last Tuesday at 7:34 PM	[redacted]	

Figure 3-86. Replication Button

Click the replication button to display a table that lists sites where replication can be enabled or disabled. As shown in, the replication table displays:

- Enable / Disable checkbox for enabling and disabling replication
- Last Action column informing the result of the last enable / disable action
- Last Replication Start Time column informing when the last replication has started
- Last Replication End Time column informing when the last replication has finished
- Last Replication Status informing the status of the last replication.

NOTE: The Last Replication Start Time, End Time, and Status columns will only be visible when replication is enabled for the site.



	Name	Last update	Created at	Created by	Actions
⌄	chmultitestag01	yesterday at 3:26 PM	yesterday at 2:29 PM	[redacted]	
⌄	chreplicationtest	last Monday at 5:07 PM	last Monday at 5:07 PM	[redacted]	

Last update: today at 3:54 PM

Enabled	Site	Last Action	Last Replication Start Time	Last Replication End Time	Last Replication Status
□	us-sjc-lab-advantest-cluster-01				
☑	us-sjc-dev-advantest-cluster-01	Enabled	today at 3:00 PM	today at 3:00 PM	Succeed
□	us-sjc-lab-advantest-cluster-02				
□	us-sjc-lab-qa2-cluster-02				
□	tw-tf-kyec-cluster-03				

Rows per page: 5 1–5 of 12

Figure 3-87. Replication Table

Enabling and Disabling Replication

Use the Enabled checkbox to enable or disable replication. Upon enabling replication, a confirmation message displays to confirm the action. If confirmed, the Container Hub will proceed with the replication process and the Last Action column will indicate "Enable In Progress" until replication has completed.

When the replication process completes (or fails), the Last Action column displays the results of the action. To get new updates, click the Refresh button.

NOTE: When enabling replication for a site, note the following:

- While replication is in process, the Enable checkbox cannot be toggled until the process has completed.
- When using replication for a site, multiple descriptors should not be setup using the same image.
- When replication is enabled on an application descriptor, it must be disabled to choose a different image.
- The replication feature currently applies only to the first container and will not work for the other containers.
- If enable/disable replication fails, please contact the Advantest support team.

4. ACS Edge Server

The ACS Edge Server is a high-performance and highly secure edge compute and analytics server that, when integrated into an existing test cell, enables ultra-fast algorithmic (AI, machine learning, and statistical) decision making.

Once integrated into an existing test cell, the main function of the ACS Edge Server is to act as a repository for container images that are pulled from the ACS Unified Server. Container images are pulled into the ACS Edge Server using APIs provided by Advantest. For information on the provided APIs, see C++ Client API Reference.

4.1 ACS Edge Server Operation

4.1.1 Operation Overview

The ACS Edge Server is always on and connected to the 93K Host Controller’s 10Gbps 4-port NIC via Cat6a Ethernet cable. The client SDK is integrated into the customers' test program to communicate with the ACS Edge Server via API calls. The client SDK is integrated into the ACS Nexus for communicating with the ACS Edge Server Containerized ML/HPC workloads (Apps), which are loaded onto the Edge Server via Test Program/Nexus during runtime.

NOTE 1: Based on customer preferences, the ACS Edge Server may or may not have pre-loaded container images.

NOTE 2: Container images can be pulled from local Unified Server (the on-premise Mirror Registry) or directly from the ACS Container Hub (external network).

The figure below illustrates the data/communication flow for the ACS Edge server within the RTDI environment.

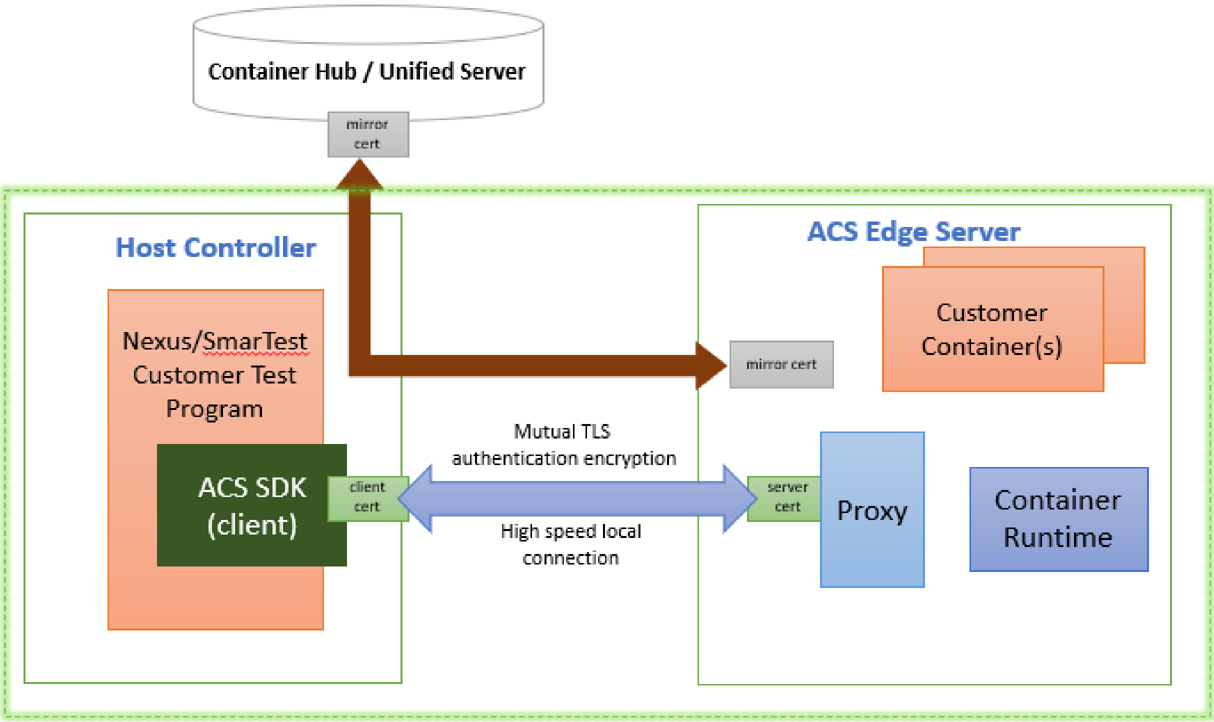
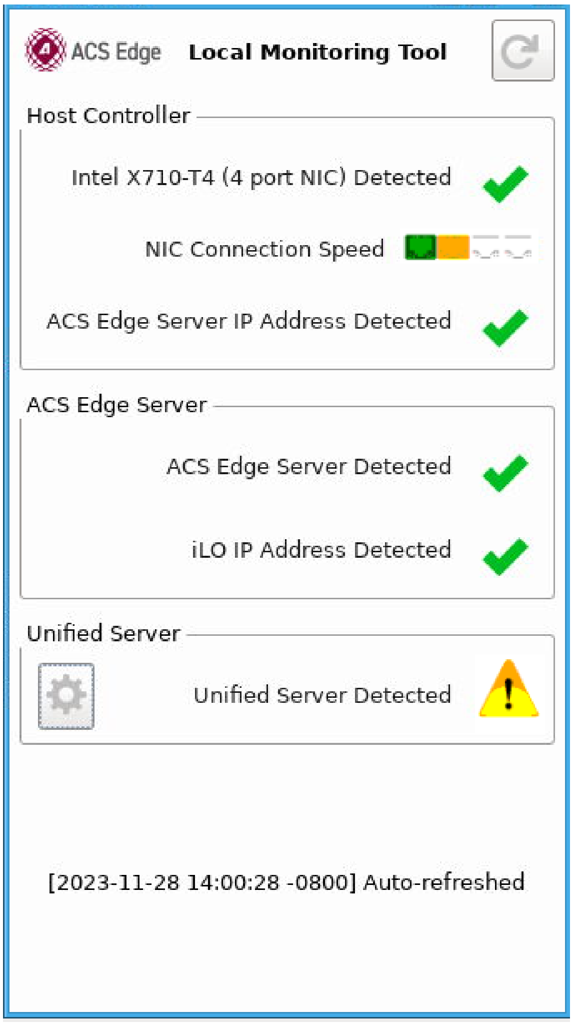


Figure 4-1. ACS Edge Server Data/Communication Flow

4.1.2 Monitoring the ACS Edge Server

The Local Monitoring Tool (LMT) is a server monitoring application that is installed on each Host Workstation. As shown in Figure 4-2, the LMT provides visual, real-time, and continuous monitoring of a server's critical sub-components, including:

- NIC hardware
- NIC connection speed
- Edge Server IP address
- Edge Server connectivity
- iLO IP address
- Unified Server connectivity



Intel X710-T4 (4 port NIC) Detected ✓

ACS Edge Server IP Address Detected ✓

ACS Edge Server Detected ✓

iLO IP Address Detected ✓

Unified Server Detected ⚠

[2023-11-28 14:00:28 -0800] Auto-refreshed

Figure 4-2. Local Monitoring Tool (LMT)

The LMT will automatically start when logging into the 93K Host Workstation. To manually start the LMT, access the RHEL OS start menu and navigate to **Application → Advantest V93000 → LMT**.

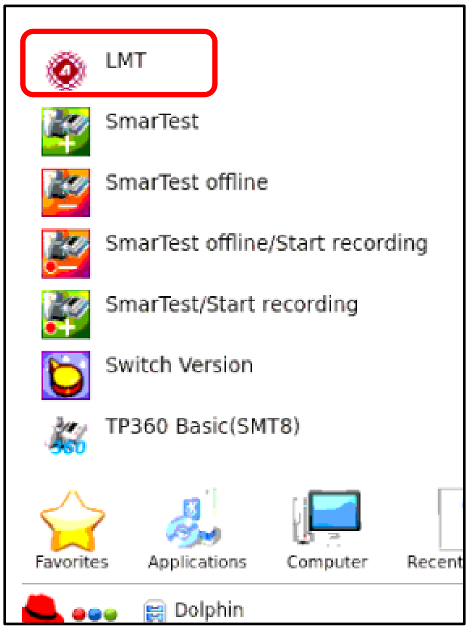


Figure 4-3. LMT Manual Start

4.2 ACS Edge Server Features

4.2.1 Security

- Server is physically locked in a mini-rack
- Server access is password protected
- SSH, display, and USB are disabled
- Secure/encrypted connection using mTLS 1.2+ / https
- Advantest PKI, security certificates refresh
- Application authorization using signed test program manifest
- Encryption of container image contents
- Latest security updates for BIOS, iLO, Ubuntu OS, and app packages
- Customer Application and Data stored in non-persistent storage

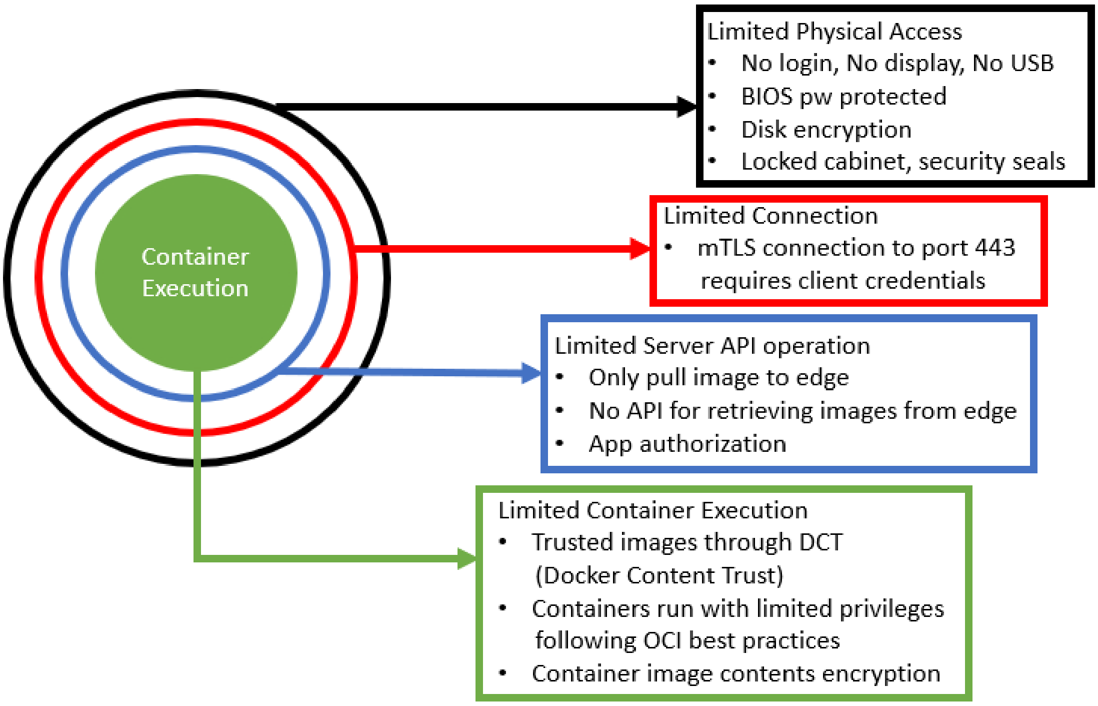

Container Execution
Limited Physical Access
• No login, No display, No USB
• BIOS pw protected
• Disk encryption
• Locked cabinet, security seals
Limited Connection
• mTLS connection to port 443 requires client credentials
Limited Server API operation
• Only pull image to edge
• No API for retrieving images from edge
• App authorization
Limited Container Execution
• Trusted images through DCT (Docker Content Trust)
• Containers run with limited privileges following OCI best practices
• Container image contents encryption

Figure 4-4. ACS Edge Security Features

4.2.2 Accessibility

The ACS Edge Server is controlled and operated by the Edge client SDK or ACS Nexus. Access to the Edge Server is allowed only through the client SDC, ACS Nexus, and Advantest (ADV) support tools.

4.2.3 Functionality

Functionality attributes of the ACS Edge Server include:

- Dedicated high performance compute solution for test operations
    - Offload in-line inferencing, computing, and machine learning (ML) capabilities or other compute-intensive applications from the Host Controller to the ACS Edge Server
        - Pull container images from the on-premise ACS Unified Server or the external ACS Container Hub.
        - Run application as a Docker image
    - Increase overall equipment efficiency (OEE) of the RTDI system by freeing up the Host Controller to be devoted to test execution control
    - AI Hardware Accelerator Compatible
- Open Container Initiative (OCI) Architecture for robust and reliable production deployment
- Ultra-Low latency (<10ms)
    - Operate near real-time on data generated by tests in the test program
- Service APIs for on-field server configurations
- Collect Edge Server system metrics, system logs, and customer application logs to share with the ACS Unified Server for monitoring capability. Refer to Table 4-1 and Table 4-2 for list of each data type and corresponding description.

Table 4-1. Edge Server Metrics

Name	Description
`edge_server_serverapi_up`	Server API is up and running. 0 for a down or 1 for up.
`edge_server_serviceAPI_up`	Service API is up and running. 0 for a down or 1 for up.
`traefik_entrypoint_request_duration_seconds`	Request processing duration histogram on an endpoint.
`traefik_entrypoint_requests_tls_total`	The total count of HTTPS requests handled by an entry point.
`traefik_service_open_connections`	The amount of open connections that exist on a service, partitioned by method and protocol.
`traefik_config_reloads_total`	Configuration reloads.
`traefik_config_reloads_failure_total`	Configuration failure reloads.
`traefik_config_last_reload_failure`	The last configuration reload failure.
`redis_up`	Shard is up and running.
`redis_used_memory`	Memory used by shard.
`edge_server_memory_usage`	The CPU utilization in percentage.
GPU Metrics	All GPU metrics

Table 4-2. Edge Server Events

Name	Description
Unauthorized Access	Unauthorized clients tried to communicate with the Edge Server.
Unified Server Unreachable	Unable to reach the Unified Server
Image Download	The Edge Server downloads an image.
Image Delete	The Edge Server deletes an image.
Container Create	The Edge Server creates a container with a specific configuration.
Container Delete	Edge Server deletes a container.
Application failed to start	Application failed to start.
Volume Create	The Edge Server creates a volume.
MirrorRegistry Config	Add container registry URL/hostname into the Edge Server.
App Auth Config	Enable/disable Application Authorization in the Edge Server.
Trusted CA Add	Add customer trust CA into the Edge Server.
Trust CA Delete	Remove customer trust CA from the Edge Server.

Edge Server Logs

Logs are collected to share with the Unified Server from the following Edge Server components

- Application
- Traefik
- Redis
- Server API
- Service API

For more information on the attributes collected for each data type (logs, results, metrics, events), refer to section 5.3.1.

4.2.4 Redundancy

Edge redundancy ensures continuous container deployment on the ACS Unified Server in case of ACS Edge Server failure. Redundancy features include:

- Automatic failover
    - If the Edge Server is unreachable for a test, the application container can be started on the Unified Server instead with the application descriptor specified, and all traffic will be redirected to the Unified Server hosted application.
    - The Edge client SDK will return error code 7 (ACSE_COULDNT_CONNECT) which triggers an automatic failover.
- Automatic switchover
    - If the ACS Edge Server is reachable again, the application container can be resumed on the Edge server. The cleanup process will automatically remove deployed containers and used volumes on the Unified Server and switch it back to the Edge Server.
- Tester identification
    - Only allowed testers can use server API and deploy containers on the Unified Server. Testers can be registered with Management CLI and registry admin API.
- Multi-tester support
    - Default capacity for the number of testers that can deploy containers on the Unified Server is set to 1. The 'Capacity' can be replaced with Management CLI.
- Smartest7 client SDK (C++) support to communicate with containers deployed on the Unified Server.
    - Traefik routing

        -
          ```
          string image_name = "acsev2-test-multiport-test";

          string tag = "latest";
          ACSEdgeConn& conn = ACSEdgeConn::getInstance();
          conn.setIp("10.120.111.54") //set Unified server IP address
          ContainerConfig contConfig;
          ports.push_back("8000");
          ports.push_back("8001");
          ports.push_back("8002");
          // configure the container by adding tester hostname and expose container
          port
          contConfig.setOption("tester_id","tester1").setOption("ports",ports)
          ACSEDGEcode  status_code = conn.imagePull(image_name, tag,response);
          status_code =
          conn.containerCreateStart(image_name,"multiporttest",contConfig,
          response);
          ```
        - Server API on the Unified Server will create a traefik routing rule (HTTP/HTTPS) dynamically, based on its container name (or hostname) and port. By default, if the request contains a single port, the endpoint will be `unifiedserver.local/<container_name>/` regardless of which port is used by the customer app inside the container.
        - If there are multiple ports used by the customer app inside the container, the endpoint will be `unifiedserver.local/<container_name>-<port>/`.

- Host port mapping

    -
      ```
      string image_name = "acsev2-test-multiport-test";
      string tag = "latest";
      string publish_port1 = "8001:8003/tcp"; //<container_port>:<host_port>
      ACSEdgeConn& conn = ACSEdgeConn::getInstance();
      conn.setIp("10.120.111.54") //set Unified server IP address
      ContainerConfig contConfig;
      vector<string> ports;
      ports.push_back(publish_port1);
      // configure the container by adding tester_id and expose container port
      contConfig.setOption("tester_id","tester1").setOption("publish-ports",
      ports);
      ACSEDGEcode  status_code = conn.imagePull(image_name, tag,response);
      status_code = conn.containerCreateStart(image_name,"multiport-
      test",contConfig,response);
      ```
    - Direct mapping container’s port to host’s port provides client ability to use custom protocol in addition to http/https which is only allowed with traefik routing method.
    - Available ports on the Unified Server: 8000 to 8010. Note that the example above is mapping the host port (8001) with container port (8003/tcp).

      NOTE: When the application is started on the Unified Server, the state of the application on the Edge Server is not migrated to the Unified Server.

4.2.5 Remote Service Upgrade

By utilizing the ACS Unified Server harbor, ACS Container Hub, and Remote Activities UI, the internal services (containers) running on the Edge Server and manifest file can be updated every 24 hours manually by enabling remote service upgrade.

Currently, all services (container images, Helm charts) are grouped and released under a single ISO version (for example, 3.3.0). When the ISO version changes, all related services (containers, deployments, images, configs) are upgraded or synchronized at once, based on that version. This means that individual service images cannot be upgraded independently on the ACS Edge server. All services must be upgraded together by switching to the desired ISO version.

The following services can be upgraded:

- serverapi
- serviceapi
- securityapi
- nexus-broker

Follow the procedure below to perform remote service upgrade:

1. Use the ACT tool to enable remote-upgrade. This will bootstrap the flux manifests file for the 4 services listed above.

2. Obtain a new ACS Edge ISO version from the ACS team.

3. Replicate images and manifests in unifiedserver harbor from containerhub.

4. Open a browser and go to:

   `https://unifiedserver.ui.local/harbor-ui/`

5. Click the replication menu on left sidebar and look for:

   - `adv-acs-edge-images` (and click the **Replicate** button)
   - `adv-acs-edge-manifests` (and click the **Replicate** button)

6. Once replication is completed successfully, go to the remote activities UI and click **terminal** on the top menu bar.

7. Select the unifiedserver ID and Edge server ID from the side bar menu. In the terminal, enter the following command for the service upgrade:

   ```
   /opt/acsev2-server/scripts/patch_flux_manifests.sh {tag}
   ```

8. Enter the following command to verify the update:

   ```
   kubectl -n default get helmrelease
   ```

4.3 ACS Edge Server Specifications

Table 4-3. Hardware Specifications

Processor	Intel Xenon CPU 3.0 GHz
Memory	128 GB
Storage	480 GB x2
Power	800W
Ethernet	10 Gigabit Ethernet Port x2
Operating System	Ubuntu 22.04 LTS

Table 4-4. Z4G4 Customer Workstation Slot Configuration with X710-T4 NIC

Slot	Type	Customer Workstation Port Configuration
1	PCIe 3 x 16	
2	PCIe 3 x 4 PCH	
3	PCIe 3 x 16	
4	PCIe 3 x 4 PCH	
5	**PCIe 3 x 8**	**Intel X710-T4**

Table 4-5. Z640 Customer Workstation Slot Configuration with X710-T4 NIC

Slot	Type	Customer Workstation Port Configuration
1	PCI Express (Gen2) x4 (1)	
2	PCI Express (Gen3) x16	
3	PCI Express (Gen2) x8 (4)	
4	**PCI Express (Gen3) x8**	**Intel X710-T4**
5	PCI Express (Gen3) x16	
6	PCI 32/33	

5. ACS Unified Server

The ACS Unified Server is a multipurpose server that supports compute and storage, application service, and database storage. It also provides True Zero Trust security for the test floor, acting as the secure gatekeeper for the ACS Edge Server and ACS Nexus, to securely communicate between the ACS Container Hub. Some of the key features of the ACS Unified Server include:

- Scalable and redundant compute and storage.
- Secured data exchange.
- Host in-house, partner, and customer applications.
- Secured container launch onto ACS Edge Server.
- Automatic pre-population of container images from the ACS container hub.

When installed on the test floor, the main function of the ACS Unified Server is to act as a mirror server for the ACS Container Hub. Container images can be pulled from the ACS Unified Server to the ACS Edge Server using APIs provided by Advantest. For information on pull operations and other APIs, refer to C++ Client API Reference.

5.1 Container Registry

The ACS Unified Server includes a Container Registry which can hold container images to be downloaded and run on the ACS Edge Servers. The Container Registry can be setup to pull container images from the ACS Container Hub either on demand or periodically.

5.2 Licensing

The Unified Server includes a licensing service that, when enabled, will limit access to customer applications to a fixed number of IP addresses. The Unified Server license checkout service queries the license server for the number of available licenses for the customer application. When a request is initially made to the customer application from a specific IP address, the middleware checks if there are any licenses available. If there are, then a license is checked out and the request is permitted. If no licenses are available, the request is blocked.

So long as the IP address is included in the list of licensed IP addresses, all requests to the customer application are permitted. The licensed IP addresses are valid for a default period of 24 hours. If no requests are received from an IP address within a 24-hour period, that IP address is removed from the list of licensed IP address and that license is released. All licensing logs are included in the monitoring logs, as described in the monitoring logs section below.

The license server running on the Unified Server is accessible to external clients (or license checkout services). To access the Unified Server license server from an external client, do the following:

1. Set the url in the `/etc/fne/client.conf` on the local client to **`unifiedserver.local`**.

2. Set the server name to **`unifiedserver.local`** in the `/opt/flexlm/license/offline.lic` file.

3. Add the IP address of the Unified Server load balancer in the `/etc/hosts` file, as shown below:

```
[root@fne-client ~]# cat /etc/fne/client.conf
server
{
    name=”FNE on central server”
    url=unifiedserver.local
}

[root@fne-client ~]# cat /opt/flexlm/license/offline.lic
SERVER unifiedserver.local 000000000000
DAEMON socbu
USE_SERVER

[root@fne-client ~]# cat /etc/hosts
127.0.0.1 localhost
10.120.230.158 unifiedserver.local #IP address of cluster load balancer
```

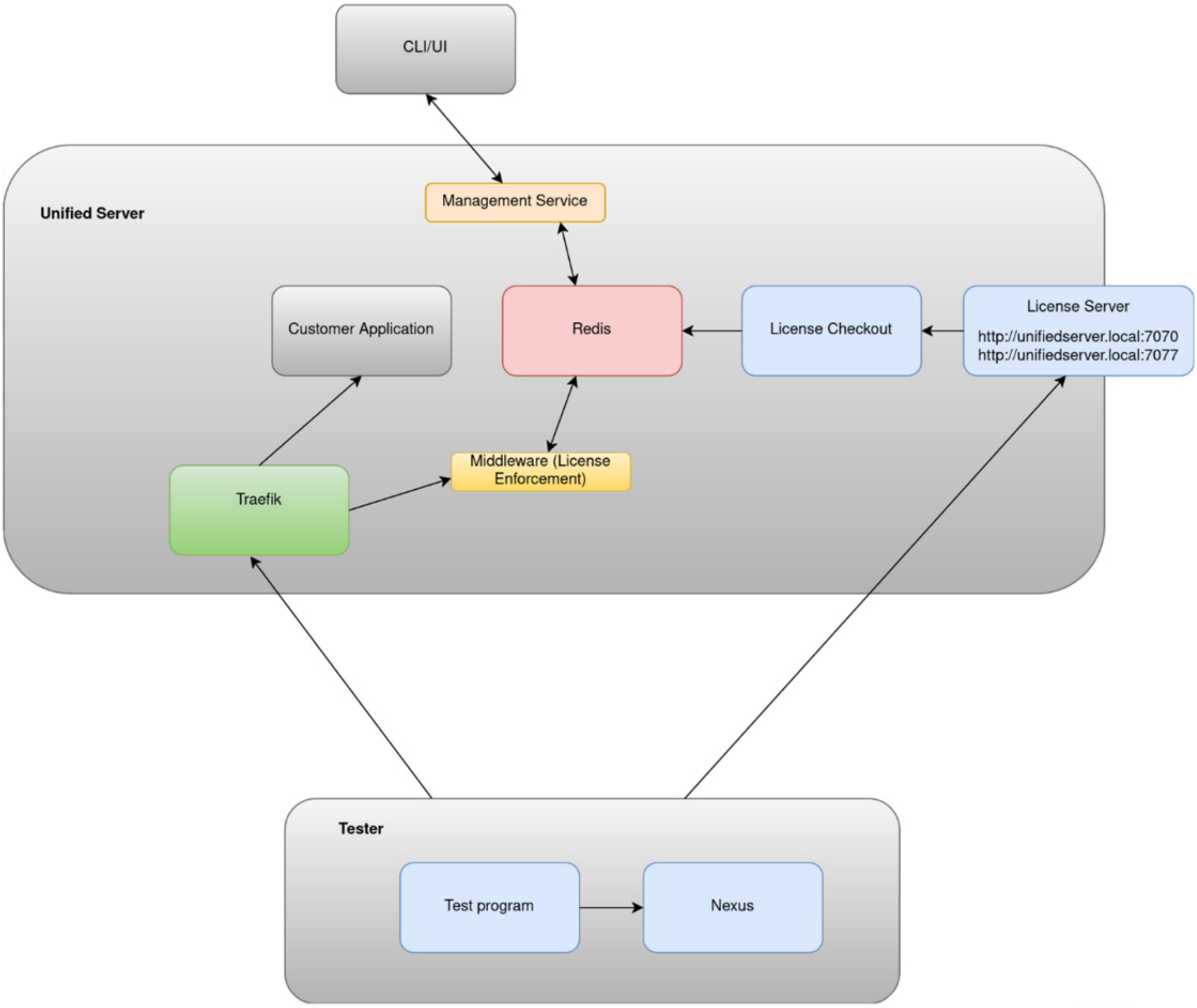


Figure 5-1. Licensing Diagram

5.3 Monitoring

The monitoring feature of the ACS Unified Server oversees registered test cells (ACS Edge Server and Host Workstation) for measuring efficiency of data transfers and applications, as well as debugging any potential issues. Status can be monitored through the Grafana web interface, and application logs can be monitored through cloud storages. Refer to Monitoring Dashboards for details on navigating Grafana and utilizing its dashboards. The application logs, results, and events are produced through the Advantest Logger SDK.

Data collected for monitoring and analysis are divided into four categories:

Table 5-1. Monitoring Data Categories

Data	Description
Logs	**Application Logs** Verbose information on what is happening in the application for tracking and debugging purposes. **System Logs** Verbose information on what is happening in the system for tracking and debugging purposes.
Application Results	Output generated from performing an operation or action within the customer's application.
System Metrics	Raw data exported from an application for quantitative measurements, such as the number of images downloaded.
Events	**Application Events** Custom events generated by the application. **System Events** Records of changes made on the system, such as user interactions, scheduled actions, and failures.

The monitoring data will be periodically uploaded to configured destinations. The Unified Server supports multiple destinations, including:

- FTP Server
- SFTP Server
- Azure File Storage
- Azure Blob Storage
- AWS S3 bucket
- Google Cloud Storage

5.3.1 Logs

Logs are divided into two types: application logs and System (RTDI) logs. The attributes differ depending on the log type.

Table 5-2. Application Log Attributes

Field	Description
testerId	The Host Workstation hostname.
edgeServerId	The Edge Server serial number.
applicationId	The application name.
timestamp	The recorded timestamp.
level	The criticality of the log, such as INFO, ERROR, etc.
message	The log message.

Following is an example of an application log.

```
{“testerId":"v93k-1", “edgeServerId": "MXQ02006VM", " applicationId ": "DPAT",
“timestamp": "1689878001“, "level":"INFO“, " message “: “ 2023-09-13T16:55:41Z INFO
application started"}
```

Table 5-3.   System Log Attributes

Field	Description
ServerType	The source of the log, such as Edge Server, Unified Server, or Nexus.
ServerId	The identification of the server.
service	The service name.
instance	The instance or container name.
timestamp	The recorded timestamp.
level	The criticality of the log, such as INFO, ERROR, etc.
message	The log message.
labels	Extra labels for additional aggregation. This attribute is optional.

Following are examples of system logs.

ACS Edge Server log:

```
{"ServerType": "Edge Server","ServerId":"v93k-1:MXQ02006VM", "service": "traefik",
"instance":"traefik", “timestamp": "1689878001“ , "level":"INFO“, "message “:
“time="2023-09-13T16:55:41Z" level=info msg="Configuration loaded from flags."}
```

ACS Unified Server log:

```
{"ServerType": "Unified Server","ServerId":"MXQ02006VM", "service": "log-consumer",
"instance":"log-consumer-1", “timestamp": "1689878001“,  "level":"INFO“, " message
“: “time="2023-09-13T16:55:41Z" level=info msg="Configuration loaded from flags."}
```

ACS Nexus log:

```
{"ServerType": "Nexus","ServerId":"v93k-1", "service": "Nexus", "instance":"",
“timestamp": "1689878001“,  "level":"INFO“ , " message “: “time="2023-09-
13T16:55:41Z" level=info msg="Configuration loaded from flags."}
```

5.3.2 Application Results

Table 5-4. Application Results Attributes

Field	Description
testerId	The Host Workstation hostname.
edgeServerId	The Edge Server serial number.
applicationId	The application name.
timestamp	The recorded timestamp.
key	The name of the result
value	The application result value.

Following is an example of application results.

```
{"testerId":"v93k-1", "edgeId":"MXQ02006VM", " applicationId ": "DPAT","timestamp":
"1689878001", "key": "new_upper_limit", "value": 10.0}
```

5.3.3 System Metrics

Table 5-5. Metrics Attributes

Field	Description
metric_name	The name of the metric
custom_labels	Extra labels for additional aggregation. This attribute is optional.
ServerType	The server type, such as Edge Server, Unified Server, or Nexus.
ServerId	The identification of the server.
value	the metric value.
timestamp	the recorded timestamp.

Following are examples of system metrics.

ACS Edge Server metrics:

```
number_of_images_downloaded{image_url="https://registry.advantest.com/image/version",
ServerType="Edge Server", ServerId="v93k-1:MXQ02006VM"} 149 1696463044000
```

ACS Unified Server metrics:

```
number_of_images_downloaded{image_url="https://registry.advantest.com/image/version",
ServerType="Unified Server", ServerId="MXQ02006VM"} 149 1696463044000
```

ACS Nexus metrics:

```
number_of_images_downloaded{image_url="https://registry.advantest.com/image/version",
ServerType="Nexus", ServerId="v93k-1"} 149 1696463044000
```

5.3.4 Events

Events are divided into two types: Application events and System (RTDI services) events. The attributes differ depending on the event type.

Table 5-6. Application Event Attributes

Field	Description
testerId	The source of the event, such as Edge Server, Unified Server, or Nexus.
edgeServerId	The identification of the server.
applicationId	The service name.
timestamp	The recorded timestamp.
eventType	The event type, such as INFO, DEBUG, or ERROR.
eventName	The event name.
message	The message of the event.
labels	Extra labels for additional aggregation. This attribute is optional.

Following is an example of an application event.

```
{"testerId":"v93k-1", "edgeServerId": "MXQ02006VM", "applicationId": "container_hub",
 "timestamp": "1689878179",
"eventType": "INFO", "eventName": "image_download", "message":"Download image
 successfully",labels:{"image_url": "https://registry.advantest.com/image/version"}}
```

Table 5-7. System Services Event Attributes

Field	Description
ServerType	The source of the event, such as Edge Server, Unified Server, or Nexus.
ServerId	The identification of the server.
timestamp	The recorded timestamp.
Level	The event type, such as INFO, DEBUG, or ERROR.
eventName	The event name.
message	The message of the event.
labels	Extra labels for additional aggregation. This attribute is optional.

Following are examples of system events.

ACS Edge Server event:

```
 {"ServerType": "Edge Server", "ServerId":"v93k-1:MXQ02006VM", "timestamp":
"1689878179",
"level": "INFO", "eventName": "imageDownloaded", "message":"Download image
successfully", labels:{"image_url": "https://registry.advantest.com/image/version"}}
```

ACS Unified Server event:

```
{"ServerType": "Unified Server", "ServerId":"MXQ02006VM", "timestamp": "1689878179",
"level": "INFO", "eventName": "image_downloaded", "message":"Download image
successfully",labels:{"image_url": "https://registry.advantest.com/image/version"}}
```

ACS Nexus event:

```
{"ServerType": "Nexus", "ServerId":"v93k-1", "timestamp": "1689878179",
"level": "INFO", "eventName": "image_downloaded", "message":"Download image
successfully",labels:{"image_url": "https://registry.advantest.com/image/version"}}
```

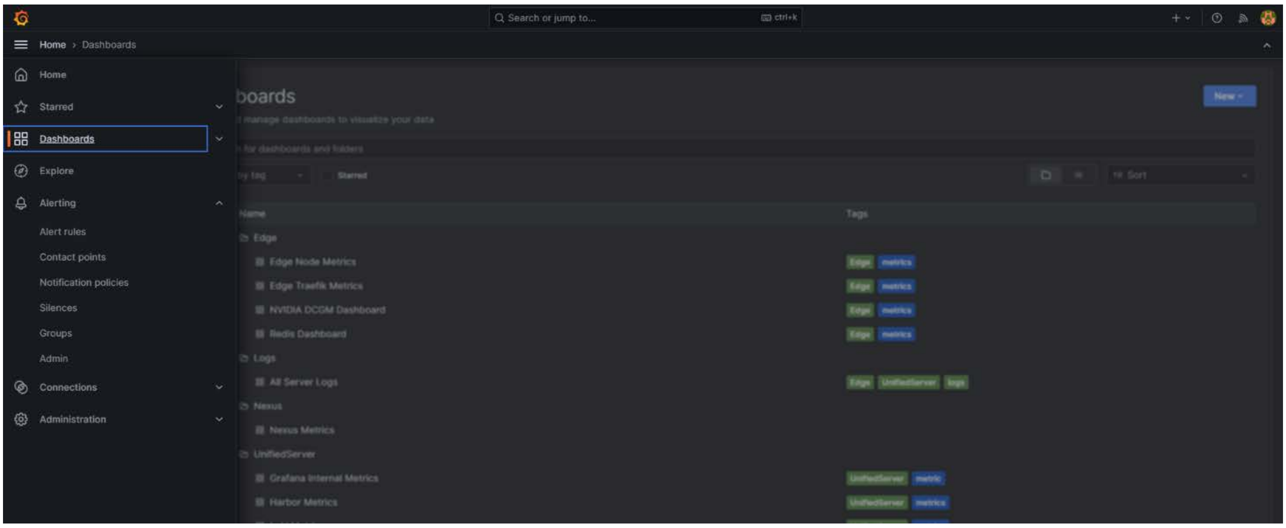
5.3.5 Monitoring Dashboards

Grafana is an analytics and interactive visualization web application that is used together with the ACS Unified Server's monitoring capabilities to measure activities around the metrics pipeline and log pipeline. Grafana can be accessed from any machine with a web browser that is on the same network as the ACS Unified Server.

To access Grafana, enter the following address in the url field of the web browser: https://unifiedserver.ui.local/grafana

5.3.5.1 Dashboards Inventory

To access dashboards, click on the left-side panel menu and select Dashboards to see the full list.



Name	Tags
Edge	
Edge Node Metrics	Edge metrics
Edge Traefik Metrics	Edge metrics
NVIDIA DCGM Dashboard	Edge metrics
Redis Dashboard	Edge metrics
Logs	
All Server Logs	Edge UnifiedServer logs
Nexus	
Nexus Metrics	
UnifiedServer	
Grafana Internal Metrics	UnifiedServer metric
Harbor Metrics	UnifiedServer metrics

Figure 5-2. Grafana Dashboard

5.3.5.2 Grafana Tutorials

For useful information on how to use Grafana, refer to the links below.

- Grafana Fundamentals: https://grafana.com/docs/grafana/latest/fundamentals/
- Querying in Explore section: https://grafana.com/docs/grafana/latest/explore/

5.4 Dynamic Certificates for MTLS

The Dynamic Certificates functionality introduces a service on the Unified Server that is used to dynamically generate server-side certificates during the deployment of the Unified Server. These certificates are used in Mutual TLS (MTLS) communication between the Unified Server and the connecting clients.

This feature also lets clients, like ACS Nexus, get their own client-side certificates from the Unified Server. During the installation of clients (ACS Nexus), a dynamically generated token is securely stored at the client side. This token serves as the authentication key to retrieving client-side certificates from the Unified Server.

Once the client has these certificates, they can use them for secure MTLS communication with the Unified Server.

Certificate Management

This service takes care of handling and automatically updating certificates on its own.

- The service is designed to automatically rotate certificates when they are about to expire, without requiring any manual input.
- For clients fetching certificates from the Unified Server, they can customize their queries based on the desired expiration time. This flexibility allows clients to generate short-lived certificates, enhancing the overall security of the system.
- You can keep track of details about which clients are retrieving certificates in the Grafana Dashboard logs.

5.5 Application Support

5.5.1 Container Hub Method

Multi-Tester Application Support is available by using harbor webhook service, where replication is done directly from the Container Hub. The administrative tasks, such as project creation and webhook configuration, are performed by the Advantest service team.

NOTES: Ensure that the project name and the repository name do not contain '`/`'. Images in Harbor are formatted as `<project>/<repo>:tag`. If a '`/`' is included in the repository name, Harbor will mistakenly interpret it as indicating another repository, causing it to fail to locate the image. For example, if a repository is named '`adv-dpat/adv_dpat`', Harbor will incorrectly assume there is a folder called '`adv-dpat`' and another called '`adv_dpat`', leading to the image not being found.

Also ensure that there are no artifacts without tags when replicating an image. If an artifact lacks a tag, the Harbor API will not be able to locate it.

5.5.2 SFTP Method

The Unified Server can run multiple applications. Multiple testers can be configured to use the application running on the Unified Server. The application lifecycle can be controlled (e.g., start/stop application) using an application descriptor or via CLI.

The Unified Server can download application images and application descriptors periodically from an SFTP Server. Based on information provided in the application descriptor, the Unified Server will start the applications. Below are step-by-step instructions for completing this process.

1. Using the Unified Server Management CLI (usm-cli), configure the Unified Server to communicate with a SFTP server for application replication dedicated for 1:N mode (refer to section 4.8.3 of the *ACS RTDI Installation Guide* for a detailed procedure).

2. Upload application (docker image archive file) and the corresponding application descriptor to the SFTP server using the procedure provided below (steps a – e). Prior to uploading the files, take note of the following:

   - Supported formats include:

Image	.tgz .tar.gz
Application Descriptor	.json

   - All image files must be moved to the target SFTP path before moving the application descriptor files.
   - The application descriptor file name must be the same as the name of the application descriptor.
   - The name of the image needs to be in the format `<project_name>/<image_name>:<image_tag>`.
   - There is no restriction on the filename of the image file.

Step a: Save the image to .tar.gz using the following command:

```
sudo docker save <project_name>/<image_name>:<image_tag> | gzip > <filename>.tar.gz
```

Step b: Use the following command to verify that the image can be loaded by docker and to check for possible image corruption.

```
sudo docker load -i <image_tar.gz_filename>
```

If the image is valid, output will indicate "Loaded image: project_name/image_name:tag" at the end.

If there is an error, re-download the image (repeat step a) and try again.

Step c: Log in to SFTP.

```
sftp  <SFTP_USER>@<SFTP_HOST>
```

Step d: Upload image (.tar.gz) and application descriptor (.json) to the root directory of the SFTP server.

NOTE: To prevent replication from occurring while files are still being uploaded, it is recommended to first upload the files to a temporary location (e.g. root), then move the files to the target folder. The permission to both the temporary location and the target folder is needed.

```
put <path to the image file or the application descriptor file on the work station>
```

Step e: Rename and move files to the correct path on the SFTP server. Ensure that the image files (.tar.gz) are moved first, and then its corresponding application descriptor files (.json)

```
rename <File_name> <SFTP_SERVER_PATH as configured from step 1 above>/<File_name>
```

NOTE: If there is a need to upload the files to two or more Unified Servers, repeat steps d and e, ensuring that `<SFTP SERVER PATH>` properly reflects the target Unified Server on step e. For example, if there are two Unified Servers to be configured, where the first Unified Server is configured to replicate from SFTP Server path /abc, while the second Unified Server is configured to replicate from SFTP Server path /xyz, steps d and e must be done twice:

First: SFTP upload and move to /abc
Second: SFTP upload and move to /xyz

This way, the files will be replicated to both Unified Servers.

Notes about Application Descriptors

Requirements:

- **exposed_ports**
  Ports opened for customers to connect their application. The format should be a single string and the ports are divided with commas.
  - mTLS

    A single port without ":" (colon) will be used as a gateway for traefik to route customer requests to their containers. An example of the URL of such a request is provided below:

    -If there is only one mTLS port provided:

    https://unifiedserver.acs/applications/{container_name}-21122/ {customer's path}

    -If there are multiple mTLS ports provided:

    https://unifiedserver.acs/applications/{container_name}-{container-port}/{customer's path}

    NOTE: Static certifications are required.

- **mapped_ports**
  - non-mTLS

    Two port numbers with ":" (colon) will map the host's port to the container port directly (for example, "<host_port>:<container_port>"). The default protocol is TCP. However, protocols such as TCP or UDP can be added optionally (for example, 8000:8000/tcp). Additional examples are as follows:

    Example 1: "8002:8002": http://unifiedserver.local:8001/{customer's path}
    Example 2: "8000:8000/tcp": http://unifiedserver.local:8000/{customer's path}

Environment:

- **OPERATION**
  Allows the customer application to be started, stopped, or ignored by monitoring the operation inside the application descriptor.
  - start: Removes older container and restarts the container
  - stop: Stops and removes the container and volumes
  - ignore: Updates the application descriptor without applying any operation on the container or volume

Example of Application Descriptor content

```
{
    "version": "1",
    "name": "testing",
    "selector": {
        "device_name": "testing",
        "product_family": "*",
        "test_program_name": "*",
        "test_program_revision": "*"
    },
    "unified": {
        "containers": [{
            "name": "testing",
            "image": "test/app-metric:latest",
            "requirements": {
                "gpu": false
                "exposed_ports":[8100, 8088],
                "mapped_ports":[ "9002:9002/tcp"],
            },
            "metrics": {
                "port": 8001,
                "path": "/metrics"
            },
            "volume_attachments": [
                "volume1:rw", "volume2"
            ],
            "environment": {

            }
        }],
        "volumes": [{
            "name": "volume1:rw",
            "image": "test/app-metric:latest"
        }, {
            "name": "volume2",
            "image": "test/app-metric:latest"
        }]
    }
}
```

The acs-application tool can be used to validate the application descriptor.

3. When the container is created/stopped, there will be customer events sent to the monitoring upload destinations as configured during Unified Server installation (refer to section 4.8.2.2 of the *ACS RTDI Installation Guide* for more details). The filename will be in the format **<unifiedserver_id>_<timestamp>.event**. See below for an example of file content:

```
{ "level": "info", "message": "[POST containers/create_start] Response detail:
Container jason-testing-metric created with metric sidecar and started with metric
sidecar", "ServerType": "Unified Server", "ServerId": "SERVER_ID'", "service":
"edge_resilience_serverapi", "instance": "edge_resilience_serverapi", "timestamp":
"event_timestamp" }
```

4. Optionally, use the usm-cli tool to control applications as needed:

   a. From usm-cli main menu, navigate to the **Applications** option.
   b. Choose between the following available options:
      - Start Application
        Starts an application specific to the user-entered application descriptor name.
      - Stop Application
        Stops an application specific to the user-entered application descriptor name.
      - Remove Volume
        Removes a volume specific to the user-entered application descriptor name.
      - List Applications
        Lists running containers with container name, created date, and started date.

5. Optionally, a liveness probe check can be done when a customer app specifies health check in a Dockerfile. An internal service (app-monitor) on the Unified Server will automatically restart the app container if the container health status becomes Unhealthy. To implement health check, see the highlighted entries below:

```
FROM python:3.9-slim

WORKDIR /app

RUN apt-get update && apt-get install -y curl && rm -rf /var/lib/apt/lists/*

COPY . /app

RUN pip install --no-cache-dir prometheus_client
HEALTHCHECK --interval=5s --timeout=5s --start-period=5s --retries=1 \
    CMD curl --fail http://localhost:8001/healthcheck || exit 1
CMD[“./start.sh”]
```

RUN apt-get update && apt-get install -y curl

Curl is a common tool for HTTP-based service, but you can use other methods depending on the nature of your application.

```
HEALTHCHECK --interval=5s --timeout=5s --start-period=5s --retries=1 \
CMD curl --fail http://localhost:8001/healthcheck || exit 1
```

Following is additional information regarding each HEALTHCHECK config setting:

- Interval: Docker will run the healthcheck command every 5 seconds.
- Timeout: If the command takes longer than 5 seconds, the health check is considered to have failed.
- Start-period: This defines the start period for the container to stabilize before starting the health checks. During this period, any health check failures are considered to be successes.
- Retries: This sets the number of consecutive failures needed to consider the container as unhealthy.
- CMD curl –fail http://localhost:8001/healthcheck || exit 1: This is the command that is executed every time the health check runs. If your application returns 4xx or 5xx HTTP status code, it causes curl to return a non-zero exit code which indicates failure.

5.5.3 1:N Application Descriptors

Application Descriptor fields specific to the Unified Server were added under the new **`unified`** field, replacing the deprecated **1N** environment variable in the `edge` field.

```
{
  "version": "1",
  "name": "app",
  "unified": {
    "containers": [
      {
        "name": "app-name",
        "image": "registry.advantest.com/project/image-name:tag”,
        “readiness_probe”: {
          “failure_threshold”: 3,
          “initial_delay_seconds”: 30,
          “period_seconds”: 10,
          “success_threshold”: 1,
          “timeout_seconds”: 10,
          “http_get”: {
            “path”: “/ready”,
            “port”: “80”
          }
        },
        “liveness_probe”: {
          “failure_threshold”: 3,
          “initial_delay_seconds”: 30,
          “period_seconds”: 10,
          “success_threshold”: 1,
          “timeout_seconds”: 10,
          “exec”: {
            “command”: [
              “/bin/sh”,
              “-c”,
              “touch /tmp/file; cat /tmp/file”
            ]
          }
        },
        “lifecycle”: {
          “pre_stop”: {
            “httpGet”: {
              “path”: “/pre_stop”,
              “port”: 80
            }
          }
        },
        “termination_grace_period”: 30
        “environment”: {
           “STREAM_ENABLED”: “true”
        },
      }
    ]
  }
}
```

Readiness and Liveness Probe

A readiness probe is used to determine if a container is ready to serve traffic. If the readiness probe fails, the pod is considered not ready, and Kubernetes will not send any traffic to it until it passes.

A liveness probe in Kubernetes is used to determine if a container is still alive and running properly. If the liveness probe fails, Kubernetes will assume the container has failed and will restart it to recover.

Both probes have the following configuration for the type of probe:

- `http_get`
    - path: The path to access on the HTTP server.
    - port: The name or number of the port to access on the container. The name must be an IANA_SVC_NAME. The valid number range is 1 to 65535.
- exec
    - command: The command line to execute inside the container. For example, [cat, /tmp/health]
- failure_threshold: The minimum consecutive failures for the probe to be considered failed after having succeeded.
- initial_delay_seconds: The number of seconds after the container has started before liveness probes are initiated.
- period_seconds: How often (in seconds) to perform the probe.
- success_threshold: The minimum consecutive successes for the probe to be considered successful after having failed.
- timeout_seconds: The number of seconds after which the probe times out.

Shutdown Hook

The Shutdown Hook feature enables users to define custom behavior immediately before an application is terminated to guarantee a graceful shutdown.

Applications may be terminated due to user request, liveness or startup probe failure, resource contention, or other Kubernetes related events. By default, when terminating a container, Kubernetes sends a SIGTERM to the main process and waits for termination_grace_period configuration. After this time, if the application still has not terminated, Kubernetes sends a SIGKILL to all remaining processes.

When the Shutdown Hook is defined via the lifecycle.pre_stop option, Kubernetes will execute this action before sending the SIGTERM. This allows any custom behavior to be implemented, such as cleanup actions. In parallel, the termination_grace_period countdown will start. This means that the time to finish the pre_stop hook plus the SIGTERM should be less than the grace period. Otherwise, the application will be subject to a non-graceful shutdown.

httpGet and exec are the two pre_stop actions that are supported. Both actions are executed inside the container itself.

- httpGet
    - port: The port number to reach the process inside the container. Valid range is 1 to 65535.
    - path: The HTTP path to reach the API inside the container.
    - Kubernetes will perform a GET request to `http://localhost:{httpGet.port}/{httpGet.path}` and expect a `200 OK` status as response.
- exec
    - A list of strings representing a shell command to be executed, from which Kubernetes expects a success code (0).

Data Streaming

The Data Streaming feature enables user applications to stream data from the Unified Server through OneAPI. To enable this feature, add **`STREAM_ENABLED: true`** to the Environment variables dictionary of your Application Descriptor. With Data Streaming enabled, your application can integrate OneAPI to stream data from testers via the Unified Server for real-time data retrieval.

5.6 Unified Server Application Testing

Customers can securely communicate with their applications running on the Unified Server using the Customer App API.

For this purpose, Advantest provides both a Web Interface (via Container Hub) and a Command Line Interface (CLI) tool.

5.6.1 Customer API via CLI

See below for details on how to use the Customer App API via CLI.

Usage Guidelines

- Allowed Special Characters: `{ +, !, @, =, -, _, * }`
- Disallowed: All other special characters

Supported Commands

The CLI currently supports the following commands:

- list-unifiedservers
- list-applications
- test

Command	`list-unifiedservers`
Description	Retrieves the list of Unified Servers hosting the customer's applications
Usage Format	`apptest-cli list-unifiedservers -o <organization_name> --username <containerhub_username> --password <containerhub_password>`
Example	`apptest-cli list-unifiedservers -o acsqa --username customer1 --password 123456`
Output	`["unifiedserver1", "unifiedserver2"]`

Command	`list-applications`
Description	Fetches applications running on a specified Unified Server. Returns app name, exposed ports, and status.
Usage Format	`apptest-cli list-applications --unifiedserver <unifiedserver_name> -o <organization_name> --username <containerhub_username> --password <containerhub_password>`
Example	`apptest-cli list-applications --unifiedserver unifiedserver1 -o acsqa --username customer1 --password 123456` **NOTE:** This uses acs-application to list the descriptor, then runs the App Monitor API to identify the matching Unified Server.
Output	`[{"Application Name": "shutdownhook031","Exposed Ports": [8080],"Application Status": "RUNNING”},{"Application Name": "demo2", "Exposed Ports": [80],"Application Status": "RUNNING"}]`

Command	`test`
Description	Calls an API endpoint of a customer application hosted on a Unified Server, and returns the actual response.
Usage Format	`apptest-cli test -X <request_type> --url https://<unifiedserver>/<app_name>-<app_port>/<api_path> -o <organization> --username <containerhub_username> --password <containerhub_password> -d <data>`
Example	`apptest-cli test -X GET --url https://unifiedserver1/app3-8000/health -o acsqa --username customer1 --password 123456` **NOTE:** This call logs the user executing the API.
Output	`{\"status_code\": 200, \"response_time_ms\": 24, \"response\": {\"message\":\"health\"}}`

NOTE: Be aware of OS-specific syntax differences. There are differences between Linux and Windows CLI usage, especially regarding quoting and escaping characters

Example CLI Command for Linux

```
apptest-cli test -X POST --url https://unifiedserver1/application1-port/endpoint --data '{ "key": "value" }' --organization org_name --username username_containerhub --password docker_secret_containerhub -H "Accept: application/json" -H "Content-Type: application/json"
```

Example CLI Command for Windows

```
apptest-cli test -X POST --url https://unifiedserver1/application1-port/endpoint --data '{"name": "value"}' --organization org_name --username username_containerhub --password docker_secret_containerhub -H '"Accept: application/json"' -H '"Content-Type: application/json"'
```

5.7 File Synchronization

File synchronization is a feature that enables ACS RTDI users to seamlessly transfer files from their local machines to Unified Servers in OSAT facilities, allowing applications deployed on these servers to access and utilize the files. The primary purpose of this feature is to support applications running on unified servers that require regular data updates in production. File synchronization streamlines the workflow and allows for a more efficient method for transferring files to the unified servers, avoiding the overhead and delays associated with file transfer via Docker image.

5.7.1 Overview

The file synchronization diagram below outlines the flow of data files from the user's local environment to the Unified Server where the customer application operates.

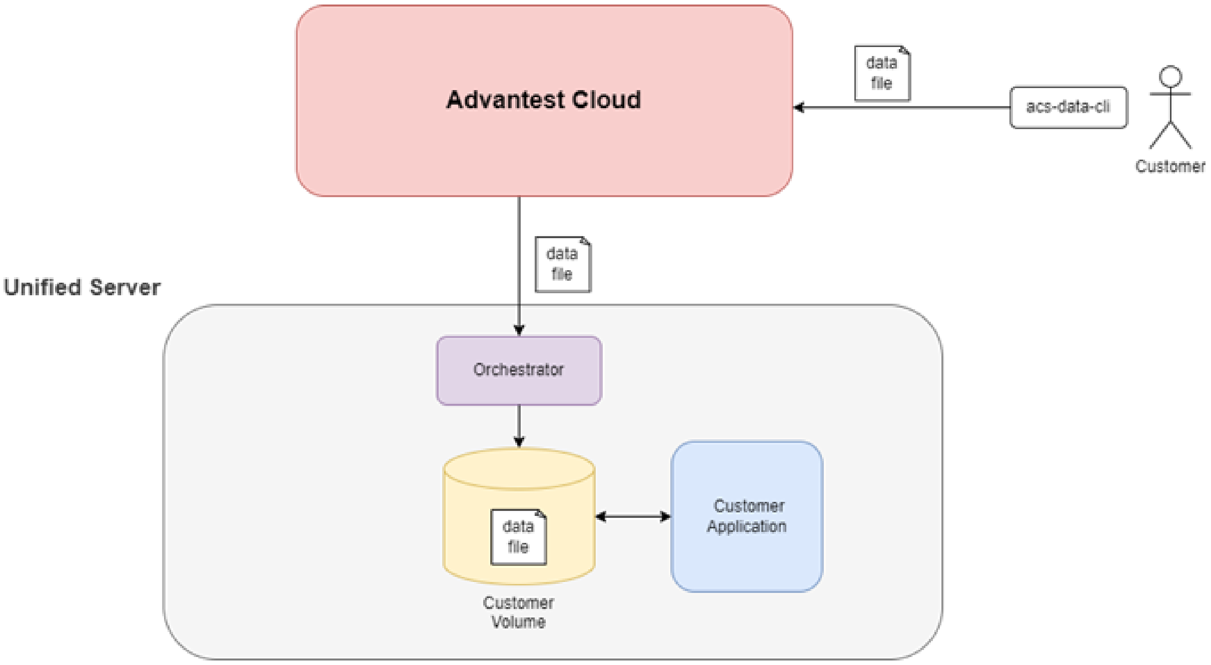


Figure 5-3. File Synchronization Diagram

Each component in the above diagram is listed and described in the table below.

Table 5-8. File Synchronization Components

Component	Description	Function
acsdata-cli (Local Machine Tool)	The acsdata-cli is a command-line tool provided to users for managing data file uploads from their local machines.	Using this tool, users can initiate the transfer of specific data files that need to be synchronized with the application on the Unified Server.
Advantest Cloud	The Advantest Cloud acts as the intermediary that facilitates the secure transfer of data files from the user's local machine to the Unified Server.	When a data file is uploaded via the acs-data-cli, it is first stored temporarily on the Advantest Cloud before being transferred to the Unified Server. This setup ensures secure and reliable data handling.

Component	Description	Function
Orchestrator (File Transfer Controller)	The Orchestrator on the Unified Server manages the final step of the data file synchronization process.	Once the data file reaches the Unified Server, the Orchestrator takes over, moving the file into the designated storage location (referred to as "Customer Volume") on the Unified Server. This ensures the data is accessible to the applications that need it.
Customer Volume (Storage for Data Files)	The Customer Volume is a dedicated storage location on the Unified Server where data files are stored for application use.	The Orchestrator deposits the data file into this volume, where it remains accessible to the customer application. This setup ensures the application always has the latest version of the data without requiring a full redeployment.
Customer Application	This is the end-user application running on the Unified Server, which utilizes the synchronized data files for its operations.	The Customer Application accesses data files directly from the Customer Volume. With this setup, the application can seamlessly use updated data files without the need for extensive server or application changes.

Based on the components listed in Table 5-8 above, the following is a summarization of how the file synchronization process works.

1. **Initiate File Upload:** The user uploads a data file from their local machine using the acsdata-cli tool.

2. **Transfer to Advantest Cloud:** The file is temporarily stored in the Advantest Cloud for secure handling.

3. **Orchestrator Transfer:** The Orchestrator on the Unified Server retrieves the file from the cloud and places it into the Customer Volume.

4. **Application Access:** The Customer Application can now directly access and utilize the updated data file from the Customer Volume.

5.7.2 acsdata-cli Command Line Tool Requirement and Path

System Requirement

- Linux on x86_64

The acsdata-cli command line tool is distributed as a single, self-contained executable binary file. It is installed on the Linux environment which is used for generating and uploading files. Request the latest version from your Advantest technical contact and copy it to a location in your PATH.

```
sudo cp ./acsdata-cli /usr/local/bin/ acsdata-cli
```

5.7.3 acsdata-cli Commands

The acsdata-cli tool is a command-line interface (CLI) for managing data files on Advantest's unified servers. This tool enables users to upload, list, and delete files stored on unified server volumes, facilitating seamless data management and synchronization with customer applications.

The acsdata-cli supports three main commands:

- **Upload**
  Uploads a data file from the local machine to a specified volume on the unified server.
- **List**
  Lists files available on a specific server volume or provides details about a particular file.
- **Delete**
  Deletes specified files from a unified server volume.

Upload Command

```
UNIFIED_SERVER --volume-name VOLUME_NAME --file-path FILE_PATH [--ttl TTL] [--overwrite]
[--volume-size VOLUME_SIZE] [--verbose]
```

<table>
<tr><td><b>Description</b></td><td>The upload command allows you to transfer a data file from your local machine to a designated volume on the Unified Server.</td></tr>
<tr><td><b>Parameters</b></td><td>Required:
<table>
<tr><td><code>--username USERNAME</code></td><td>Your Container Hub username.</td></tr>
<tr><td><code>--password PASSWORD</code></td><td>Your Container Hub password.</td></tr>
<tr><td><code>--unified-server UNIFIED_SERVER</code></td><td>The name of the Unified Server.</td></tr>
<tr><td><code>--volume-name VOLUME_NAME</code></td><td>The name of the volume on the Unified Server.</td></tr>
<tr><td><code>--file-path FILE_PATH</code></td><td>Path to the data file on your local machine.</td></tr>
</table>
Optional:
<table>
<tr><td><code>--ttl TTL</code></td><td>Time-to-Live in days (default is 15).</td></tr>
<tr><td><code>--overwrite</code></td><td>Overwrites the file if it already exists in the specified volume.</td></tr>
<tr><td><code>--volume-size VOLUME_SIZE</code></td><td>Volume size in GB (default is 10).</td></tr>
<tr><td><code>--verbose</code></td><td>Enables verbose output, providing more detailed feedback.</td></tr>
</table>
</td></tr>
<tr><td><b>Example</b></td><td><code>acsdata-cli upload --username johndoe –-password password123 --
unified-server server1 --volume-name vol1 --file-path
/path/to/file.json --ttl 30 --overwrite --verbose</code></td></tr>
</table>

List Command

`acsdata-cli list --username USERNAME --password PASSWORD [--unified-server UNIFIED_SERVER] [--volume-name VOLUME_NAME] [--file-name FILE_NAME] [--verbose]`

<table>
<tr><td><b>Description</b></td><td>The list command retrieves information about files stored on the Unified Server volumes. This command can list all files in a volume or details about a specific file.</td></tr>
<tr><td><b>Parameters</b></td><td>Required:
<table>
<tr><td><code>--username USERNAME</code></td><td>Your Container Hub username.</td></tr>
<tr><td><code>--password PASSWORD</code></td><td>Your Container Hub password.</td></tr>
</table>
Optional:
<table>
<tr><td><code>--unified-server UNIFIED_SERVER</code></td><td>The name of the Unified Server.</td></tr>
<tr><td><code>--volume-name VOLUME_NAME</code></td><td>The name of the volume on the Unified Server.</td></tr>
<tr><td><code>--file-name FILE_NAME</code></td><td>The specific file name to retrieve details for.</td></tr>
<tr><td><code>--verbose</code></td><td>Enables verbose output.</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td><code>acsdata-cli list --username johndoe –-password password123 --unified-server server1 --volume-name vol1 --verbose</code></td></tr>
</table>

Delete Command

```
acsdata-cli list --username USERNAME --password PASSWORD [--unified-server
UNIFIED_SERVER] [--volume-name VOLUME_NAME] [--file-name FILE_NAME] [--verbose]
```

<table>
<tr><td><b>Description</b></td><td>The delete command allows you to remove one or more files from a specified volume on the Unified Server.</td></tr>
<tr><td><b>Parameters</b></td><td>Required:
<table>
<tr><td><code>--username USERNAME</code></td><td>Your Container Hub username.</td></tr>
<tr><td><code>--password PASSWORD</code></td><td>Your Container Hub password.</td></tr>
<tr><td><code>--unified-server UNIFIED_SERVER</code></td><td>The name of the Unified Server.</td></tr>
<tr><td><code>--volume-name VOLUME_NAME</code></td><td>The name of the volume on the Unified Server.</td></tr>
<tr><td><code>--file-name FILE_NAME</code></td><td>The name of the file to delete.</td></tr>
</table>
Optional:
<table>
<tr><td><code>--verbose</code></td><td>Enables verbose output.</td></tr>
</table>
</td></tr>
<tr><td><b>Example</b></td><td><code>acsdata-cli delete --username johndoe --unified-server server1 --volume-name vol1 --file-name file.json --verbose</code></td></tr>
</table>

Additional Notes:

The following notes apply to all commands.

- **Volume Name Requirements**
  Volume names can only contain lowercase letters, numbers, periods (.), and hyphens (-). Volume names cannot start or end with periods or hyphens.
- **Supported File Types**
  Only JSON and Parquet files are accepted. If a file with a different MIME type is uploaded, an error will be returned.
- **Retry Mechanism**
  For uploads, the tool retries failed attempts up to 3 times, with a 30-second delay between attempts.

5.8 Redis Memory Store SDK for Cross-Cluster Replication

The MemoryStore SDK provides a simple interface for interacting with a Redis key-value store. This SDK allows you to set, get, delete, and check the existence of keys for Redis datatypes – like Redis Strings, Redis Hash, and Redis Set – in a Redis database.

This SDK enables clients to replicate their data in real-time stored in a Redis Cluster to other Unified Servers. It includes a boolean parameter (`is_global`) that is used across various functions of this SDK to determine whether the data is global—requiring replication to other Unified Servers—or local, in which case it remains confined to the current server. The default value of this parameter is set to 'True' implying that all the data modifications done in Redis Cluster using this SDK will be replicated across other connected Unified Servers in real-time.

5.8.1 Using the SDK in your Application

To obtain the SDK, request the latest MemoryStore SDK from your Advantest technical contact.

Installation prerequisites:

- Python 3.7 or later is running
- redis-py library (version 5.0.0 or later) is installed using one of the following methods:
    - `pip install redis`
    - `pip install -r requirements.txt`
      This installs the dependency via the bundled `requirements.txt` (which specifies `redis>=5.0.0`)

Integrate the SDK:

Place the SDK script (e.g. memory_store.py) alongside your application code.

Import the main class in your Python module:

```
from memory_store import MemoryStore
```

Explore the API:

See the sections below for detailed examples of each SDK method.

5.8.2 Configuring Replication

Replications can be configured as either unidirectional or bidirectional. To set up a Unified Server to replicate data from a source Unified Server, use the UnifiedServer config CLI or UI tool. For bidirectional replication, repeat the same configuration steps on the source Unified Server, treating the current server as its source. For assistance, please contact your Advantest technical representative.

5.8.3 Resolving Conflicts from Concurrent Redis Key Updates Across Clusters

When multiple Unified Servers update the same Redis key simultaneously during data replication, conflicts are resolved using a last-writer-wins approach. The value from the most recent successful write is propagated to—and preserved by—all Redis clusters managed by the participating Unified Servers, ensuring a consistent state across sites.

5.8.4 Impact of Network Disruptions between Cross-Site UnifiedServer Clusters

During cross-site outages, the originating UnifiedServer buffers unsynchronized Redis key changes for up to five days or 5 GB of accumulated data, whichever limit is reached first. When connectivity is restored, all buffered updates are automatically replicated to the Redis clusters of the other UnifiedServers, ensuring data consistency across sites.

5.8.5 Allowed Redis Value Size

When performing insert or update operations with MemoryStore class—namely the set, hset, and sadd methods—each value stored for a key must be no larger than 25 MiB.

5.8.6 MemoryStore Class Methods

The MemoryStore class provides an interface for interacting with a Redis cluster, offering basic key-value operations such as setting, retrieving, deleting keys for Redis datatypes (such as Redis Strings, Redis Hash and Redis Set), and managing Redis configuration (such as memory limits).

`keys(self, pattern: str = "*", is_global: bool = True) -> list`

<table>
<tr><td><b>Description</b></td><td>Returns a list of keys that match the given pattern.
<b>NOTE:</b> This method scans all the nodes of the Redis Cluster to get the keys based on the given pattern. This is an expansive operation and might slow down other operations performed concurrently on the RedisCluster. Not recommended to be heavily used in Production settings.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td><code>pattern</code></td><td>(str)
The pattern to match keys against.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is 'True']
If set to 'True', the keys are returned from 'global:' namespace – the keys which are replicated to/from other UnifiedServers.
If 'False', the keys are returned from 'local:' namespace – the keys local to this UnifiedServer.</td></tr>
</table></td></tr>
<tr><td><b>Returns</b></td><td>(list)
A list of keys that match the pattern. Returns an empty list otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>all_keys = store.keys()</code></td></tr>
</table>

`type(self, key: str, is_global: bool = True) -> str`

Description	Determines the Redis data type of the given Redis key, allowing the client to call the appropriate commands—string (SET, GET), hash (HSET, HGET, HDEL, HGETALL), or set (SADD, SREM, SMEMBERS)—for that key.
Parameters	<table><tr><td>`key`</td><td>(str) The key to get the type for.</td></tr><tr><td>`is_global`</td><td>(bool) [Default is 'True'] If set to 'True', the keys from 'global:' namespace – the keys replicated to/from other UnifiedServers are used. If 'False', the keys from 'local:' namespace – the keys local to this UnifiedServer are used.</td></tr></table>
Returns	(list) A list of keys that match the pattern. Returns an empty list otherwise.
Example	`all_keys = store.keys()`

`set(self, key: str, value: str, expiration: Optional[int]=None, is_global: bool=True) -> bool`

Description	Sets a key-value pair in the Redis with an optional expiration time. **NOTE:** If 'is_global' is set as False, then this change is not replicated to other connected UnifiedServer Redis Clusters.
Parameters	<table><tr><td>`key`</td><td>(str) The key to store in Redis.</td></tr><tr><td>`value`</td><td>(str) The value associated with the key.</td></tr><tr><td>`expiration`</td><td>(int, optional) Time in seconds until the key expires. Default is None.</td></tr><tr><td>`is_global`</td><td>(bool) [Default is `True`] If set to `True`, the change is replicated globally to other connected UnifiedServers.</td></tr></table>
Returns	(bool) True if the key was set successfully. False otherwise.
Example	`is_added = store.set("example_key", "example_value")`

`get(self, key: str, is_global: bool = True) -> Optional[str]`

<table>
<tr><td><b>Description</b></td><td>Retrieves the value associated with the specified key.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The key to retrieve the value for.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the key is returned from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the keys are returned from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>Optional[str]
The value if the key exists, otherwise None.</td></tr>
<tr><td><b>Example</b></td><td><code>value = store.get("example_key")</code></td></tr>
</table>

`delete(self, key: str, is_global: bool = True) -> bool`

<table>
<tr><td><b>Description</b></td><td>Deletes the key-value pair for the given key.
<b>NOTE:</b> If `is_global` is set as False, then this change is not replicated to other connected UnifiedServer Redis Clusters.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The key to delete.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the key to be deleted is fetched from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the key to be deleted is fetched from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the key was deleted. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>is_deleted = store.delete("example_key")</code></td></tr>
</table>

exists(self, key: str, is_global: bool = True) -> bool

<table>
<tr><td><b>Description</b></td><td>Checks if a key exists in the Redis Cluster.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td>key</td><td>(str)
The key to check.</td></tr>
<tr><td>is_global</td><td>(bool) [Default is `True`]
If set to `True`, the key is checked from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the key is checked from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the key exists. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>exists = store.exists("example_key")</code></td></tr>
</table>

`hset(self, key: str, mapping: Dict[str, Any], expiration: Optional[int]=None, is_global: bool=True) -> bool`

<table>
<tr><td><b>Description</b></td><td>Inserts the supplied mapping— a dictionary of field-value pairs—into an existing Redis hash, or creates the hash if it doesn’t yet exist. Accepts an optional expiration to set an expiration time for the hash key.
If `is_global` is set as False, then this change is not replicated to other connected UnifiedServer Redis Clusters.
<b>NOTE:</b> Replication occurs at the individual field level. For example, if 5 field–value pairs are added to a hash, each field is evaluated independently under the last-writer-wins rule; the most recent update for every field is synchronized across all UnifiedServers.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The Redis Hash key to store in Redis.</td></tr>
<tr><td><code>mapping</code></td><td>(Dict[str, Any])
The dictionary of field-value pairs for the Redis Hash.</td></tr>
<tr><td><code>expiration</code></td><td>(int, optional)
Time in seconds until the key expires. Default is None.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the change is replicated globally to other connected UnifiedServers.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the hash mapping was set successfully. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><pre>is_added = store.set("example_hash_key",
                     {"field_1": "value_1",
                      "field_2": "value_2", ...,
                      "field_n": "value_n"})</pre></td></tr>
</table>

`hget(self, key: str, field: str, is_global: bool = True) -> Optional[str]`

<table>
<tr><td><b>Description</b></td><td>Retrieves the value of a single field from the specified Redis hash. If the hash (or field) does not exist, returns None.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The Redis hash key.</td></tr>
<tr><td><code>field</code></td><td>(str)
The field in Redis hash to retrieve the value for.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the key is returned from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the key is returned from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>Optional[str]
The value if the key exists, otherwise None.</td></tr>
<tr><td><b>Example</b></td><td><code>value = store.hget("example_hash_key", "field_1")</code></td></tr>
</table>

`hdel(self, key: str, fields: List[str], is_global: bool = True) -> bool`

<table>
<tr><td><b>Description</b></td><td>Deletes one or more fields from the specified Redis hash. Returns True if at least one field was removed, otherwise False.
If `is_global` is set as False, then this change is not replicated to other connected UnifiedServer Redis Clusters.
<b>NOTE:</b> Replication occurs at the individual field level. For example, if 5 fields are deleted from a hash, each field is evaluated independently under the last-writer-wins rule; the most recent operation for every field is synchronized across all UnifiedServers.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The Redis hash key.</td></tr>
<tr><td><code>fields</code></td><td>(List[str])
The list of fields to be deleted from the provided Redis hash key.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the keys are deleted from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the keys are deleted from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if at least one field was removed. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>is_deleted = store.hdel("example_hash_key",["field_1", ..., "field_n"])</code></td></tr>
</table>

`hgetall(self, key: str, is_global: bool = True) -> Dict[str, Any]`

<table>
<tr><td><b>Description</b></td><td>Fetches every field-value pair stored in the given Redis hash and returns them as a dictionary. If the hash is missing, an empty dictionary is returned.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td><code>key</code></td><td>(str)
The Redis hash key to retrieve the field-value pairs for.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the keys are returned from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the keys are returned from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table></td></tr>
<tr><td><b>Returns</b></td><td>Dict[str, Any]
The dictionary mapping of field-value pairs from Redis hash if the key exists, otherwise empty dictionary.</td></tr>
<tr><td><b>Example</b></td><td><code>field_value_mapping = store.hgetall("example_hash_key")</code></td></tr>
</table>

`hexists(self, key: str, field: str, is_global: bool = True) -> bool`

<table>
<tr><td><b>Description</b></td><td>Checks whether a specific field exists within the target Redis hash. Returns True if the field is present, otherwise False.</td></tr>
<tr><td><b>Parameters</b></td><td><table>
<tr><td><code>key</code></td><td>(str)
The Redis hash key.</td></tr>
<tr><td><code>field</code></td><td>(str)
The field to be checked within the Redis hash.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the key is checked from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the key is checked from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table></td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the key exists. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>exists = store.hexists("example_hash_key", "field_1")</code></td></tr>
</table>

`sadd(self, key: str, members: List[Any], expiration: Optional[int]=None, is_global: bool=True) -> bool`

<table>
<tr><td><b>Description</b></td><td>Adds the supplied list of members to a Redis set, creating the set if it doesn’t already exist. Accepts an optional expiration to set an expiration time for the set key.
If `is_global` is set as False, then this change is not replicated to other connected UnifiedServer Redis Clusters.
<b>NOTE:</b> For Redis sets, replication treats the entire set as a single unit rather than tracking individual members. Any change—whether adding or removing members—overwrites the previous version of the set under the last-writer-wins rule in every UnifiedServer, ensuring that all clusters maintain an identical, up-to-date copy.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The key to store in Redis.</td></tr>
<tr><td><code>members</code></td><td>(List[Any])
The list of members to be added to the Redis set.</td></tr>
<tr><td><code>expiration</code></td><td>(int, optional)
Time in seconds until the key expires. Default is None.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the change is replicated globally to other connected UnifiedServers.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the members were added to an existing or new Redis set successfully. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><pre>is_added = store.sadd("example_set_key", ["member_1", ...,
                                          "member_n"])</pre></td></tr>
</table>

srem(self, key: str, members: List[str], is_global: bool = True) -> bool

<table>
<tr><td><b>Description</b></td><td>Removes one or more members from the specified Redis set. Returns True if at least one member was deleted; otherwise, False.
If `is_global` is set as False, then this change is not replicated to other connected UnifiedServer Redis Clusters.
<b>NOTE:</b> For Redis sets, replication treats the entire set as a single unit rather than tracking individual members. Any change—whether adding or removing members—overwrites the previous version of the set under the last-writer-wins rule in every UnifiedServer, ensuring that all clusters maintain an identical, up-to-date copy.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The Redis set key.</td></tr>
<tr><td><code>members</code></td><td>(List[str])
The list of members to be deleted from the provided Redis set key.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the keys are deleted from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the keys are deleted from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if at least one member was removed. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>is_deleted = store.srem("example_set_key", ["member_1", ..,"member_n"])</code></td></tr>
</table>

`smembers(self, key: str, is_global: bool = True) -> List[Any]`

<table>
<tr><td><b>Description</b></td><td>Retrieves all members of the given Redis set as a Python set. If the set does not exist, an empty set is returned.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The Redis set key to retrieve the members for.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the keys are returned from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the keys are returned from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>List[Any]
The python list containing members of the set if it exists. Returns an empty list otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>members= store.smembers("example_set_key")</code></td></tr>
</table>

`sismember(self, key: str, member: Any, is_global: bool = True) -> bool`

<table>
<tr><td><b>Description</b></td><td>Checks whether the provided member exists in the target Redis set. Returns <code>True</code> if present, otherwise <code>False</code>.</td></tr>
<tr><td><b>Parameters</b></td><td>
<table>
<tr><td><code>key</code></td><td>(str)
The Redis set key.</td></tr>
<tr><td><code>members</code></td><td>(Any)
The member to be checked within the Redis set.</td></tr>
<tr><td><code>is_global</code></td><td>(bool) [Default is `True`]
If set to `True`, the key is checked from `global:` namespace – the keys which are replicated to/from other UnifiedServers.
If `False`, the key is checked from `local:` namespace – the keys local to this UnifiedServer.</td></tr>
</table>
</td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the member exists. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>exists = store.sismember("example_set_key", "member_n")</code></td></tr>
</table>

`get_max_mem_config(self) -> Optional[int]`

Description	Retrieves the maxmemory configuration for all master nodes in the Redis cluster.
Returns	Optional[int] The total maxmemory of the master nodes if successful, otherwise None.
Example	`max_mem = store.get_max_mem_config()`

`set_max_mem_config(self, max_memory: int) -> bool`

<table>
<tr><td><b>Description</b></td><td>Sets the maxmemory configuration for the Redis cluster, dividing the maxmemory value across all master nodes.
The max limit to set max mem config is `10380902400` bytes (~ 9.67 GiB)</td></tr>
<tr><td><b>Parameters</b></td><td><table><tr><td><code>max_memory</code></td><td>(int)
The maxmemory value to set (in bytes).</td></tr></table></td></tr>
<tr><td><b>Returns</b></td><td>(bool)
True if the maxmemory was set successfully. False otherwise.</td></tr>
<tr><td><b>Example</b></td><td><code>store.set_max_mem_config("625000000")</code></td></tr>
</table>

5.9 Data Feed Forward

The Data Feed Forward (DFF) feature enables transferring data between Unified Servers. Therefore, this feature is applicable to those who have multiple Unified Servers set up, whether it be within the same floor, across different OSATs, or across regions. At the source, the ACS Unified Server collects data from ACS Nexus to first store it locally and then forwards the data to the destination Unified Server by having the data go through Advantest Cloud.

5.9.1 Overview

The diagram below presents the architecture of Data Feed Forward.

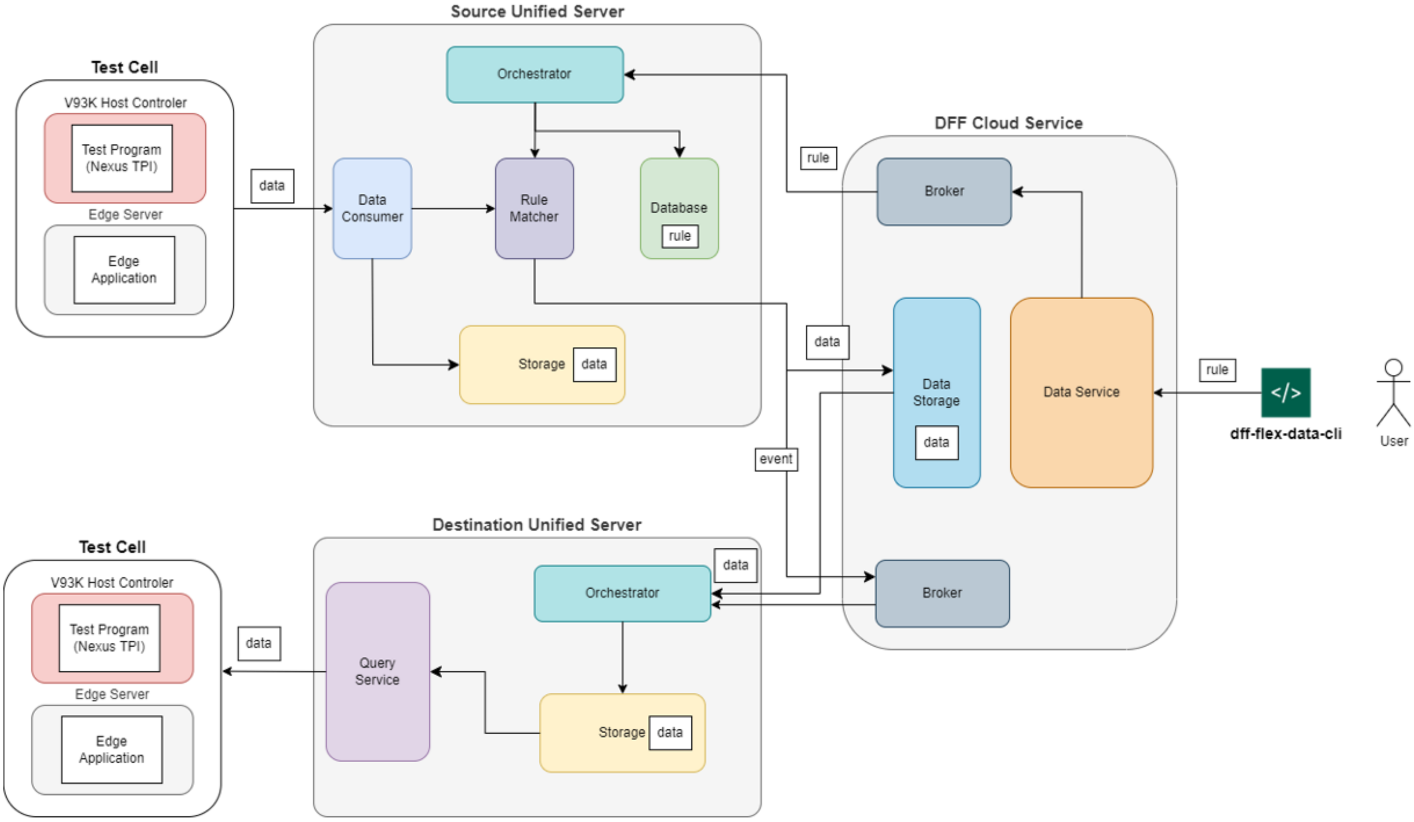


Figure 5-4. Data Feed Forward

The DFF facilitates seamless, rule-based data transfer between the Unified Servers and cloud service. It ensures efficient data collection, processing, and delivery while also maintaining transparency and control for users.

Table 5-9. DFF Components

Component	Description
Nexus	Nexus is an agent to collect the EDL data on ATE workstation and stream the data into a Unified Server.

Component	Description
Nexus TPI	The Nexus TPI is a test program interface that users can include in their test program to interact with DFF services in the ACS Unified Server.
Edge Server	A computing edge designed to handle heavy computation tasks
Unified Server—Data Consumer	Receives data from the V93K Host Controller or Edge Application and stores the data in the ACS Unified Server
Unified Server—Rule Matcher	Evaluates incoming rules to identify data that matches the specified conditions. Filters data based on criteria that is defined in SQL-like rules.
Unified Server—Orchestrator	Coordinates the operations of various components in the Unified Server, ensuring data flow.
DFF Cloud—Data Storage	Centralized repository for uploading and storing data files securely.
DFF Cloud—Data Service	Accepts rules from UI or CLI and routes the rules into the Unified Server. Provides the interface for UI and CLI to monitor the status of the data transfer.
DFF-cli	The command-line interface provides rule management and monitoring of the data upload and download status.

5.9.2 Data CLI Tool Requirement

System Requirement

- Linux on x86_64

The dff-cli command line tool is distributed as a single, self-contained executable binary file. It is installed on the Linux environment. Request the latest version from your Advantest technical contact.

5.9.3 Data CLI Commands

The dff-cli is a command-line interface designed for managing rules and monitoring the status of data feed forward operations within Advantest's ACS RTDI infrastructure. This tool allows users to perform various actions, including retrieving, listing, creating, and deleting DFF rules, as well as accessing and listing data transfer records and schemas.

The dff-cli requires users to authenticate with MyAdvantest and are subject to authorization rules. There are two types of authentications: **browser login** and **access credentials**.

Browser Login

This is the default authentication method. When the CLI program starts, before the given operation is executed, a MyAdvantest URL will be printed. The user must open the URL on an Internet browser and enter their MyAdvantest login credentials. After a successful login, a code will be displayed on the DFF UI. The user must copy this code and paste it on the terminal to proceed with the selected operation.

Access Credentials

This authentication method was designed to be more convenient for automated executions and for users who don't have access to an Internet browser. To use access credentials, the user must specify either `--username` or `--username` and `--password` arguments. For example:

```
dff-cli --username john.doe@advantest.com --password abc123#* list rules
```

If only `–username` is provided, the password will be obtained using a prompt.

Username is the user email registered on MyAdvantest. *Password* is a random string generated by DFF that can be obtained on the DFF UI in the user section (at the top right corner of the screen). Note that the password is not the user password on MyAdvantest.

The dff-cli supports four main commands, as described below. Further details on these commands are available in the subsections that follow.

- **Get**
  Get a rule, data transfer record and schema from cloud storage.
- **List**
  Lists rules, data transfer records and schemas by specific query parameters
- **Create**
  Create a rule by filling the parameters.
- **Delete**
  Delete a rule by specifying the name of rule

5.9.3.1 Get Commands

These commands fetch specific resources by ID.

Get Rule

```
dff-cli get rule <id>
```

Get data transfer record

```
dff-cli get data-transfer-record <id>
```

Get Schema

```
dff-cli get schema <id>
```

5.9.3.2 List Commands

These commands display available resources with optional filters.

List Rules

```
dff-cli list rules [--page <page>] [--pageSize <size>] [--sourceUnifiedServer <name>] [-
-destinationUnifiedServer <name>] [--sourceSchema <schema>] [--destinationSchema
<schema>]
```

Optional Filters	--page: Specify the page of results to retrieve (default: 1). --pageSize: Number of results per page (default: 10). --sourceUnifiedServer: Filter by the name of the source server uploading data. --destinationUnifiedServer: Filter by the name of the destination server downloading data. --sourceSchema: Filter by the source metadata schema name. --destinationSchema: Filter by the destination metadata schema name.

List Data Transfer Records

```
dff-cli list data-transfer-records [--page <page>] [--pageSize <size>] [--transferType
<type>] [--unifiedServerName <name>] [--ruleName <name>] [--metadataSchema <schema>] [--
status <status>]
```

Optional Filters	--page: Specify the page of results to retrieve (default: 1). --pageSize: Number of results per page (default: 10). --transferType: Filter by type of transfer (e.g., upload, download). --unifiedServerName: Filter by the Unified Server name. --ruleName: Filter by the name of the rule associated with the record. --metadataSchema: Filter by the metadata schema name. --status: Filter by the status of the data transfer (e.g., uploading, uploaded, Downloading, Downloaded).

List Schemas

```
dff-cli list schemas [--page <page>] [--pageSize <size>]
```

Optional Filters	--page: Specify the page of results to retrieve (default: 1). --pageSize: Number of results per page (default: 10).

5.9.3.3 Create Command

This command creates new rules.

Create Rule

```
dff-cli create rule <ruleName> [--sus <sourceUnifiedServer>] [--dus
<destinationUnifiedServer>] [--ss <sourceSchema>] [--ds <destinationSchema>] [-f
<filter>] [-t <TTL>]
```

Arguments	<ruleName>: Unique name for the rule (Required). --sus, --sourceUnifiedServer: Name of the Unified Server uploading the data. --dus, --destinationUnifiedServer: Name of the Unified Server downloading the data. --ss, --sourceSchema: Name of the source metadata schema. --ds, --destinationSchema: Name of the destination metadata schema. -f, --filter : SQL filter against the source metadata. Example: SELECT * FROM SCHEMA_1 WHERE LOT_ID = 'XYZ' to get data associated with the SCHEMA_1 metadata schema and matching the LOT_ID specification. Note that SCHEMA_1 must exist in the schema database and columns after WHERE must exist in the schema.

5.9.3.4 Delete Command

This command deletes a rule according to the rule name.

Delete Rule

`dff-cli delete rule <ruleName>`

Argument	<ruleName>: Unique name for the rule (Required).

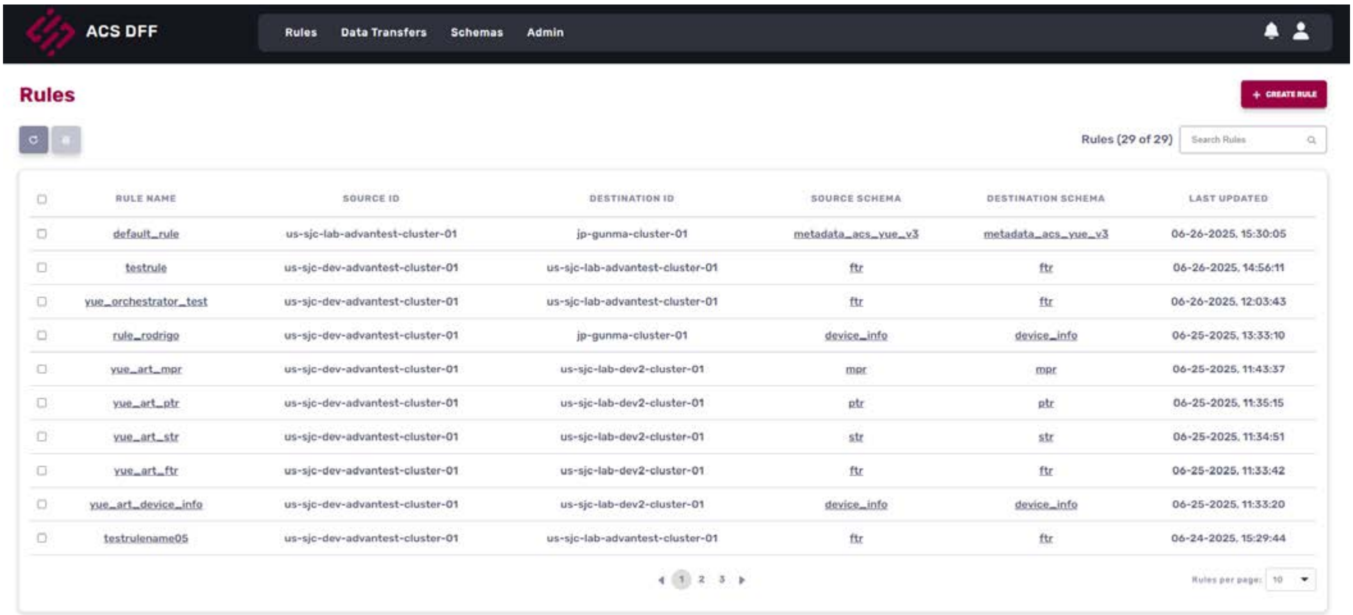
5.9.4 DFF UI

The DFF UI is the main interface through which users interact with the Data Feed Forward solution. The DFF UI is designed to be intuitive and user-friendly, providing access to key features that support the creation, monitoring, and management of data transfers throughout the testing process.

5.9.4.1 Rules

The Rules page in the Data Feed Forward (DFF) platform provides users a comprehensive interface to manage data transfer rules. Key functionalities (as illustrated in Figure 5-5) include:

- **View Rule List:** Display a complete list of existing rules and relevant details.
- **Create New Rules:** Click the "Create Rule" button to configure and register a new data transfer rule.
- **View Rule Details:** Click on any rule name to access its detailed configuration and settings.
- **Bulk Delete:** Select multiple rules using the checkboxes and delete them simultaneously using the trash icon.
- **Search Functionality:** Use the search bar to quickly find specific rules by name.



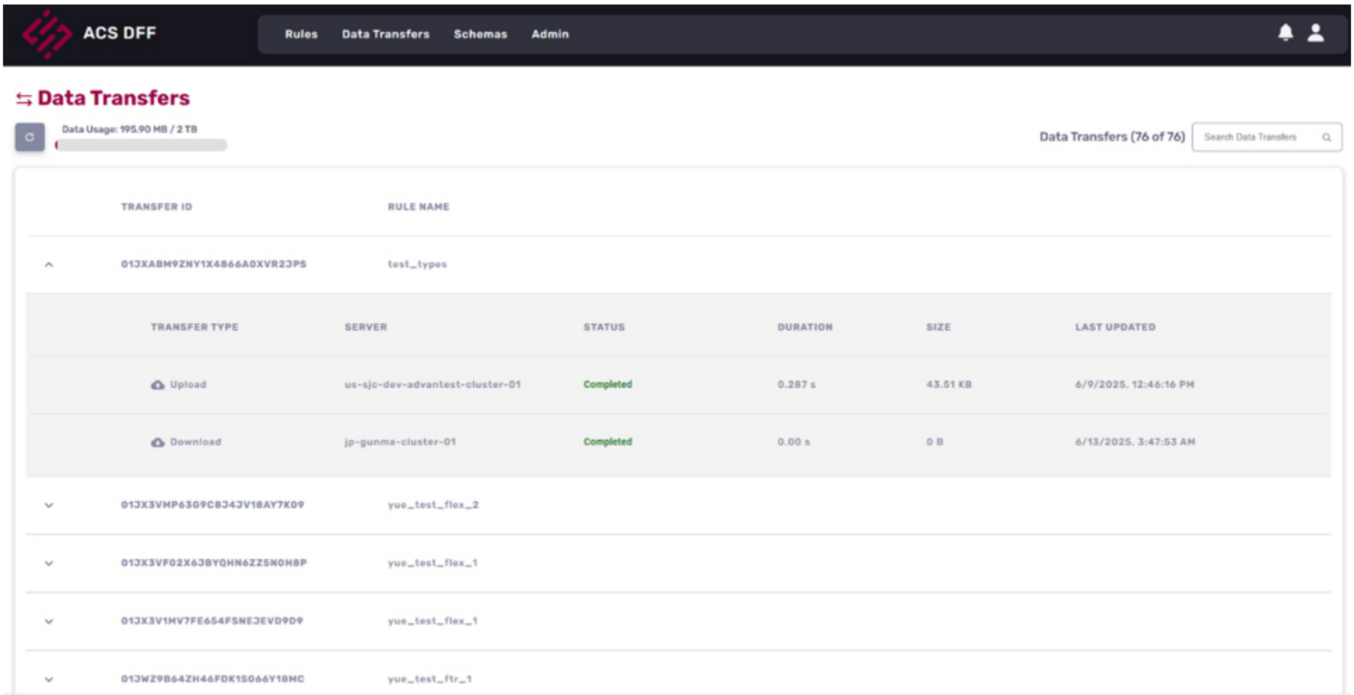
□	RULE NAME	SOURCE ID	DESTINATION ID	SOURCE SCHEMA	DESTINATION SCHEMA	LAST UPDATED
□	default_rule	us-sjc-lab-advantest-cluster-01	jp-gunma-cluster-01	metadata_acs_yue_v3	metadata_acs_yue_v3	06-26-2025, 15:30:05
□	testrule	us-sjc-dev-advantest-cluster-01	us-sjc-lab-advantest-cluster-01	ftr	ftr	06-26-2025, 14:56:11
□	yue_orchestrator_test	us-sjc-dev-advantest-cluster-01	us-sjc-lab-advantest-cluster-01	ftr	ftr	06-26-2025, 12:03:43
□	rule_rodrigo	us-sjc-dev-advantest-cluster-01	jp-gunma-cluster-01	device_info	device_info	06-25-2025, 13:33:10
□	yue_art_mpr	us-sjc-dev-advantest-cluster-01	us-sjc-lab-dev2-cluster-01	mpr	mpr	06-25-2025, 11:43:37
□	yue_art_ptr	us-sjc-dev-advantest-cluster-01	us-sjc-lab-dev2-cluster-01	ptr	ptr	06-25-2025, 11:35:15
□	yue_art_str	us-sjc-dev-advantest-cluster-01	us-sjc-lab-dev2-cluster-01	str	str	06-25-2025, 11:34:51
□	yue_art_ftr	us-sjc-dev-advantest-cluster-01	us-sjc-lab-dev2-cluster-01	ftr	ftr	06-25-2025, 11:33:42
□	yue_art_device_info	us-sjc-dev-advantest-cluster-01	us-sjc-lab-dev2-cluster-01	device_info	device_info	06-25-2025, 11:33:20
□	testrulename05	us-sjc-dev-advantest-cluster-01	us-sjc-lab-advantest-cluster-01	ftr	ftr	06-24-2025, 15:29:44

Figure 5-5. Rules Tab

5.9.4.2 Data Transfers

The Data Transfers page in the Data Feed Forward (DFF) platform provides users a detailed, read-only view of all executed data transfers. This interface is designed to help monitor and track the history of data movement across clusters. Key features (as illustrated in Figure 5-6) include:

- **Transfer Logs Visualization:** Displays a comprehensive list of data transfers, with detailed information as shown in the figure below. Each transfer event includes both upload and download records, which are organized together for clarity.
- **Rule Linking:** Each transfer record includes a link to its associated Rule, enabling users to quickly access detailed rule configurations.
- **Data Usage Tracking:** A visual progress bar on the page displays the status of data usage.
- **Search Functionality:** Users can use the search bar to filter data transfers by ID or rule name for easier navigation.



Data Transfers

Data Usage: 195.90 MB / 2 TB

Data Transfers (76 of 76)

TRANSFER ID	RULE NAME
01JXABM9ZNY1X4866A0XVR2JPS	test_types
01JX3VMP63G9C8J4JV18AY7K09	yue_test_flex_2
01JX3VF02X6JBYQHN6ZZ5N0H8P	yue_test_flex_1
01JX3V1MV7FE654FSNEJEVD9D9	yue_test_flex_1
01JWZ9B64ZH46FDK1S066Y18MC	yue_test_ftr_1

TRANSFER TYPE	SERVER	STATUS	DURATION	SIZE	LAST UPDATED
Upload	us-sjc-dev-advantest-cluster-01	Completed	0.287 s	43.51 KB	6/9/2025, 12:46:16 PM
Download	jp-gunma-cluster-01	Completed	0.00 s	0 B	6/13/2025, 3:47:53 AM

Figure 5-6. Data Transfers Tab

5.9.4.3 Schemas

The Schemas page enables users to browse all available data schemas associated with a specific Unified Server. All registered schemas are displayed in a structured format that includes the schema name and content (JSON format).

NOTE: The schema plays an important role in rule creation, as summarized below.

- During rule creation, the available source tables for a given Unified Server are matched against the schemas listed in this tab.
- The fields available for selection in filter conditions are derived from the columns defined within the corresponding tables.

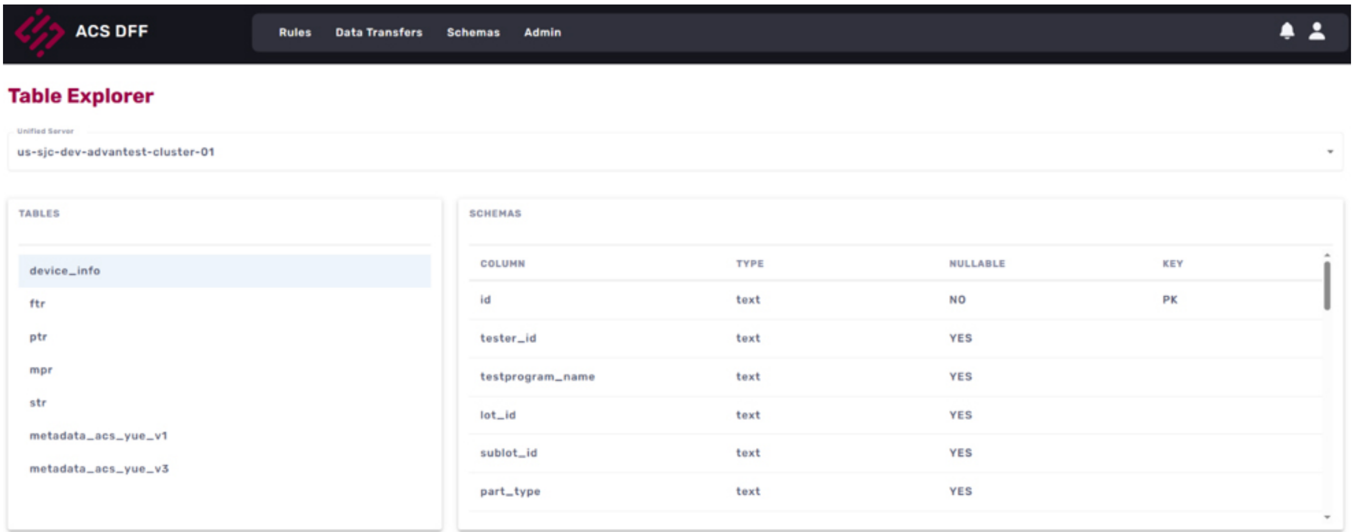


Table Explorer

Unified Server: us-sjc-dev-advantest-cluster-01

TABLES

- device_info
- ftr
- ptr
- mpr
- str
- metadata_acs_yue_v1
- metadata_acs_yue_v3

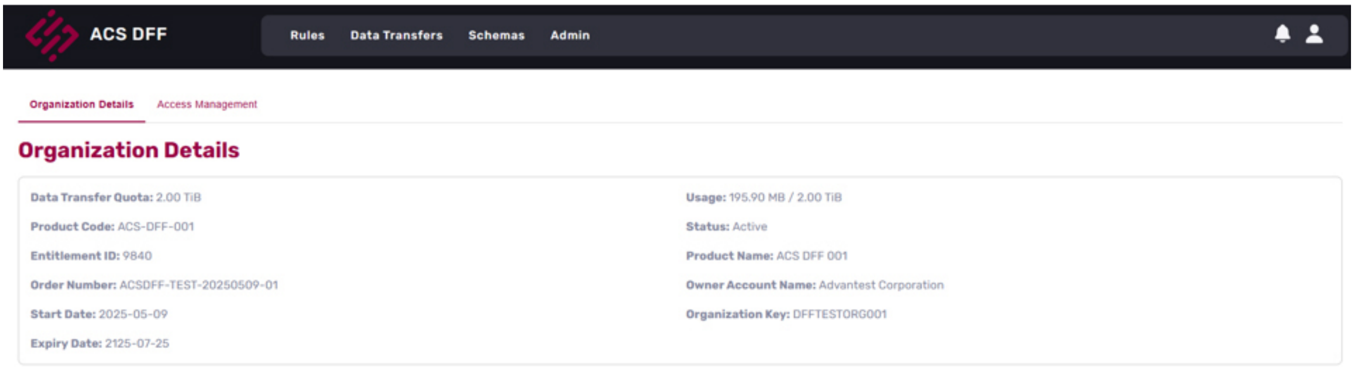
SCHEMAS

COLUMN	TYPE	NULLABLE	KEY
id	text	NO	PK
tester_id	text	YES	
testprogram_name	text	YES	
lot_id	text	YES	
sublot_id	text	YES	
part_type	text	YES	

Figure 5-7. Schemas Tab

5.9.4.4 Admin

The Admin page in the DFF UI is accessible only to administrators and is responsible for managing user permissions across the platform and displaying the organization in detail. This access management This access management section is divided into three tabs: Users, Roles, and Groups.



Organization Details

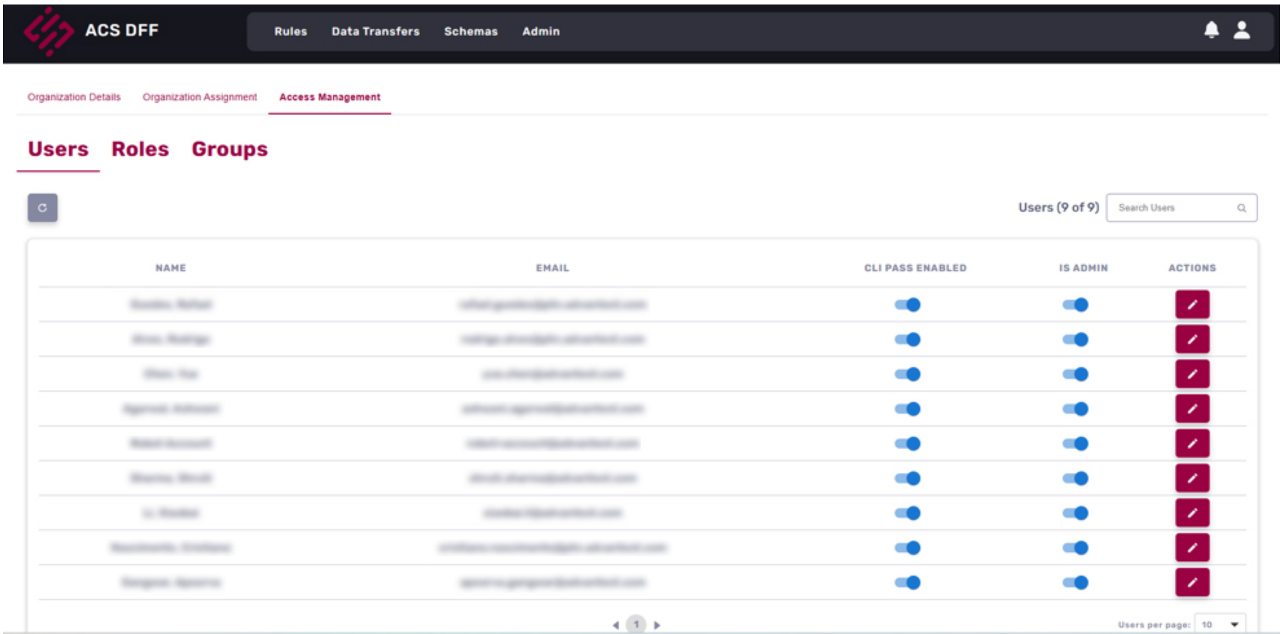
Data Transfer Quota: 2.00 TiB	**Usage:** 195.90 MB / 2.00 TiB
Product Code: ACS-DFF-001	**Status:** Active
Entitlement ID: 9840	**Product Name:** ACS DFF 001
Order Number: ACSDFF-TEST-20250509-01	**Owner Account Name:** Advantest Corporation
Start Date: 2025-05-09	**Organization Key:** DFFTESTORG001
Expiry Date: 2125-07-25	

Figure 5-8. Admin – Organization Details

Users Tab

This view lists all platform users and allows fine-grained permission control. It includes the following editable fields:

- **CLI Pass Enabled:** Toggles the visibility of the user's CLI token for use in the DFF UI.
- **Is Admin:** Grants or revokes administrator privileges to the user.
- **Actions:** Opens a permission editor for the user, allowing direct customization of CRUD scopes for each DFF functionality (e.g., Rules, Data Transfers, Schemas).



Users (9 of 9)

NAME	EMAIL	CLI PASS ENABLED	IS ADMIN	ACTIONS
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎
[redacted]	[redacted]	●	●	✎

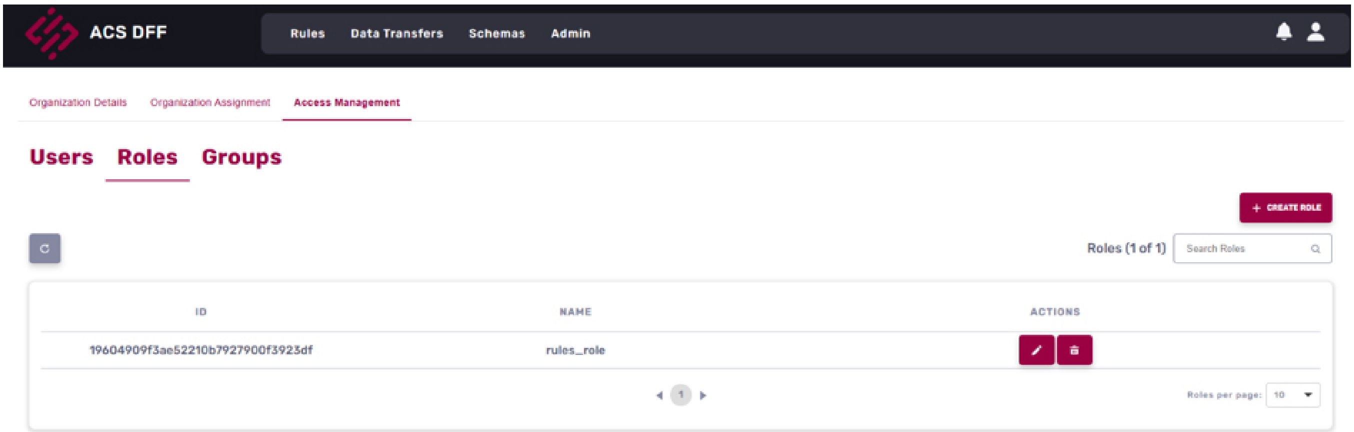
Users per page: 10

Figure 5-9. Admin – Access Management - Users Tab

Roles Tab

Roles define reusable permission sets (scopes) that can be assigned to users via groups. Features include:

- Creating new roles with selected permission scopes.
- Editing existing roles.
- Managing permissions in a centralized and consistent way.



Roles (1 of 1)

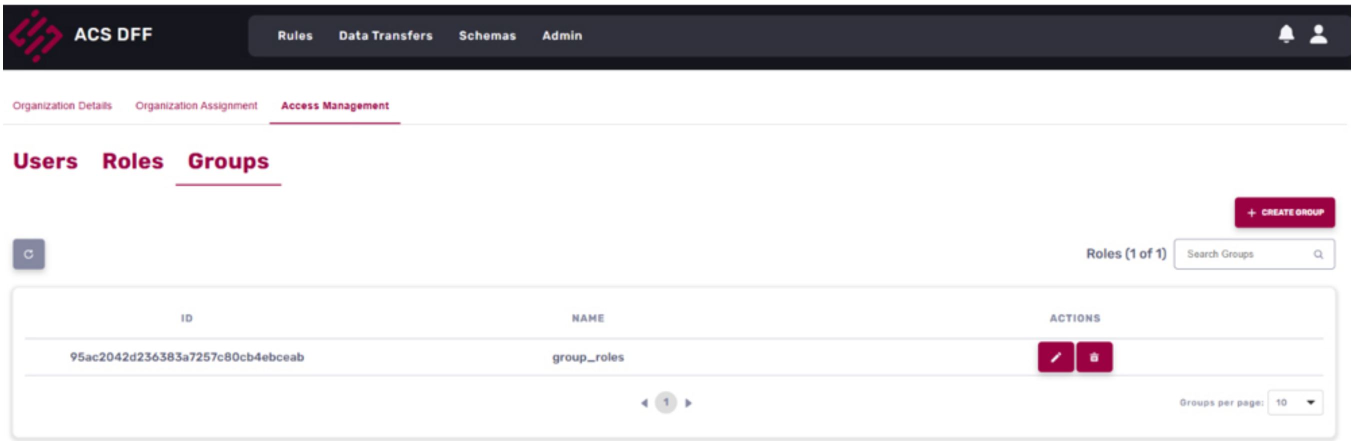
ID	NAME	ACTIONS
19604909f3ae52210b7927900f3923df	rules_role	

Figure 5-10. Admin – Access Management - Roles Tab

Groups Tab

Groups are containers that bring users and roles together. Group features include:

- **User Assignment:** Add one or more users to the group.
- **Role Assignment:** Attach one or more roles to the group.
- **Direct Scope Editing:** Optionally override or extend user permissions directly within the group without needing to use a role.



Roles (1 of 1)

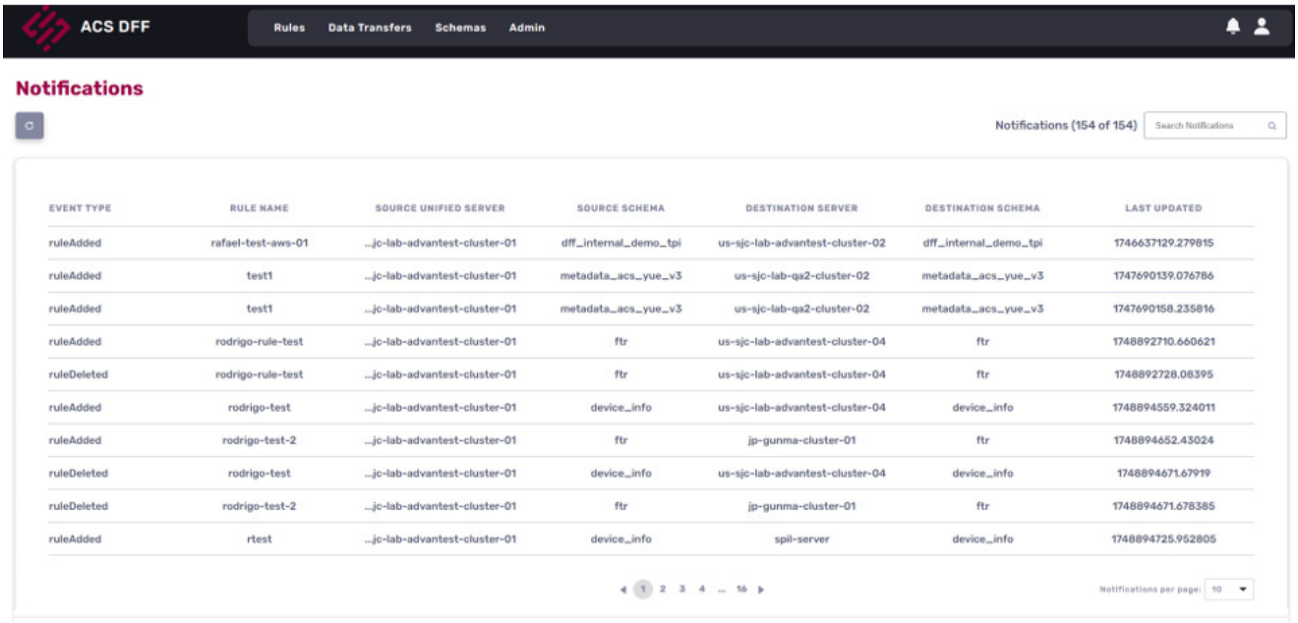
ID	NAME	ACTIONS
95ac2042d236383a7257c80cb4ebceab	group_roles	

Figure 5-11. Admin – Access Management - Groups Tab

5.9.4.5 Notifications

The Notifications tab in the Data Feed Forward (DFF) platform provides a centralized audit trail of all significant system activities. This feature ensures transparency, traceability, and accountability across all user actions and system events. The Notifications tab can be accessed by clicking **notifications_group** located in the *Name* column of the Admin Group tab. Key features of the Notifications tab include:

- **Activity Log:** Displays real-time system events such as:
  - Rule creation, modification, or deletion (ruleAdded, ruleDeleted, etc.)
  - Data transfer events (e.g., when a transfer starts or completes)
  - Any other major actions affecting system operations
- **Detailed View:** For each notification, the fields are displayed, as shown in
- **Search Functionality:** Easily filter logs using the search bar to quickly locate specific events or rule changes.



Notifications

Notifications (154 of 154)

EVENT TYPE	RULE NAME	SOURCE UNIFIED SERVER	SOURCE SCHEMA	DESTINATION SERVER	DESTINATION SCHEMA	LAST UPDATED
ruleAdded	rafael-test-aws-01	...jc-lab-advantest-cluster-01	dff_internal_demo_tpi	us-sjc-lab-advantest-cluster-02	dff_internal_demo_tpi	1746637129.279815
ruleAdded	test1	...jc-lab-advantest-cluster-01	metadata_acs_yue_v3	us-sjc-lab-qa2-cluster-02	metadata_acs_yue_v3	1747690139.076786
ruleAdded	test1	...jc-lab-advantest-cluster-01	metadata_acs_yue_v3	us-sjc-lab-qa2-cluster-02	metadata_acs_yue_v3	1747690158.235816
ruleAdded	rodrigo-rule-test	...jc-lab-advantest-cluster-01	ftr	us-sjc-lab-advantest-cluster-04	ftr	1748892710.660621
ruleDeleted	rodrigo-rule-test	...jc-lab-advantest-cluster-01	ftr	us-sjc-lab-advantest-cluster-04	ftr	1748892728.08395
ruleAdded	rodrigo-test	...jc-lab-advantest-cluster-01	device_info	us-sjc-lab-advantest-cluster-04	device_info	1748894559.324011
ruleAdded	rodrigo-test-2	...jc-lab-advantest-cluster-01	ftr	jp-gunma-cluster-01	ftr	1748894652.43024
ruleDeleted	rodrigo-test	...jc-lab-advantest-cluster-01	device_info	us-sjc-lab-advantest-cluster-04	device_info	1748894671.67919
ruleDeleted	rodrigo-test-2	...jc-lab-advantest-cluster-01	ftr	jp-gunma-cluster-01	ftr	1748894671.678385
ruleAdded	rtest	...jc-lab-advantest-cluster-01	device_info	spil-server	device_info	1748894725.952805

Notifications per page: 10

Figure 5-12. Notifications Tab

6. C++ Client API Reference

The user test program communicates with the ACS Edge Server through API calls, and the Edge Server provides in-line inferencing, computing, and machine learning capability to the test program based on the container that is loaded and executed as part of the test flow. Depending on user preferences, the ACS Edge Server may not have pre-loaded container images. However, container images can be pulled locally from the ACS Unified Server or directly from the ACS Container Hub.

6.1 ACSEdgeConn

```
class ACSEdgeConn
```

This Class is used for ACS Edge connection and communication.

6.1.1 Public Functions

ACSEdgeConn

```
~ACSEdgeConn()
```

Description: Destroy the ACSEdgeConn object

setRegHost

```
void setRegHost(const string registry_hostname)
```

Description: Set the ACS Container Hub registry Host object. The container registry host is used for creating the container, creating the volume, and adding the prefix on an image name when pulling the container image. The default value is `nullptr`. If the value is nullptr, the container image that is used in imagePull, imageDelete, contianerCreate, containerCreateStart, and volumeCreate will be added to the Mirror registry URL. If the value is not nullptr, the container image will be added to the registry URL that is passed into the function. For example, if the value is "registry.advantest.com," the container image will be pulled from "registry.advantest.com" registry host. If the value is set to “”, the container image will not have any prefix added.

Parameter	Description
`registry_hostname`	The URL for the container registry. If this parameter is an empty string, the ACS Edge Server will use the mirror container registry (ACS Unified Server).

setOptTimeout

```
void setOptTimeout(const long int time)
```

Description: Set the Opt Timeout object.

Parameter	Description
`time`	The operation timeout in seconds.

setConTimeout

```
void setConTimeout(const long int time)
```

Description: Set the Con Timeout object.

Parameter	Description
`time`	The connection time in seconds.

setProxy

```
ACSEDGEcode setProxy(const string proxyUrl, const long port)
```

Description: Set the proxy server URL and port object.

Parameter	Description
`proxyUrl`	The URL of the proxy server.
`port`	The port number of the proxy server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

setIp

```
ACSEDGEcode setIp(const string ip_addr)
```

Description: Set the ACS Edge Server IP address.

Parameter	Description
`ip_addr`	The IP address of ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getIp

```
ACSEDGEcode getIp(string &ip_addr)
```

Description: Get the ACS Edge Server IP address.

Parameter	Description
`ip_addr`	The IP address of ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

modelInit

```
ACSEDGEcode modelInit(const string container_name, const string data, string &response)
```

Description: Initialize a model in a running container.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`data`	The data passing to the model.
`response`	The response string from ACS Edge server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

modelPredict

```
ACSEDGEcode modelPredict(const string container_name, const string data, string
&response)
```

Description: Run inference in a running container.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`data`	The data passing to the model.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getServerVersion

```
ACSEDGEcode getServerVersion(string &response)
```

Description: Get the ACS Edge Server version information.

Parameter	Description
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getServerSystemUsage

```
ACSEDGEcode getServerSystemUsage(string &response)
```

Description: Get the ACS Edge Server system usage.

Parameter	Description
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getRunningContainers

```
ACSEDGEcode getRunningContainers(string &response)
```

Description: Get the list of running containers.

Parameter	Description
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getContainerStatus

```
ACSEDGEcode getContainerStatus(const string container_name, string &response)
```

Description: Stores the state of the container in response string. The stored state can be one of either paused, created, restarting, running, removing, exited, or dead.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

imagePull

```
ACSEDGEcode imagePull(const string image_name, const string tag, const string username,
const string password, const string manifest_filePath, string &response)
```

Description: Pull a docker image from the ACS Container Hub registry into the ACS Edge Server.

Parameter	Description
`image_name`	The name of the image in the ACS Container Hub registry.
`tag`	The tag of the image. The default value is "latest."
`username`	The username to be used to log in to the ACS Container Hub registry.
`password`	The password to be used to log in to the ACS Container Hub registry.
`manifest_filePath`	The path of the manifest file.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

imagePull

```
ACSEDGEcode imagePull(const string image_name, const string tag, const string username,
const string password, string &response)
```

Description: Pull a docker image from the ACS Container Hub registry into the ACS Edge Server.

Parameter	Description
image_name	The name of the image in the ACS Container Hub registry.
tag	The tag of the image. The default value is "latest."
username	The username to be used to log in to the ACS Container Hub registry.
password	The password to be used to log in to the ACS Container Hub registry.
response	The response string from ACS Edge Server.

Returns	Description
ACSEDGEcode	The status code for the client SDK.

imagePull

```
ACSEDGEcode imagePull(const string image_name, const string tag, const string
manifest_filePath, string &response
```

Description: Pull a docker image from the ACS Container Hub registry into the ACS Edge Server.

Parameter	Description
image_name	The name of the image in the ACS Container Hub registry.
tag	The tag of the image. The default value is "latest."
manifest_filePath	The path of the manifest file.
response	The response string from ACS Edge Server.

Returns	Description
ACSEDGEcode	The status code for the client SDK.

imagePull

```
ACSEDGEcode imagePull(const string image_name, const string tag, string &response)
```

Description: Pull a docker image from the ACS Container Hub registry into the ACS Edge Server.

Parameter	Description
`image_name`	The name of the image in the ACS Container Hub registry.
`tag`	The tag of the image. The default value is "latest."
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

imageLoad

```
ACSEDGEcode imageLoad(const string image_file_path, string &response)
```

Description: Loads a docker image from a tar file into the ACS Edge Server.

NOTE: By default, this function is not supported. For support of this function, contact your Advantest representative.

Parameter	Description
`image_file`	The name path of the tar file
`response`	The response string from ACS Edge Server.

imageDelete

```
ACSEDGEcode imageDelete(const string image_name, const string tag, string &response)
```

Description: Delete an image in the ACS Edge Server.

Parameter	Description
`image_name`	The name of the image in the ACS Edge Server.
`tag`	The tag of the image. The default value is "latest."
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

imageDelete

```
ACSEDGEcode imageDelete(const string image_name, string &response)
```

Description: Delete an image with the "latest" tag in the ACS Edge Server.

Parameter	Description
image_name	The name of the image in the ACS Edge Server.
response	The response string from ACS Edge Server.

Returns	Description
ACSEDGEcode	The status code for the client SDK.

containerCreate

```
ACSEDGEcode containerCreate(const string image_name, const string container_name,
ContainerConfig &contConfig, string &response)
```

Description: Create and start a container using an image (referenced by image_name) and using container_name as the container name (which can be then referenced by name) and enable the `<name>` proxy to interact with the container. To configure the container, a ContainerConfig object is needed to pass into the function.

Parameter	Description
image_name	The name of the image in the ACS Edge Server.
container_name	The name/alias used when the container was created.
contConfig	The container configuration class object.
response	The response string from ACS Edge Server.

Returns	Description
ACSEDGEcode	The status code for the client SDK.

containerCreate

```
ACSEDGEcode containerCreate(const string image_name, const string container_name, string
&response)
```

Description: Create and start a container using an image (referenced by `image_name`) and using `container_name` as the container name (which can be then referenced by name) and enable the `<name>` proxy to interact with the container.

Parameter	Description
`image_name`	The name of the image in the ACS Edge Server.
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

containerStart

```
ACSEDGEcode containerStart(const string container_name, string &response
```

Description: Start a previously created container.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

containerCreateStart

```
ACSEDGEcode containerCreateStart(const string image_name, const string container_name,
ContainerConfig &contConfig, string &response)
```

Description: Create and start a container using an image (referenced by `image_name)` and using `container_name` as the container name (which can be then referenced by name) and enable the `<name>` proxy to interact with the container. The `gpu` option can be used to enable access to the GPU device for the container. The `volume_attach` parameter is used to attach mounted volumes under the /data/ directory within the container. For each `volume_attach` entry, a subdirectory under /data/ is created. The `ports` option can be used to expose container ports.

Parameter	Description
`image_name`	The name of image in the ACS Edge Server.
`container_name`	The name/alias used when the container was created.
`contConfig`	The container configuration class object.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

containerCreateStart

```
ACSEDGEcode containerCreateStart(const string image_name, const string container_name,
string &response)
```

Description: Create and start a container using an image (referenced by `image_name)` and using `container_name` as the container name (which can be then referenced by name) and enable the `<name>` proxy to interact with the container. The `gpu` option can be used to enable access to the GPU device for the container.

Parameter	Description
`image_name`	The name of image in the ACS Edge Server.
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

containerStop

```
ACSEDGEcode containerStop(const string container_name, string &response)
```

Description: Stop a running container.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

containerDelete

```
ACSEDGEcode containerDelete(const string container_name, string &response)
```

Description: Delete a non-running container.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getContainerLogs

```
ACSEDGEcode getContainerLogs(const string container_name, string &response)
```

Description: Get the logs of a container.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

volumeCreate

```
ACSEDGEcode volumeCreate(const string name, const string image_name, const string tag,
string &response)
```

Description: Create a volume and copy files from the provided image.

Parameter	Description
`name`	The volume name/alias to be used for all references.
`image_name`	The name of the image in the ACS Edge Server.
`tag`	The tag of the image. The default value is "latest."
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

volumeDelete

```
ACSEDGEcode volumeDelete(const string name, string &response)
```

Description: Remove a volume from the ACS Edge Server.

Parameter	Description
`name`	The volume name/alias to be used for all references.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

containerPurge

```
ACSEDGEcode containerPurge(const string container_name, string &response)
```

Description: Stop the container and remove the container, the image, and associated volumes.

Parameter	Description
`container_name`	The volume name/alias to be used for all references.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

post

```
ACSEDGEcode post(const string container_name, const string path, const map<string,
string> &parameters, const string data, string &response)
```

Description: The general HTTP POST function. Several options are provided for sending RESTFul command to the container interface. The HealthCheck endpoint is used to check the health of the container that has been added to the container image.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`path`	The URL path to the exact location of the container endpoint.
`parameters`	The query parameters to the container interface.
`data`	The data stream fed into the container.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

post

```
ACSEDGEcode post(const string container_name, const string path, const string data,
string &response)
```

Description: The general HTTP POST function. Several options are provided for sending RESTFul command to the container interface.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`path`	The URL path to the exact location of the container endpoint.
`data`	The data stream fed into the container.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

post

```
ACSEDGEcode post(const string container_name, const string path, const map<string,
string> &parameters, string &response)
```

Description: The general HTTP POST function. Several options are provided for sending RESTFul command to the container interface.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`path`	The URL path to the exact location of the container endpoint.
`parameters`	The query parameters to the container interface.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

post

```
ACSEDGEcode post(const string container_name, const string path, string &response)
```

Description: The general HTTP POST function. Several options are provided for sending RESTFul command to the container interface.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`path`	The URL path to the exact location of the container endpoint.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

get

```
ACSEDGEcode get(const string container_name, const string path, const map<string,
string> &parameters, string &response)
```

Description: The general HTTP GET function. Several options are provided for sending RESTFul command to the container interface.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`path`	The URL path to the exact location of the container endpoint.
`parameters`	The query parameters to the container interface.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

get

```
ACSEDGEcode get(const string container_name, const string path, string &response)
```

Description: The general HTTP GET function. Several options are provided for sending RESTFul command to the container interface.

Parameter	Description
`container_name`	The name/alias used when the container was created.
`path`	The URL path to the exact location of the container endpoint.
`response`	The response string from ACS Edge Server.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

deinit

```
ACSEDGEcode deinit()
```

Description: The de-initialization function to clear the ACS Edge Instance.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getInstance

```
static ACSEdgeConn &getInstance()
```

Description: Get the instance object.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

getInstance

```
static ACSEdgeConn &getInstance(const string ip_addr)
```

Description: Get the instance object.

Returns	Description
`ACSEDGEcode`	The status code for the client SDK.

6.2 ContainerConfig

```
class ContainerConfig
```

This class is used to add different options to configure the container. You can use `setOption()` method to add configuration options to the container. This method can be called in a chain. For example:

```
ContainerConfig cont;
cont.setOption("gpu",true).setOption("hostname","customhostname")
```

The following options are available for the current version (3.3.x) of the ACS Edge Server.

- **`gpu`**
  The `gpu` option can be used to enable access to the GPU device for the container. The value of this option is boolean value (true or false). If not provided, the container can't use `gpu`.

- **`volume_attach`**
  The `volume_attach` option is used to attach mounted volumes under the /data/ directory within the container. For each `volume_attach` entry, a subdirectory under /data/ is created. The value of the option is a C++ vector of strings. Each string is the name of the volume to be attached. For example, if the vector contains {"vol1","vol2"}, then the volumes `/data/0` and `/data/1` are attached on the container. By default, the volume is read only. If you want to write data into volume, add ":rw" after the volume name. For example, if the vector contains {"vol1:rw","vol2"}, then the volume `vol1` is read and write, but the volume `vol2` is read only.

- **`ports`**
  The ports option can be used to expose container ports. A list of ports that are exposed on the container is provided using C++ string vector. Each string is a port number. The ports that are exposed by the `ports` option can only be communicated by the SDK `post()` and `get()`, which are authorized by the ACS Edge's mTLS. Note that a single port is mapped to the default container name endpoint, whereas multiple ports map to endpoint `/<containerName>-<port>`. For example, if the container name is "demo" and if the vector contains `{"8001","8002"}`, then the container will be exposed on ports 8001 and 8002. The two endpoints generated are "demo-8001" and "demo-8002". If the vector is empty, then the container will only be exposed on port 8000 as the default and the endpoint is "demo." If the vector is `{"8001"}`, the container will be exposed on port 8001 but the only one endpoint is still "demo."

- **`publish-ports`**
  The publish-ports option exposes and publishes container ports. A list of ports that are exposed on the container is provided using C++ string vector. The format of the value is "hostPort:containerPort/protocol." The protocol can be tcp, udp, sctp, or dccp. If protocol is missing, the default protocol used is /tcp. For example, if the vector contains `{"8001:8001/tcp","8002:8002/udp}`, then the container will be exposed on port 8001 (via tcp protocol) and port 8002 (via udp protocol) on the Edge Server.

- **`hostname`**
  The hostname option defines the hostname of the container. If not provided, the default short version of the container ID will be used instead.

- **`auto_remove`**
  The auto_remove option can be used to remove the container when it is stopped. The value of the option is a boolean value (true or false). If not provided, the container will not be removed.

- **`environment`**
  The environment option can be used to set environment variables inside the container. The value of the option is a C++ vector of strings. Each string is an environment variable definition. For example, if the vector contains {"ENV1=VALUE1","ENV2=VALUE2"}, then the following environment variables are set: ENV1=VALUE1 and ENV2=VALUE2. If the vector is empty, then no environment variables are set.

- **`local`**
  The local option can be used to enable the server to use a local image. If local=True, it ignores the registry value set on the `setRegHost()` function and no prefix is added to the image name.

- **`tester_id`**
  The tester_id option is required when the client tries to create a container on the Edge resilience service in the ACS Unified Server. The value of the option is a string, which is the hostname of the Workstation.

- **`metrics_port`**
  The metrics_port option is required when the client tries to create a container on the Edge Server and also wants to retrieve the metrics data from the container into the RTDI monitoring pipeline. The (integer) value of the option is the port number of the container that is used to expose the metrics data. For example, if the endpoint is "demo-app:8000/metrics" then “8000” is the value of the option.

- **`metrics_path`**
  The metrics_path option is required when the client tries to create a container on the Edge Server and also wants to retrieve the metrics data from the container into the RTDI monitoring pipeline. The (string) value of the option is the path of the metrics endpoint. For example, if the endpoint is “demo-app:8000/metrics” then "/metrics" is the value of the option. The value of the option is a string.

- **`metrics_scrape_interval`**
  The metrics_scrape_interval option is required when the client tries to create a container on the Edge Server and also wants to retrieve the metrics data from the container into RTDI monitoring pipeline. The (integer) value of the option is the interval of the metrics data scraping. The unit of the value is seconds and the default value is 10.

- **`metrics_scrape_timeout`**
  The metrics_scrape_timeout option is required when the client tries to create a container on the Edge Server and also wants to retrieve the metrics data from the container into the RTDI monitoring pipeline. The (integer) value of the option is the timeout of the metrics data scraping. The unit of the value is seconds and the default value is 5.

- **`restart_policy`**
  The restart policy is required when the client tries to create a container on the Edge Server and also wants to apply the restart policy in case a Docker container crashes or stops. The value of the option is a string. The default value is *always*. This means that the container will always restart (regardless of the exit status) if it is not provided from the client.

  All available options include:

no	The container will not restart automatically.
always	The container will always restart regardless of the exit status. It will also restart on system reboot or Docker daemon restart.
unless-stopped	Similar to *always*, but the container will not restart if it was stopped manually.
on-failure	The container will restart only if it exits with a non-zero status. This option is typically used to handle failures in the container's process. Also, the retry count is fixed to 3 and it is non-changeable.

6.2.1 Public Function

`ContainerConfig()`

Description: Construct a new container config object.

`~ContainerConfig()`

Description: Destroy the container config object.

`setOption`

```
ContainerConfig &setOption(const string option, const string value)
```

Description: Set the option object. The value of the option is a string. The option `hostname` uses this method to set the hostname and the restart policy of the container.

Parameter	Description
`option`	The container configuration options.
`value`	The string value of the option.

Returns
`ContainerConfig&` the object reference.

Example Code:

```
ContainerConfig cont;
string hostname = "customhostname";
string restart_policy = “unless-stopped”;
cont.setOption("hostname",hostname).setOption(“restart_policy”, restart_policy);
```

setOption

```
ContainerConfig &setOption(const string option, const char value)
```

Description: Set the option object. The value of the option is char*.

Parameter	Description
`option`	The container configuration options.
`value`	The char* value of the option.

Returns
`ContainerConfig&` the object reference.

Example Code:

```
ContainerConfig cont;
cont.setOption("hostname","customhostname");
```

setOption

```
ContainerConfig &setOption(const string option, const bool value)
```

Description: Set the option object. The value is boolean (true or false). The option `gpu` and `auto_remove` use this method to set the value.

Parameter	Description
`option`	The container configuration options.
`value`	The boolean value of the option.

Returns
`ContainerConfig&` the object reference.

Example Code:

```
ContainerConfig cont;
cont.setOption("gpu",true).setOption("auto_remove",false);
```

setOption

```
ContainerConfig &setOption(const string option, vector<string> value)
```

Description: Set the option object. The value is a string vector. The option `volume_attach`, `ports,` `publish-ports` and `environment` use this method to set the value.

Parameter	Description
`option`	The container configuration options.
`value`	The vector value of the option.

Returns
`ContainerConfig&` the object reference.

Example Code:

```
ContainerConfig cont;
vector<string> publish-ports;
publish-ports.push_back("8001/tcp:8001");
publish-ports.push_back("8002/udp:8002");
vector<string> ports;
ports.push_back("8003");
ports.push_back("8004");
vector<string> environment;
environment.push_back("ENV1=VALUE1");
vector<string> volumes;
volumes.push_back("volume1");
cont.setOption("publish-ports",publish-ports)
    .setOption("ports",ports).setOption("environment",environment)
    .setOption("volume_attach",volumes);
```

setOption

```
ContainerConfig &setOption(const string option, const int value)
```

Description: Set the option object.

Parameter	Description
`option`	The container configuration options.
`value`	The integer value of the option.

Returns
`ContainerConfig&` the object reference.

toString

```
string toString()
```

Description: Return container configuration parameter in a string. The string data includes all configuration information.

Example Code:

```
ContainerConfig cont;
string hostname = "customhostname";
cont.setOption("hostname",hostname);
string config = cont.toString();
```

getErrorCode

```
ACSEDGEcode getErrorCode()
```

Description: Return the container configuration error code.

Returns	Description
`ACSEDGEcode`	The error code.

Example Code:

```
ContainerConfig cont;
string hostname = "customhostname";
cont.setOption("hostname",hostname);
ACSEDGEcode err = cont.getErrorCode();
```

6.3 ACSEDGEcode

```
enum ACSEDGECode
```

This is the Help function to parse the response string from the ACS Edge Server compatible container. The response string is formatted by Json. There are two fields in the response string; msg and data. The msg (message) field is followed by the customized message that was applied in ACS Edge container. The data field is followed by the prediction/computation result, which is encoded by base64 so the `result_parser` function can parse the response string and return the value of the specified field name.

```
@param name the field name in the json formatted response data from ACS Edge Server.
```

ACS Edge Server

Param response: The json-formatted response data from the ACS Edge server.

Return: The string extracted from the json data string based on the name parameter. All possible status codes are included in libACSEdgeConn. Future versions may return other values. Never remove any values. The return codes should remain the same.

Enumerator values for ACSEDGEcode are listed in Table 6-1 below.

Table 6-1. ACSEDGEcode Enumerators

Enumerators		Additional Information
ACSE_OK	= 0	
ACSE_UNSUPPORTED_PROTOCOL	= 1	
ACSE_FAILED_INIT	= 2	
ACSE_URL_MALFORMAT	= 3	
ACSE_NOT_BUILT_IN	= 4	
ACSE_COULDNT_RESOLVE_PROXY	= 5	
ACSE_COULDNT_RESOLVE_HOST	= 6	
ACSE_COULDNT_CONNECT	= 7	
ACSE_WEIRD_SERVER_REPLY	= 8	
ACSE_REMOTE_ACCESS_DENIED	= 9	A service was denied by the server due to lack of access. (When login fails this is not returned.)
ACSE_FTP_ACCEPT_FAILED	= 10	
ACSE_FTP_WEIRD_PASS_REPLY	= 11	
ACSE_FTP_ACCEPT_TIMEOUT	= 12	Timeout occurred accepting the server.
ACSE_FTP_WEIRD_PASV_REPLY	= 13	
ACSE_FTP_WEIRD_227_FORMAT	= 14	
ACSE_FTP_CANT_GET_HOST	= 15	
ACSE_HTTP2	= 16	There is a problem in the http2 framing layer.

Enumerators	Additional Information
ACSE_FTP_COULDNT_SET_TYPE = 17	
ACSE_PARTIAL_FILE = 18	
ACSE_FTP_COULDNT_RETR_FILE = 19	
ACSE_OBSOLETE20 = 20	Not used.
ACSE_QUOTE_ERROR = 21	Quote command failure.
ACSE_HTTP_RETURNED_ERROR = 22	
ACSE_WRITE_ERROR = 23	
ACSE_OBSOLETE24 = 24	Not used
ACSE_UPLOAD_FAILED = 25	Failed to upload command.
ACSE_READ_ERROR = 26	Could not open file or could not read from file.
ACSE_OUT_OF_MEMORY = 27	OUT_OF_MEMORY may sometimes indicate a conversion error instead of a memory allocation error if DOES_CONVERSIONS is defined.
ACSE_OPERATION_TIMEDOUT = 28	The timeout time was reached.
ACSE_OBSOLETE29 = 29	Not used.
ACSE_FTP_PORT_FAILED = 30	FTP PORT operation failed.
ACSE_FTP_COULDNT_USE_REST = 31	The REST command failed.
ACSE_OBSOLETE32 = 32	Not used.
ACSE_RANGE_ERROR = 33	RANGE command did not work.
ACSE_HTTP_POST_ERROR = 34	
ACSE_SSL_CONNECT_ERROR = 35	Something went wrong when connecting with SSL.
ACSE_BAD_DOWNLOAD_RESUME = 36	Could not resume download.
ACSE_FILE_COULDNT_READ_FILE = 37	
ACSE_LDAP_CANNOT_BIND = 38	
ACSE_LDAP_SEARCH_FAILED = 39	
ACSE_OBSOLETE40 = 40	Not used.
ACSE_FUNCTION_NOT_FOUND = 41	Not used.
ACSE_ABORTED_BY_CALLBACK = 42	
ACSE_BAD_FUNCTION_ARGUMENT = 43	
ACSE_OBSOLETE44 = 44	Not used.
ACSE_INTERFACE_FAILED = 45	OPT_INTERFACE failed.
ACSE_OBSOLETE46 = 46	Not used.
ACSE_TOO_MANY_REDIRECTS = 47	Catching endless re-direct loops.

Enumerators	Additional Information
`ACSE_UNKNOWN_OPTION = 48`	User specified an unknown option.
`ACSE_TELNET_OPTION_SYNTAX = 48`	Malformed telnet option.
`ACSE_OBSOLETE50 = 50`	Not used.
`ACSE_OBSOLETE51 = 51`	Not used.
`ACSE_GOT_NOTHING = 52`	When there is a specific error.
`ACSE_SSL_ENGINE_NOTFOUND = 53`	SSL crypto engine not found.
`ACSE_SSL_ENGINE_SETFAILED = 54`	Cannot set SSL crypto engine as default.
`ACSE_SEND_ERROR = 55`	Failed sending network data.
`ACSE_RECV_ERROR = 56`	Failure in receiving network data.
`ACSE_OBSOLETE57 = 57`	Not used.
`ACSE_SSL_CERTPROBLEM = 58`	There is a problem with the local certificate.
`ACSE_SSL_CIPHER = 59`	Could not use specified cipher.
`ACSE_PEER_FAILED_VERIFICATION = 60`	The peer’s certificate or fingerprint failed verification.
`ACSE_BAD_CONTENT_ENCODING = 61`	Unrecognized or bad encoding.
`ACSE_LDAP_INVALID_URL = 62`	Invalid LDAP URL.
`ACSE_FILESIZE_EXCEEDED = 63`	Maximum file size is exceeded.
`ACSE_USE_SSL_FAILED = 64`	Requested FTP SSL level failed.
`ACSE_SEND_FAIL_REWIND = 65`	Sending the data requires a rewind that failed.
`ACSE_SSL_ENGINE_INITFAILED = 66`	Failed to initialize ENGINE.
`ACSE_LOGIN_DENIED = 67`	User, password, or similar was not accepted and login failed.
`ACSE_TFTP_NOTFOUND = 68`	File not found on the server.
`ACSE_TFTP_PERM = 69`	There is a permission problem on server.
`ACSE_REMOTE_DISK_FULL = 70`	Out of disk space on the server.
`ACSE_TFTP_ILLEGAL = 71`	Illegal TFTP operation.
`ACSE_TFTP_UNKNOWNID = 72`	Unknown transfer ID.
`ACSE_REMOTE_FILE_EXISTS = 73`	File already exists.
`ACSE_TFTP_NOSUCHUSER = 74`	No such user.
`ACSE_CONV_FAILED = 75`	Conversion Failed.
`ACSE_CONV_REQD = 76`	
`ACSE_CACERT_BADFILE = 77`	Could not load CACERT file. It is either missing or in the wrong format.
`ACSE_REMOTE_FILE_NOT_FOUND = 78`	remote file not found

Enumerators		Additional Information
`ACSE_SSH_ERROR`	= 79	There is an error from the SSH layer.
`ACSE_SSL_SHUTDOWN_FAILED`	= 80	Failed to shut down the SSL connection.
`ACSE_AGAIN`	= 81	Socket is not ready for send/receive. Wait until it is ready and try again.
`ACSE_SSL_CRL_BADFILE`	= 82	Could not load CRL file. It is either missing or in the wrong format.
`ACSE_SSL_ISSUER_ERROR`	= 83	Issuer check failed.
`ACSE_FTP_PRET_FAILED`	= 84	A PRET command failed
`ACSE_RTSP_CSEQ_ERROR`	= 85	There is a mismatch of RTSP CSeq numbers.
`ACSE_RTSP_SESSION_ERROR`	= 86	There is a mismatch of RTSP Session Ids.
`ACSE_FTP_BAD_FILE_LIST`	= 87	Unable to parse the FTP file list.
`ACSE_CHUNK_FAILED`	= 88	A chunk callback reported error occurred.
`ACSE_NO_CONNECTION_AVAILABLE`	= 89	No connection available. The session will be queued.
`ACSE_SSL_PINNEDPUBKEYNOTMATCH`	= 90	The specified pinned public key did not match.
`ACSE_SSL_INVALIDCERTSTATUS`	= 91	Invalid certificate status.
`ACSE_HTTP2_STREAM`	= 92	There is a stream error in HTTP/2 framing layer.
`ACSE_RECURSIVE_API_CALL`	= 93	An API function was called from inside a callback.
`ACSE_AUTH_ERROR`	= 94	An authentication function returned an error.
`ACSE_HTTP3`	= 95	There is a problem with HTTP/3 layer.
`ACSE_QUIC_CONNECT_ERROR`	= 96	There is a QUIC connection error.
`ACSE_IN_NUM_OUT_OF_RANGE`	= 97	Input number is out of range.
`ACSE_NO_CONTAINER_NAME`	= 98	A container NAME has not been set.
`ACSE_NO_IMAGE_NAME`	= 99	An image NAME has not been set.
`ACSE_NO_VOLUME_NAME`	= 100	A volume NAME has not been set.
`ACSE_SERVER_API_ERROR`	= 101	There is an error from server.
`ACSE_INVALID_PORT_NUMBER`	= 102	Invalid Port Number.
`ACSE_EMPTY_PROXY_URL`	= 103	There is an empty string on the proxy URL.
`ACSE_EMPTY_IP_INPUT`	= 104	There is an empty IP address input on the setIP function.
`ACSE_IP_NOT_FOUND`	= 105	ACSEdge IP address not found.
`ACSE_MANIFEST_FILE_NO_FOUND`	= 106	Manifest file not found.
`ACSE_CONTAINER_NAME_CONFLICT`	= 107	There is a container name conflict.
`ACSE_INVALID_ENVIRONMENT`	= 108	Invalid environment variable definition

6.4 Establishing Connection to the ACS Edge Server

This tutorial uses SmarTest 7 examples to demonstrate the procedure for establishing a connection to ACS Edge Server. Similar steps will apply for other C++ platforms.

1. Create a connection object.

Include the `ACSEdgeConn.hpp` header file into the C++ program.

```
#include "ACSEdgeConn.hpp"
```

Create an object to interface with the ACS Edge Server. For the V93000 test system, it is not necessary to specify an IP address, but if the V93000 Workstation does not have a local connection with the ACS Edge Server (such as for Cloud testing) or the client SDK is used on a system other than the V93000, the IP address must be provided.

```
//V93000 system with ACS Edge Server
ACSEdgeConn& conn = ACSEdgeConn::getInstance();

//Non V93000 system with ACS Edge Server or V93000 system with ACS Edge cloud
instance
ACSEdgeConn& conn = ACSEdgeConn::getInstance("10.10.120.2");
```

2. Configure the proxy server (optional).

If a proxy server exists between the ACS Edge Server and the host controller, call the `setProxy()` API to configure the proxy server. This option is most commonly used in a development environment where the user wants to access an ACS Edge Server (placed outside of the facility network) through a proxy server. In a production environment, the proxy is not necessary since a local connection exists between the test system and the ACS Edge Server.

There are three types of proxy servers available: HTTP, HTTPS and SOCKS. The proxy string is prefixed with [scheme]:// to specify the proxy type. The following example shows the setup for three different proxy servers with an IP of 196.168.555.555 and port 6666.

```
ACSEdgeConn& conn = ACSEdgeConn::getInstance();
conn.setProxy("http://196.168.555.555",6666L);
conn.setProxy("https://196.168.555.555",6666L);
conn.setProxy("socks4://196.168.555.555",6666L);
conn.setProxy("socks4a://196.168.555.555",6666L);
conn.setProxy("socks5://196.168.555.555",6666L);
conn.setProxy("socks5h://196.168.555.555",6666L);
```

3. Connect to the ACS Edge Server

Creating an interface object does not establish a connection to the ACS Edge Server. The connection is established through interface object method calls. An easy way to test the connection is by fetching the version information from the server:

```
ACSEdgeConn& conn = ACSEdgeConn::getInstance();

string response;
ACSEDGEcode  status_code = conn.getServerVersion(response);

if (status_code == 0){
    cout << response << endl;
}
else{
    cout << "Error code: " << (int)status_code << " at " <<__FILE__ << endl;
    cout << response << endl;
}
```

All methods return an ACSEDGEcode result that indicates the status of the connection. If the result is anything other than ACSE_OK, an error has occurred.

4. Setting Timeout

There are two timeout configurations for communication between client and server.

- Connection timeout
  The maximum time (in seconds) allowed to connect to the server. The default is 15 seconds.
- Operation timeout
  The maximum time (in seconds) allowed to complete any operations. The default is 15 seconds. For example, when pulling a large image (1GB) from a container registry, if the image pull operation cannot complete within the time frame, the `imagePull()` function will return status code 28, meaning OPERATION_TIMEOUT.

```
ACSEdgeConn& conn = ACSEdgeConn::getInstance();
conn.setRegHost("hub.demohub.io");
conn.setOptTimeout(2000L);
conn.setConnTimeout(2000L);
string response;
ACSEDGEcode status_code = conn.imagePull("demoimage","v2","admin","pass",response);
```

6.5 Container Deployment Example

After creating an ACSEdgeConn instance using the guidelines from Establishing Connection to the ACS Edge Server, You can begin deploying your container into the ACS Edge Server. The full procedure for how to deploy a container into the ACS Edge Server is provided with examples below. In the examples, dummy data is used for the parameters.

1. Pull an image from the container registry.

```
ACSEdgeConn& conn = ACSEdgeConn::getInstance();
conn.setRegHost("hub.advantest.com");
string image_name = "smart-image";
string tag = "latest";
string username = "smartguy";
string password = "YouNeverForget"
string manifest_file_path = "./manifest.jwt";

string response;
// pull an image from cloud container registry
conn.setRegHost("hub.advantest.com");
ACSEDGEcode  status_code = conn.imagePull(image_name,tag,username, password,response);
// pull an image from cloud container registry with Manifest
//ACSEDGEcode  status_code = conn.imagePull(image_name,tag,username,password,
//                                            manifest_file_path,response);
// pull an image from mirror container registry
// conn.setRegHost(nullptr);
//ACSEDGEcode  status_code = conn.imagePull(image_name,tag, response);
// pull an image from mirror container registry with Manifest
//ACSEDGEcode  status_code = conn.imagePull(image_name,tag,manifest_file_path,response);

if (status_code == 0){
    cout << response << endl;
}
else{
    cout << "Error code: " << (int)status_code << " at " <<__FILE__ << endl;
    cout << response << endl;
}
```

In the above example, it is assumed you are using a V93K RH7 system and creating an ACSEdgeConn instance without specifying the ACS Edge Server IP address. If using a different test system or if using a cloud instance, add the IP parameters into the ACSEdgeConn instance.

The container registry URL must be specified by using `setRegHost()` so that the ACS Edge Server knows which container registry it will communicate with. Afterward, you can call the `imagePull()` function by passing image_name, tag, username, password and the response reference into it. If there is a mirror server used on your test floor, the username and password are not needed.

If the ACS Edge Server is activated with the Application Authorization feature, the `imagePull` function needs another parameter to specify the manifest file path. The manifest file provides the list of images that are authorized to be pulled from the registry and executed by the ACS Edge Server. If the user is not authorized to pull the image or the image is not in the manifest file, the `imagePull()` function will return an error.

The `imagePull()` will return the status code and response string from the ACS Edge Server for this communication. If everything is good, after the `imagePull()` function, the ACS Edge server pulls an image called smart-image:latest from the ACS Container Hub ("registry.advantest.com").

2. Create and start a container.

The next step is to create a container based on the image that was just pulled. There are several options you can use to configure your container. You can add options in the ContainerConfig Object. For example, you can add gpu capability, expose ports on containers, and mount docker volumes with the container. The ContainerConfig Object is fed into `containerCreate()` or `containerCreateStart()` functions. For configuration options and functions, refer to the ACS Edge Container Configuration.

The example below shows the full procedure for how to configure the container by publishing the ports on the container. The host controller can communicate with the container via these port directly by using the same protocol.

```
string container_name = "smart-container";
ContainerConfig contConfig;
vector<string> ports;
// Publish port 8080 with tcp protocol on container/Edge Server to host controller,
// host controller can communicate with the container via port 8080 directly by using
any tcp protocol.
ports.push_back("8080/tcp:8080");
// Mount a docker volume with the container, the volume will be created in the
container's filesystem.
vector<string> volumes;
volumes.push_back("vol1");
// Add gpu capability to the container. The container will be able to use the gpu
device.
contConfig.setOption("gpu","true").setOption("publish-
ports",ports).setOption("volume_attach",volumes).setOption("hostname","demo");
status_code =
conn.containerCreateStart(image_name,container_name,contConfig,response);
string EdgeIp;
status_code = conn.getIp(EdgeIp);

// Using Curl http post to the container's port 8080 as an example.
// But don't have to use http method and libcurl to build the connection,
// Use your tcp Application-layer protocols to communicate with the container
directly,
// It depends on the interface of your app in the container.
string url = "http://" + EdgeIp + ":8080/" + container_name;
CURL *curl;
CURLcode res;

/* In windows, this will init the winsock stuff */
curl_global_init(CURL_GLOBAL_ALL);

/* get a curl handle */
curl = curl_easy_init();
if(curl) {
    /* First set the URL that is about to receive our POST. This URL can
    just as well be a https:// URL if that is what should receive the
    data. */
    curl_easy_setopt(curl, CURLOPT_URL, url.c_str();
    /* Now specify the POST data */
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, "name=daniel&project=curl");

    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);
    /* Check for errors */
    if(res != CURLE_OK)
    fprintf(stderr, "curl_easy_perform() failed: %s\n",
            curl_easy_strerror(res));

    /* always cleanup */
    curl_easy_cleanup(curl);
}
curl_global_cleanup();


if (status_code == 0){
    cout << response << endl;
}
else{
    cout << "Error code: " << (int)status_code << " at " <<__FILE__ << endl;
    cout << response << endl;
}
```

This next example shows the full procedure for how to configure the container by exposing the ports on the container. Unlike the first example, in the below example the container will not be able to communicate with the host controller directly. You will have to use Advantest http protocol to communicate with the container.

```
string container_name = "smart-container";
ContainerConfig contConfig;
vector<string> ports;
// Publish port 8080 with tcp protocol on container/Edge Server to host controller,
// host controller can communicate with the container via port 8080 directly by using
any tcp protocol.
ports.push_back("8001");
ports.push_back("8002");
// Mount a docker volume with the container, the volume will be created in the
container's filesystem.
vector<string> volumes;
volumes.push_back("vol1");
// Add gpu capability to the container. The container will be able to use the gpu
device.
contConfig.setOption("gpu","true").setOption("ports",ports).setOption("volume_attach"
,volumes).setOption("hostname","demo");
status_code =
conn.containerCreateStart(image_name,container_name,contConfig,response);
// Two endpoints are generated for the ports exposed on the container.
status_code = conn.post(container_name + "-8001","init",response);
status_code = conn.post(container_name + "-8002","predict",response);

if (status_code == 0){
    cout << response << endl;
}
else{
    cout << "Error code: " << (int)status_code << " at " <<__FILE__ << endl;
    cout << response << endl;
    }
```

3. Communicate with a container.

It is assumed that a container has successfully been created and started in the ACS Edge Server, and the container is running an http server which is listening from client. If your application is an ACS Edge compatible ML application, we provide two functions: `modelInit()` and `modelPredict()`. If the application is fully user-customized, Advantest provides general HTTP method wrapper functions: `post()` and `get()`. For additional information on these functions, refer to the C++ Client API Reference.

In the below example, the `post()` function is used.

```
response.clear();
string path = "v2/models/MNIST/infer";
string data = "{\"id\":\"1\",\"inputs\":[{\"name\":\"input\",\"shape\":[1,2,28],"
                "\"datatype\":\"UINT8\",\"data\":"
                "[[0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0],"
                "[0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0]] }],"
                "\"outputs\":[{\"name\":\"output\"}] }";
                "\"outputs\":[{\"name\":\"output\"}] }";
status_code = conn.post(container_name,path,data,response);

if (status_code == 0){
    cout << response << endl;
}
else{
    cout << "Error code: " << (int)status_code << " at " <<__FILE__ << endl;
    cout << response << endl;
}
```

4. Purge container and image.

The container and image created contain the customer's IP or data. Due to security concerns, it is recommended that you purge the container and image if you do not plan to use the container or image anymore, even though a high security connection is provided between the client and the ACS Edge Server. Purging non-used containers and images is simply a good practice.

There are several functions for purging a container and image. You can call `containerStop()`, `containerDelete()` and `imageDelete()` to delete the container and image. Or you can use just one function, `containerPurge()`, to delete the container and its associated image. In the example below, `containerPurge()` is used.

```
response.clear();
status_code = conn.containerPurge(container_name,response);

if (status_code == 0){
    cout << response << endl;
}
else{
    cout << "Error code: " << (int)status_code << " at " <<__FILE__ << endl;
    cout << response << endl;
}
```

7. ACS RTDI Troubleshooting

This troubleshooting section is intended to address possible hardware and software connectivity, set-up, and configuration issues that may be encountered during system integration and through daily operation. This guide is not all-inclusive and may not cover unique issues that arise due to a specific set of circumstances.

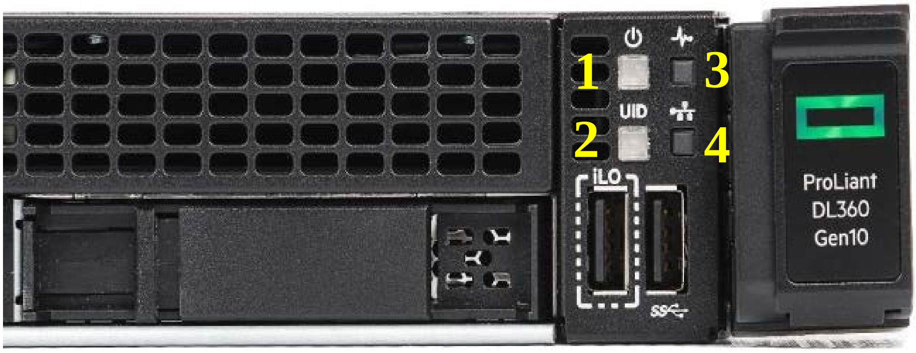
Use the links below to jump to the desired platform for troubleshooting issues.

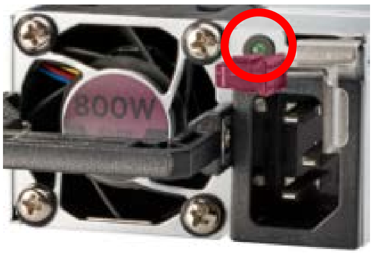
- ACS Edge Server Troubleshooting
- ACS Unified Server Troubleshooting
- ACS Container Hub Troubleshooting

7.1 ACS Edge Server Troubleshooting

If the Edge Server faces IP connectivity issues with the Host Controller, follow the troubleshooting suggestions recommended in the table below in the order they appear. To resolve the issue, it may be necessary to perform one or more of the specified suggestions.

If the problem persists after performing all the suggested troubleshooting actions indicated in the list below, contact your Advantest representative.

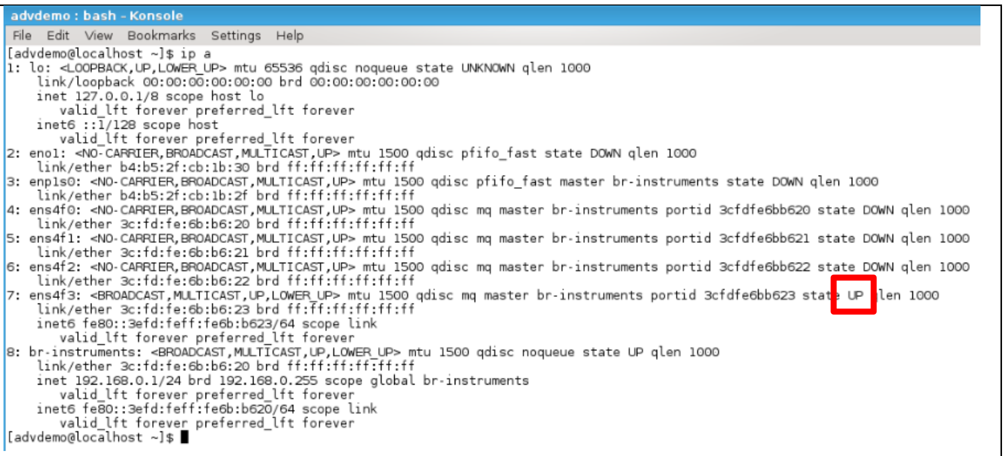
Suggestions	Procedure
Check Physical Setup	1. Check if the PDU power cord is plugged into the designated wall outlet. 2. Open the mini-rack back panel and check if all ethernet and power cables are securely connected. 3. Confirm that the Ethernet cable is connected between the Host Controller and the Edge Server.
Confirm Power	1. Confirm that the wall outlet is outputting the required power. 2. Check if mini-rack fan is powered ON. The fan should always be ON, as there is no on/off switch for mini-rack fan. 3. Check if the Edger Server is powered ON. 4. Check if the Host Controller is powered ON.
Manually Power ON Server	The Edge Server is configured to power ON and boot up automatically when connected to a power source. If the server does not automatically start, it can be manually powered ON as outlined below. 1. Use the mini-rack key to open the front panel. 2. Remove front bezel on the mini-rack. 3. Press Power On button (#1 in the image below) on Edge Server.  1 2 3 4 UID iLO SS ProLiant DL360 Gen10 4. Allow 2 minutes for the server to complete the bootup sequence. 5. After the server is fully booted, check the state of the LED indicators and observe that they appear as indicated in the following table.

<table>
<tr>
<td></td>
<td>
<table>
<tr><th>#</th><th>LED Indicator</th><th>Desired State</th><th>Description</th></tr>
<tr><td>1</td><td>Power Button and LED</td><td>Solid Green</td><td>System ON</td></tr>
<tr><td>2</td><td>UID Button and LED</td><td>OFF</td><td>UID is deactivated and iLO Service Port is ready for use</td></tr>
<tr><td>3</td><td>Health LED</td><td>Solid Green</td><td>Normal</td></tr>
<tr><td>4</td><td>NIC status LED</td><td>Flashing Green</td><td>Network active</td></tr>
</table>
6. If the server does not power ON, check Edge Server Power Supply module.
</td>
</tr>
<tr>
<td>Check Power Supply Module</td>
<td>
If the Edge Server will not power on automatically or manually, check the power supply module.
1. Use the mini-rack key to open the back panel.
2. Verify the Edge Server power cable is connected securely to Edge Server and PDU.
3. Verify the PDU power cable is connected securely to the PDU and wall outlet.
4. Verify the wall outlet has power.
5. Locate the power supply at the rear of the Edge Server (usually located on right side when observing from rear).
6. Verify that the power LED on the power supply is illuminated (solid green) while the power cable is connected and the power is ON. See image below.


7. If the power LED on the power supply is not illuminated:
a. Check for any loose power cables and secure if necessary.
b. Check for proper supply voltage and current from wall outlet.
c. Plug another device into the grounded power outlet to be sure the outlet works.
NOTE: Ensure that the power source meets applicable standards.
d. Replace the power cord with a known functional power cord to be sure it is not faulty.
e. Disconnect the power cable from the power supply for 3 minutes, then reconnect it back.
</td>
</tr>
<tr>
<td>Reboot System Setup</td>
<td>
To reboot the system setup, follow the steps below:
1. Verify that the Ethernet cable is connected between Host Controller and Edge Server.
2. Reboot the Host Controller.
3. Log in to the Host Controller and wait 2 minutes.
</td>
</tr>
</table>

	4. Reboot the Edge Server and wait 3 minutes.
Check Physical Ethernet Connectivity	To check physical Ethernet connectivity between the Host Controller and the Edge Server, follow the steps below: 1. Verify the Host Controller and the Edge Server are powered ON. 2. Verify the Ethernet cable is connected between the Host Controller and the Edge Server. 3. Check the rear of the Edge Server and verify that the P1 Ethernet port LEDs are illuminated. a. If the P1 Ethernet port LEDs are illuminated, perform steps listed in Reboot System Setup. 4. If the P1 Ethernet port LEDs are not illuminated, do the following: *Check for loose cable connections* a. Check each end of the Ethernet Cable to identify if there is a loose Ethernet cable connection. *Reconnect cables* a. Disconnect and reconnect the Ethernet cable from the P1 port of the Edge Server. b. Disconnect and reconnect the Ethernet cable from the NIC on the Host Controller. *Try different Ethernet ports* a. Disconnect the Ethernet cable from the P1 port of the Edge Server and connect that same Ethernet cable into the P2 port on Edge Server. b. Disconnect the Ethernet cable from port #4 of the NIC on the Host Controller and connect that same Ethernet cable into port #3 of NIC on the Host Controller. *Check Ethernet cable for issues* a. Disconnect the Ethernet cable from the P1 port of the Edge Server and connect that same Ethernet cable into another Ethernet port-enabled device to see if the other Ethernet port LEDs illuminate or not. This is to confirm the problem is within the Edge Server rather than within the Host Controller. *Replace Ethernet cable* a. If the above steps determine a faulty Ethernet cable, replace the faulty Ethernet cable with a known functional Ethernet cable.
Check the NIC Status using 'ip a' Command	1. On the Host Workstation, issue the below command on the terminal (Konsole): `$ ip a` 2. Check the state of the NIC Card #4 port and ensure the state is "UP," as identified in the reference example below.

<table>
<tr>
<td></td>
<td>

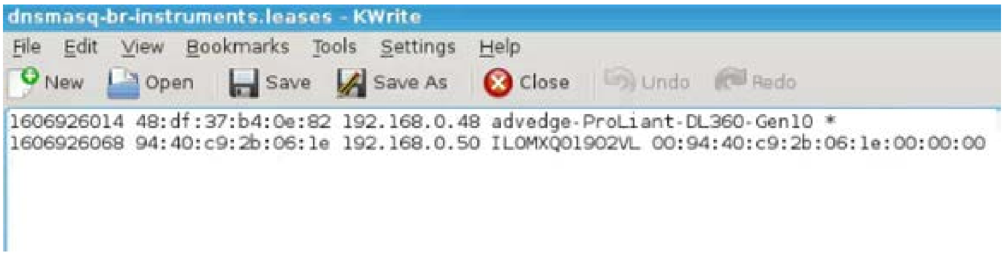


```
[advdemo@localhost ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether b4:b5:2f:cb:1b:30 brd ff:ff:ff:ff:ff:ff
3: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master br-instruments state DOWN qlen 1000
    link/ether b4:b5:2f:cb:1b:2f brd ff:ff:ff:ff:ff:ff
4: ens4f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-instruments portid 3cfdfe6bb620 state DOWN qlen 1000
    link/ether 3c:fd:fe:6b:b6:20 brd ff:ff:ff:ff:ff:ff
5: ens4f1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-instruments portid 3cfdfe6bb621 state DOWN qlen 1000
    link/ether 3c:fd:fe:6b:b6:21 brd ff:ff:ff:ff:ff:ff
6: ens4f2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master br-instruments portid 3cfdfe6bb622 state DOWN qlen 1000
    link/ether 3c:fd:fe:6b:b6:22 brd ff:ff:ff:ff:ff:ff
7: ens4f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br-instruments portid 3cfdfe6bb623 state UP qlen 1000
    link/ether 3c:fd:fe:6b:b6:23 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::3efd:feff:fe6b:b623/64 scope link
       valid_lft forever preferred_lft forever
8: br-instruments: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP qlen 1000
    link/ether 3c:fd:fe:6b:b6:20 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.1/24 brd 192.168.0.255 scope global br-instruments
       valid_lft forever preferred_lft forever
    inet6 fe80::3efd:feff:fe6b:b620/64 scope link
       valid_lft forever preferred_lft forever
[advdemo@localhost ~]$ █
```

</td>
</tr>
<tr>
<td>Check the Edge Server IP Address</td>
<td>

1. On the Host Controller, open the following file:

   `/var/run/dnsmasq-br-instruments.leases.txt`



dnsmasq-br-instruments.leases - KWrite
File Edit View Bookmarks Tools Settings Help
New Open Save Save As Close Undo Redo

```
1606926014 48:df:37:b4:0e:82 192.168.0.48 advedge-ProLiant-DL360-Gen10 *
1606926068 94:40:c9:2b:06:1e 192.168.0.50 ILOMXQ01902VL 00:94:40:c9:2b:06:1e:00:00:00
```

2. If a line item exists with "advedge-ProLiant-DL360-Gen10" with an associated IP address, the Edge Server is detected and ready to use.
3. If a line item does not exist with "advedge-ProLiant-DL360-Gen10," follow Reboot System Setup procedure and check again.

</td>
</tr>
<tr>
<td>Replace the Ethernet Cable</td>
<td>

To replace the Ethernet cable, follow the steps below:

NOTE: It is recommended that this procedure is performed by Advantest personnel.

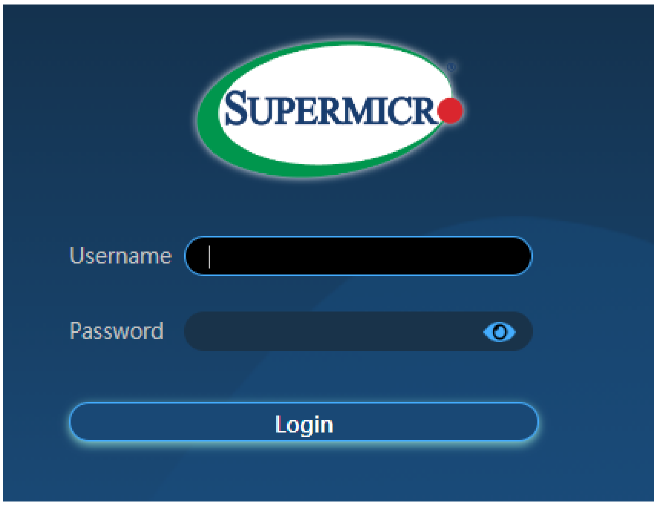
1. Disconnect the Ethernet cable from the Host Controller.
2. Open the mini-rack back panel and disconnect the Ethernet cable from the Edge Server.
3. Replace the Ethernet cable with a new cable (Cat6A ONLY).
4. Connect the new cable to the Edge Server and route it out of the mini-rack.
5. Connect other end of the cable to the Host Controller.
6. Secure the cable properly between the Host Controller and Edge Server.
7. Reboot the Edge Server and wait 3 minutes.
8. For connectivity, verify the lights on the Edge Server Ethernet port and on the NIC card at the rear of the Host Controller.
9. Close mini-rack back panel using the key.

</td>
</tr>
</table>

7.2 ACS Unified Server Troubleshooting

If the Unified Server is not responding, verify whether the server is accessible using the BMC (IPMI) interface before contacting your Advantest representative. The BMC web interface can be accessed through a browser by entering the BMC IP address into the URL. Make sure that the web browser is running from a machine within the same network as the target server's BMC interface.

If the BMC login page loads, contact your Advantest representative for remote support, as access to BMC is only allowed by Advantest personnel.



If the BMC login page does not load, follow the procedure below.

Suggestions	Procedure
Check BMC IP Address	1. Connect a keyboard, mouse, and monitor to the target server. 2. Power cycle the server. During boot, press the <DEL> key to access the system BIOS. 3. Navigate to the BMC tab. 4. Confirm that "Configuration Address source" is set to [Static], and that the BMC IP address being used for access validation matches the BMC IP address found from the system BIOS. 5. If the Configuration Address Source is not set to [Static], set it to [Static] and assign an IP address that is available for use.
Complete Shutdown	1. Power OFF the server. 2. Unplug the power cord for 5 minutes to completely reset the BMC. 3. Reconnect the power cord to AC power and power ON the server. 4. Attempt to access the BMC login page.

7.3 ACS Container Hub Troubleshooting

The main issues that may be encountered when connecting to the ACS Container Hub are related to IP Whitelisting and debugging Docker pull. These issues are addressed below.

7.3.1 IP Whitelisting

To pull images from ACS Container Hub, a set of IP addresses and whole ranges need to be whitelisted for HTTPS (port 443) communication.

All Docker Registry HTTP API V2 requests are initially served by Container Hub Registry API endpoints. This includes returning the image manifests. For image layer downloads, the Container Hub Registry instead generates pre-signed URLs to the associated S3 objects and responds with HTTP 307 to advise clients to download BLOB data directly from S3, effectively bypassing the proxy.

Container Hub IP Addresses		
Container Registry API Endpoints	AWS S3 Endpoints in Region us-west-2 (Oregon)	Container Hub Web Portal
52.12.197.227	3.5.76.0/22	35.163.233.31
54.214.73.203	18.34.244.0/22	54.188.252.161
52.38.194.140	18.34.48.0/20	
50.112.153.207	3.5.80.0/21	
35.162.83.151	52.218.128.0/17	
52.39.85.26	52.92.128.0/17	
	35.80.36.208/28	
	35.80.36.224/28	

To obtain the most recent list of IP ranges, go to:

https://aws.amazon.com/premiumsupport/knowledge-center/s3-find-ip-address-ranges/

As a shortcut, enter the following command in Konsole to pull the IP ranges:

```
$ curl -s https://ip-ranges.amazonaws.com/ip-ranges.json | jq -r '.prefixes[] |
select(.region=="us-west-2") | select(.service=="S3"
) | .ip_prefix'
```

NOTE: The above command will automatically download the JSON file that contains all AWS IP address ranges.

You can use the jq tool to parse the JSON file. This tool can be downloaded from https://jqlang.github.io/jq/download/.

Appendix 1. Python Logger, Result, and Event

This section describes the code for generating logs, events, and results in a Python-based application using json format. To install the package, in ACS Edge, you can find the AdvantestLogger.py from the following directory:

```
oneAPI_py/bin/
```

Prior to utilizing the package, there is a prerequisite to execute the following in the above directory:

```
python -m pip install -r requirement.txt
```

NOTE: The 'EDGE_ID', 'TESTER_ID', 'LOG_FILE_PATH', 'RESULT_FILE_PATH' are defined in the environment variable of the container created by the Edge Server.

A1.1 Logger

Log is in the following json format:

```
{"testerId": "env_tester_id", "edgeId": "env_edge_id", "applicationId": "custom_app",
 "timestamp": "1694791374", "level": "INFO", "message": "Log Initializing"}
```

To get the logger, you must first initialize an instance from Advantestlogger class with an instance name and app name. Two optional parameters are `tester_id` and `edge_id`. If you provide these optional parameters, the output will contain the provided optional values. Otherwise, these optional values will be retrieved from the environmental variable.

```
logger = AdvantestLogger("my_app", app="custom_app",
edge_id="custom_edge_id"(optional), tester_id="custom_tester_id"(optional))
```

After initializing the instance, you can use the log functions with the message you want to generate the log for (info, debug, error, etc.) in the python logging class.

```
logger.info("Log Initializing")
```

The log is stored in the filepath which is specified in the environment variable 'LOG_FILE_PATH'.

A1.2 Result

Result is in the following json format:

```
{"testerId": "v93k-1", "edgeId": "Edge-MXQ02006VM", "applicationId": "DPAT",
 "timestamp": "1694701266", "key": "new_upper_limit", "value": 10.0}
```

To get the result, you must first initialize an instance from AdvantestResult class with an instance name and app name. Two optional parameters are `tester_id` and `edge_id`. If you provide these optional parameters, the output will contain the provided optional values. Otherwise, these optional values will be retrieved from the environmental variable. If the container is created by the Edge Server, the two environment variables will be available.

```
result = ResultLoggerWrapper("result_logger", app="DPAT", edge_id="Edge-
MXQ02006VM"(optional), tester_id="v93k-1"(optional))
```

After initializing the instance, you can use a function called result`()` with the message you want and the value of the result (which is very similar to the `info()` or `error()` function in the python logging class) to get the result message. The result is stored in the filepath which is specified in the environment variable 'RESULT_FILE_PATH'.

A1.3 Event

Event is in the following json format:

```
{"testerId": "custom_tester_id", "edgeId": "custom_edge_id", "applicationId":
 "custom_app", "timestamp": "1694791374", "eventType": "INFO", "eventName": "event
 ends"}
```

To get the event, you must first initialize an instance from AdvantestEvent class with an instance name and app name. Two optional parameters are `tester_id` and `edge_id`. If you provide these optional parameters, the output will contain the provided optional values. Otherwise, these optional values will be retrieved from the environmental variable.

```
event = AdvantestEvent("my_app", app="custom_app",
edge_id="custom_edge_id"(optional), tester_id="custom_tester_id"(optional))
```

After initializing the instance, you can use a function called `event()` with the message you want (which is very similar to the `info()` or `error()` function in the python logging class) to get the event message. You can also add custom labels to add to the event logging.

Example 1

```
event.event("eventName","event Initializing")
```

Example 2

```
custom_labels = {"image_url": "https://registry.advantest.com/image/version"}
event.event("eventName","image downloaded", custom_labels=custom_labels)
```

For Example 2, the output would be:

```
{"testerId": "custom_tester_id", "edgeId": "custom_edge_id", "applicationId":
"custom_app", "timestamp": "1694701769", "eventType": "INFO", "eventName": "image
downloaded", "image_url": "https://registry.advantest.com/image/version"}
```

The event is stored in the filepath which is specified in the environment variable.

Appendix 2. Application Descriptor Command Line Tool

Application Descriptors contain information for ACS Nexus to auto-deploy applications onto the ACS Edge Server without the need to modify the test program. They are required for running ACS Nexus in auto-deploy mode. Application Descriptors encode which container images must be pulled and started as containers. They can also include additional configuration options per container, such as environment variables or monitoring configuration properties.

Application Descriptors are managed on ACS Container Hub alongside container images. In production, Application Descriptors and container images are replicated from ACS Container Hub to the ACS Unified Server, from where they are provided locally to test cells that have ACS Nexus and ACS Edge enabled.

Technically, Application Descriptors are machine-readable files stored in JSON format. These descriptors are conventionally created and managed using the ACS Container Hub web interface. However, `acs-application` is a command line tool that provides an alternative for managing Application Descriptors outside of the ACS Container Hub.

A2.1 acs-application Command Line Tool Requirements and Path

System Requirement:

- Linux on x86_64

The acs-application command line tool is distributed as a single, self-contained executable binary file. It is installed on the workstation which is used for developing and building the container images. Request the latest version from your Advantest technical contact and copy it to a location in your `PATH`.

```
$ sudo cp ./acs-application /usr/local/bin/acs-application
```

A2.2 Application Descriptor Format

To manage Application Descriptors using the acs-application command line interface, users will need to work with the JSON format directly. Below is the Application Descriptor schema (V1) presented within the JSON schema specification.

```
{
  // REQUIRED: Application Descriptor format version. Must be "1".
  "version": "1",
  // REQUIRED: Application Descriptor name. Customer-defined, used for display and logging.
  "name": "My Application XYZ v2",
  // REQUIRED: Unique information to resolve this application descriptor before lot start
  "selector": {
    // REQUIRED: Device name as defined by test program
    "device_name": "my_device_folder_name",
    // REQUIRED: product family as defined by test program or "*" to match any
    "product_family": "*",
    // REQUIRED: test program name as defined by test program or "*" to match any
    "test_program_name": "*",
    // REQUIRED: test program revision as defined by test program or "*" to match any
    "test_program_revision": "*"
  },
  // REQUIRED: Describes runtime configuration for ACS Edge
  "edge": {
    // REQUIRED: List of containers to be started. Exactly 1 container is supported.
    "containers": [
      {
        // REQUIRED: Name of container on ACS Edge
        "name": "mycontainer",
        // REQUIRED: Image reference; local to registry.advantest.com
        "image": "myorg-myproject/myrepo:v1",
        // OPTIONAL: ACS Edge server requirements
        "requirements": {
          // OPTIONAL: Should GPU be enabled for this container? Default: false
          "gpu": false
        },
        // OPTIONAL: Environment variables that are passed into the container
        "environment": {
          "VARNAME": "value"
        },
        // OPTIONAL: Must be set if Prometheus scrape endpoint has different port and/or path
than default
        "metrics": {
          // OPTIONAL: Port to scrape metrics. Default: 9001
          "port": 9001,
          // OPTIONAL: Path to scrape metrics at. Default: /metrics
          "path": "/metrics"
        }
      }
    ]
  }
}
```

A2.2.1 Application Descriptor Selector

The selector of any Application Descriptor is a crucial element. It defines how the right Application Descriptor is determined by ACS RTDI when running on a test floor.

The selector attribute `device_name` is a mandatory selector attribute and must have a concrete value that is exactly matched during production. All other attributes such as `test_program_name`, `product_family` and `test_program_revision` may be set to a wildcard value (`*`), which will match any concrete value coming from the test cell in production.

Every selector attribute combination can only exist once among all application descriptors. An `acs-application create` command or `acs-application update` command that would result in a duplicate selector is not allowed and will result in an error.

CAUTION: Make sure that your released Test Program and Application Descriptor make a perfect match. Otherwise, it is technically possible to create application descriptors with different selectors that lead to ambiguous matches during Application Descriptor selection in production. Take the following example for 2 application descriptors:

Descriptor #1 Selector	Descriptor #2 Selector
Device name: my_device	Device name: my_device
Product family: my_product	Product family: ANY
Test program name: TestProgram_1	Test program name: TestProgram_1
Test program revision: ANY	Test program revision: v1

In the above scenario, the test program in production would emit the following attributes:

- Device name: my_device
- Product family: my_product
- Test program name: TestProgram_1
- Test program revision: v1

In this example scenario, both descriptors match, because each selector matches the device name and 2 other attributes. This is an error condition in production, and a proper descriptor selection cannot be made. To avoid this type of condition, specify as many selector attributes as possible in the application descriptor. If a descriptor shall be useable for multiple products and/or test programs, make sure to not create an ambiguous situation as presented in the above example.

The acs-application command-line interface reads application descriptors from JSON files. When creating a new descriptor, create a new JSON file first, such as "appdesc-my_application.json." The next page illustrates an example for creating a new descriptor.

Application Descriptor Example

```
{
  "version": "1",
  "name": "Test",
  "selector": {
    "device_name": "my_device_folder_name",
    "product_family": "*",
    "test_program_name": "*",
    "test_program_revision": "*"
  },
  "edge": {
    "containers": [
      {
        "name": "app",
        "image": "customer-myproject/app:latest",
        "requirements": {
          "gpu": true
        },
        "environment": {
          "VARNAME": "value"
        },
        "metrics": {
          "port": 1234,
          "path": "/prometheus"
        }
      }
    ]
  }
}
```

A2.3 acs-application Command Line Tool General Information

The table below contains a list of available commands used with the acs-application command line tool. To enter a command, enter the following in the command line:

```
acs-application [command]
```

Commands and Flags	Description
`create`	Create an application descriptor on ACS Container Hub.
`delete`	Delete application descriptor(s) on ACS Container Hub.
`get`	Get application descriptor contents from ACS Container Hub.
`help`	Receive help about any command.
`list`	List application descriptors on ACS Container Hub.
`query`	Query application descriptor on ACS Container Hub.
`update`	Update application descriptor on ACS Container Hub.
`validate`	Validate application descriptor JSON files.
`version`	Print the current version of the acs-application tool. This is helpful for troubleshooting and when contacting Advantest for support if acs-application issues are encountered.
`--debug`	Outputs verbose log messages.
`-h / --help`	Receive help for acs-application tool. Enter "`acs-application [command] --help`" for more information regarding usage and flags for a command.
`--trace`	Outputs extremely verbose log messages.
`-v / --verbose`	Outputs log messages.

A2.3.1 Authentication

The acs-application tool commands that work with the ACS Container Hub require an Organization ID and according authentication information. Most commonly, a username/password combination is required to authenticate at a registry. If the acs-application tool is used directly with the mirror container registry on the ACS Unified Server, an additional client certificate is required to perform mTLS (mutual Transport Layer Security) authentication.

Tool Option	Description
`-o / --org-id`	The ID of your ACS Container Hub Organization. You find this in the Customer Portal user menu (see Figure 3-7). as the prefix for all of your projects and client credentials. For example, if your Organization is "Example Corp.," when you log in on the ACS Container Hub web interface, you find a list of projects such as: • exa • exa-app-1 • exa-app-2 In this case, "exa" is your Organization ID. If you work against a local registry, this must be the namespace (project name) where Application Descriptors are stored.
`-u / --username`	Your myAdvantest email address (for example "john.doe@example.com") or the name of a client credential (for example "access+exa-mycredential").
`-p / --password`	Your Docker secret or the secret of a client credential. You can leave this parameter out if you do not want to reveal your password in your shell history. In that case acs-application will prompt for the password, similar as the docker login command does.

The credentials described in the table below may be used with the ACS Container Hub. To create an Application Descriptor, you must be an Organization Administrator and must use your personal account with Docker secret.

Type	Source	Required Permissions
Docker secret	myAdvantest email address / Docker secret copied from the ACS Container Hub Customer Portal. (These are the same credentials used for docker login.)	Organization Admin
Client Credential	Name/secret of a Container Hub Client Credential. Works only with read commands (list, get, query).	Must have pull permission for the [organization-id] project

A2.3.2 Trusted TLS Certificates

By default, the acs-application command line tool uses HTTPS connections to connect to container registries. It verifies the server certificate and only establishes a connection if the server certificate is trustworthy. For verification, the tool relies on the trusted certificates that are stored on the operating system.

If you require to connect to a registry with a self-signed or otherwise untrusted certificate, you must import the server certificates into the trusted certificate list of your operating system to connect to container registries.

A2.4 Command Usage and Examples

This section describes the available acs-application tool commands and provides examples.

A2.4.1 List Command

The `list` command lists metadata for all Application Descriptors. Metadata includes ID, name, selector, and additional attributes. Command output is in JSON format per Application Descriptor.

List Command Example

```
$ acs-application list -o exa -u john.doe@example.com
Password: [... paste or type secret ...]
{
  "id": "06a643ff-7948-4c24-b9ec-8a2290a385d6",
  "name": "My Application 1",
  "creator": "john.doe@example.com",
  "creation_date": 1695198731522,
  "last_update_date": 1695198731522,
  "selector": {
    "device_name": "my_device_name",
    "product_family": "*",
    "test_program_name": "*",
    "test_program_revision": "*"
  }
}
```

A2.4.2 Get Command

The `get` command retrieves a specific Application Descriptor. To get a specific Application Descriptor you must first know its ID. You can get the ID using the `list` or `query` command. The ID is also printed for every newly created Application Descriptor. Command output is the full Application Descriptor JSON.

Get Command Example

```
$ acs-application get -o exa -u john.doe@example.com 06a643ff-7948-4c24-b9ec-
8a2290a385d6
Password: [... paste or type secret ...]
{
  "version": "1",
  "name": "My Application 1",
  "selector": {
    "device_name": "my_device_name",
    "product_family": "*",
    "test_program_name": "*",
    "test_program_revision": "*"
  },
  "edge": {
    "containers": [
      {
        "name": "app-container-1",
        "image": "exa-app-1/edge-app:v1.0",
        "requirements": {}
      }
    ]
  }
}
```

A2.4.3 Query Command

The `query` command simulates the Application Descriptor lookup during production test. You must know the exact selector attributes that are going to be sent by ACS Nexus to find the Application Descriptor for the current test program. Command output is the Application Descriptor ID.

Query Command Example

```
$ acs-application query -o exa -u john.doe@example.com device_name=my_device_name
Password: [... paste or type secret ...]
Result: 06a643ff-7948-4c24-b9ec-8a2290a385d6
```

You can also request the full JSON by specifying the `--print-json` option:

```
$ acs-application query -o exa -u john.doe@example.com --print-json
device_name=my_device_name
Password: [... paste or type secret ...]
Result: 06a643ff-7948-4c24-b9ec-8a2290a385d6

{
  "version": "1",
  "name": "My Application 1",
  "selector": {
    "device_name": "my_device_name",
    "product_family": "*",
    "test_program_name": "*",
    "test_program_revision": "*"
  },
  "edge": {
    "containers": [
      {
        "name": "app-container-1",
        "image": "exa-app-1/edge-app:v1.0",
        "requirements": {}
      }
    ]
  }
}
```

A2.4.4 Validate Command

Creating Application Descriptors manually can be error-prone. You can use the `validate` command to validate your JSON file for syntactical and semantic correctness before creating it.

If the file is valid, the command output is: `OK: <Application Descriptor Name>`

Validate Command Example (valid file)

```
$ acs-application validate ./path/to/my-appdesc.json
OK: Test
```

If the file is invalid, the validation error is printed. For the example below, an invalid selector is detected:

Validate Command Example (invalid file)

```
$ acs-application validate ./path/to/my-appdesc.json
Error: validation failed - [invalid_descriptor] missing or empty selector:
"device_name"
```

A2.4.5 Create Command

The `create` command creates a new Application Descriptor. New Application Descriptors are created from JSON files that need to be prepared upfront. Command output is the ID of the newly created descriptor.

Create Command Example

```
$ acs-application create -o exa -u john.doe@example.com ./path/to/my-appdesc.json
Password: [... paste or type secret ...]
Created: 06a643ff-7948-4c24-b9ec-8a2290a385d6
```

A2.4.6 Delete Command

The `delete` command removes obsolete Application Descriptors from the ACS Container Hub using the descriptor ID.

Delete Command Example

```
$ acs-application delete -o exa -u john.doe@example.com 06a643ff-7948-4c24-b9ec-
8a2290a385d6
Password: [... paste or type secret ...]
Deleted: 06a643ff-7948-4c24-b9ec-8a2290a385d6
```

A2.4.7 Update Command

The `update` command is used to modify existing Application Descriptors. For example, the command can be used to change the descriptor's selector attributes, update the container image tag, or specify additional container environment variables. Typically, you would start by retrieving the desired Application Descriptor and storing it as a JSON file (as shown in Example 1), which you then modify (as shown in Example 2).

Update Command Example 1

```
$ acs-application get -o exa -u john.doe@example.com --output-file ./path/to/my-appdesc.json 06a643ff-7948-4c24-b9ec-8a2290a385d6

Password: [... paste or type secret ...]

Written: ./path/to/my-appdesc.json
```

Then modify `my-appdesc.json` and update it on the ACS Container Hub:

Update Command Example 2

```
$ acs-application update -o exa -u john.doe@example.com 06a643ff-7948-4c24-b9ec-8a2290a385d6 ./path/to/my-appdesc.json

Password: [... paste or type secret ...]

Updated: 06a643ff-7948-4c24-b9ec-8a2290a385d6
```

A2.5 Exit Codes

When using the acs-application command line tool, the command-line interface returns different exit codes depending on the success of an operation or the category of an error. The table below lists and describes the exit codes used.

Exit Code	Description
`0`	Operation successful.
`1`	There was a user error. For example, invalid or missing option or argument.
`2`	A system error occurred. For example, a network communication error with the container registry.
`101`	Validation error. A provided Application Descriptor is invalid.
`102`	Registry error. The container registry returned an error. For example, invalid credentials were used.

Appendix 3. Azure Hosted Container Registry

To support customers who prefer services to be hosted on Azure Cloud, a Container Registry is set up on Azure. The location of the container registry is harbor.az.unified.advantest.com.

The sample docker CLI commands are listed below:

Docker Login

```
harbor.az.unified.advantest.com --username username@advantest.com
```

Docker Build and Push

```
docker build:  --tag harbor.az.unified.advantest.com/advtesting-azure-project1/advtesting-azure-repository:latest
docker push:   harbor.az.unified.advantest.com/advtesting-azure-project1/advtesting-azure-repository:latest
```

Docker Tag and Push

```
docker tag:    localImageName1 harbor.az.unified.advantest.com/advtesting-azure-project1/advtesting-azure-
               repository:latest

docker push:  harbor.az.unified.advantest.com/advtesting-azure-project1/advtesting-azure-repository:latest
```

Docker Pull

```
harbor.az.unified.advantest.com/advtesting-azure-project1/advtesting-azure-repository:latest
```