Vantage
Application Note

Vantage SDK REST Integration Guide for Java

For use with Vantage 6.3 or later

Version 1.1

160924

May 2015

Synopsis

The Vantage SDK is designed to enable third-party programs to submit jobs to Vantage workflows, monitor and manage jobs, and access job's metadata and their files. The Vantage SDK is language independent and can be implemented using REST, SOAP or WCF protocols. SDK examples are provided in Java and C#.

This application note describes how to integrate a Java program with Vantage, using a RESTful Web Service API—via HTTP Get and Post protocols. How to perform common Vantage tasks are described.

Telestream suggests that you use this app note in the following manner:

1. Read the first four topics, meeting all of the pre-requisites.
   - Vantage Version and Licensing Requirements
   - Obtaining the Vantage SDK
   - Example Java Rest Program and Workflow
   - Using the Vantage REST Utility
2. Next, read Getting Started With the Vantage SDK—create your first job submission program and get it working.
3. Use the remaining topics to learn how to perform other common Vantage tasks.

Note: This guide is written for software engineers who have a working knowledge of Java development, their development environment, and XML. You should also be familiar with using Vantage or have access to someone who is. To use Vantage effectively via the Vantage SDK, you should know how to create and manage workflows and submit jobs using Vantage Workflow Designer. If you aren't familiar with Vantage, we suggest that you review the Vantage User's Guide.

Vantage Version and Licensing Requirements

This app note is written for Vantage customers using Vantage version 6.3 or later. You should have Vantage 6.3 or later installed and operational before implementing programs using the Vantage SDK.

There are no special licensing requirements when integrating third-party programs with Vantage via the Vantage SDK. You can programmatically perform any task permitted by the Vantage license you have installed.

Please contact your Telestream representative or Telestream Customer Service if you have Vantage licensing questions.

Obtaining the Vantage SDK

To obtain the Vantage SDK, follow these steps:

1. Log in to the Telestream Web site (www.telestream.net).
2. Navigate to the Vantage Customer Center to download the Vantage SDK zip file.
3. On the Customer Center page, locate the Software Development Kit (SDK) link in the left column, near the bottom of the page.
4. Click the Software Development Kit (SDK) link.
5. Acknowledge agreement to the SDK End User License agreement.
6. Select the Vantage SDK.
7. Answer the remaining questions regarding your relationship with Telestream, and your interest in a support contract.
8. Click Download Now.
9. Expand the zip file on your computer.

Note: There are no installations required to utilize the SDK.

What's in the SDK

The Vantage SDK contains code examples, a Vantage SDK REST utility, and SDK documents.

1. *Samples*—The SDK contains several example programs in C# and in Java, using WCF, SOAP and REST. Telestream recommends that you study the examples, and use them as the basis for custom programs you plan to develop.

- *Vantage SDK REST Utility*—This utility is a developer tool which enables you to execute Get and Post methods interactively and view the XML results. It is also the definitive list of all REST methods in the Vantage API. It speeds development by enabling you to work with Vantage dynamically, walking through a series of methods to accomplish a task.
- *Vantage 6 SDK Overview*—Telestream recommends that you read this PDF to understand the goals of the SDK, the architecture of Vantage, and the Vantage class structure. You'll also learn about which classes in the Vantage namespace are used to work with actions, jobs, variables, and binders.
- *Vantage Action SDK PDF*—The Action SDK is an extension of the Vantage SDK. It is designed to provide maximum capability and flexibility for clients interested in using the Vantage Flip Transcoder without having to construct or manage a typical Vantage workflow. The

Action SDK, like the traditional Vantage SDK may be accessed using a variety of client protocols: WCF, Soap Web Services, or REST.

- *Vantage SDK Documentation*—This CHM identifies and describes each of the classes in the Vantage SDK namespace. These classes are built on WCF; SOAP implementations are fully-described. For REST, the Get methods are described, but the Post methods are not.

Note: For a complete list of all REST methods in the Vantage API, use the Vantage REST utility.

Example Java Rest Program and Workflow

The Vantage SDK also contains a sample Java REST program which illustrates each of the tasks described in this app note. The program file is named *VantageJobComplete.java*, and you can find it in the *\Samples\RestSamples\JavaRestAppNoteSample* folder.

The accompanying *Java Rest App Note Sample* workflow is located in the *\Samples\JavaRestAppNoteWorkflow* folder.

This sample program uses a specific workflow, Before you can run the sample Java program, you should import the workflow into your Vantage domain and make any configuration changes required (such as changing storage locations) to use it. Make sure the target workflow is activated when running SDK programs.

Using the Vantage REST Utility

The Vantage REST Utility is a Windows program that enables you to use the Get and Post methods in the Vantage SDK interactively.

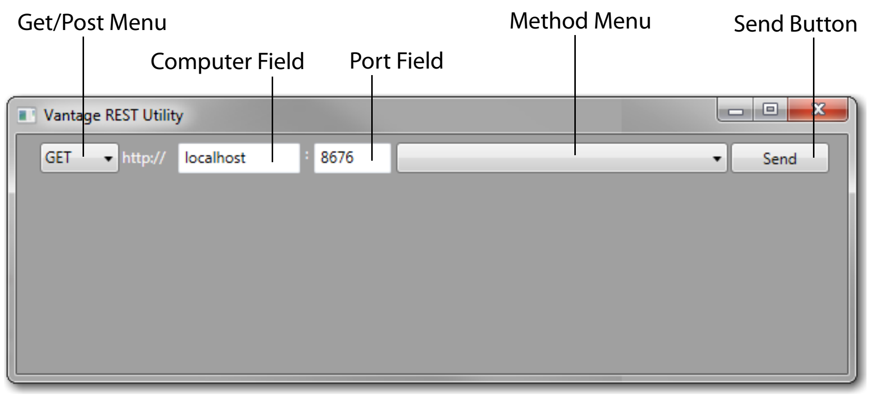
The REST Utility improves REST development by enabling you to dynamically select and execute REST methods in your Vantage domain. It displays all of the methods in the API, and provides a user interface for entering parameter values and reviewing the XML results returned.

Starting and Configuring the REST Utility

To start the Vantage REST Utility, double-click the RestUtility.exe file.

When it starts, the utility displays this window:

Figure 1. Vantage REST Utility Window



To configure the utility for use in your environment, you update the computer name and port as needed.

If you installed the client on a computer other than the Vantage domain sever, enter the computer name of your Vantage domain server (or Vantage database server in an array) in the computer name field.

If you have altered the port number that the Vantage SDK service uses, enter it in the port field.

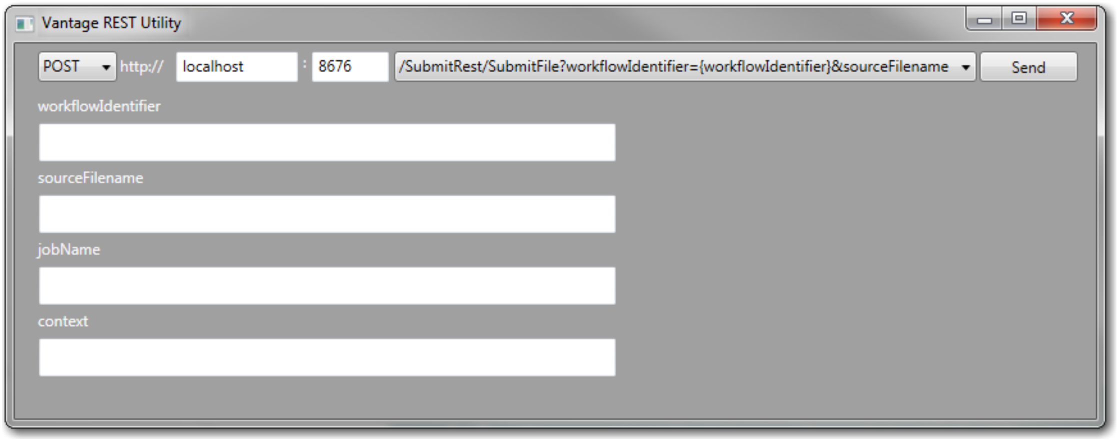
Using the Utility to Execute Methods

For each command you want to execute, follow these steps:

1. Select Get | Post from the menu.
2. Select the method to execute from the Method menu. When you select a method, the method's parameter fields are displayed.
3. Fill out the required parameter values.
4. Click the Send button.

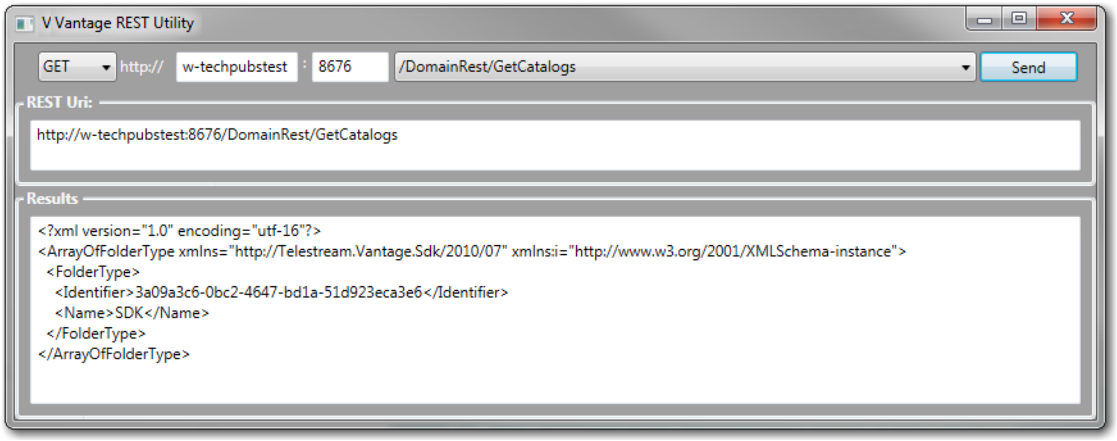
Here's an example of the window with the *SubmitFile* method selected, showing the four parameters you need to supply for this particular method:

Figure 2. REST Utility Method Selection and Parameter Entry



When you execute a method, the query string and results (or error) are displayed:

Figure 3. REST Utility Method Execution Results



http://w-techpubstest:8676/DomainRest/GetCatalogs

```
<?xml version="1.0" encoding="utf-16"?>
<ArrayOfFolderType xmlns="http://Telestream.Vantage.Sdk/2010/07" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <FolderType>
    <Identifier>3a09a3c6-0bc2-4647-bd1a-51d923eca3e6</Identifier>
    <Name>SDK</Name>
  </FolderType>
</ArrayOfFolderType>
```

The Results field displays the XML returned from the method. You can review the XML, and browse through it to identify GUIDs (or other values) you need to execute the next call in a logical sequence. For example, you can execute *GetWorkflows*, and then browse the *ArrayOfProcedure* XML to identify the workflow you want to use, and copy its GUID for the *SubmitFile* method.

Getting Started With the Vantage SDK

With your Vantage domain operational and the Vantage SDK downloaded, you're ready to get started.

Preliminary tasks include making sure you have a set of Java programming tools, reviewing the SDK overview, and creating a simple workflow or importing the sample workflow provided.

1. If you don't have a Java source code editor or IDE installed, select one and install it (for example, Eclipse Kepler, Net Beans, IntelliJ IDEA, etc.).
2. Make sure that a Java Development Kit (for example, Java Standard Edition (SE)) is installed.
3. Read the *Vantage 6 SDK Overview* in the Vantage SDK folder.
4. In Vantage Workflow Designer, create a simple workflow—a Receive action and a Flip action—for initial testing and development. Or, import the sample workflow provided in the Vantage SDK. See Example Java Rest Program and Workflow.
5. Configure the actions as necessary, and activate the workflow.

Note: All workflows intended for processing media being submitted from a Vantage SDK-based program must start with a Receive action.

6. Proceed to Creating a Simple Job Submission Program to create your first integration program.

Creating a Simple Job Submission Program

The simple Java program included here provides a basic framework for interacting with Vantage and submitting a job to a workflow in Vantage.

This sample program performs the following functions:

- Configures a URI to communicate with a specific workflow in a Vantage domain
- Creates an HTTP Client connection
- Executes an HTTP Post to submit the job, as specified in the URI
- Receives the HTTP response
- Closes the connection.

Here are steps to create a Java project that submits a job to Vantage:

1. Creating a Sample Job Submit Program and Project
2. Updating the Program for Your Domain
3. Submitting the Job to Your Vantage Workflow

Creating a Sample Job Submit Program and Project

To create a sample Vantage job submit project, follow these steps:

1. Create a new Java project.
2. Copy the example Java program code below, and add it to your project.

```
//package org.vantage.RestfulClient.webservices;
//Classes for exception processing
import java.io.IOException;
import java.net.URISyntaxException;
import org.apache.http.client.ClientProtocolException;
//Class for HTTP processing
import java.net.URI;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.utils.URIBuilder;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;

//Main Job Submit class
public class VantageJobSubmit {
  String result = null;
  public static void main(String[] args) throws ClientProtocolException,
IOException, URISyntaxException {
    /* Create a Job Submit URI for the Post -
     * host name of Vantage domain server (or database server if array)
as IP address|HostName|'localhost' + SDK Service port (default 8676)
     * Path for SubmitFile method
     * Workflow Identifier GUID
     * Source media - full path to input media. Must be accessible to
Vantage service executing the workflow's Receive action
     * Context - input variables for the job, if any
     * Job name
     */
     // Use URIBuilder to generate the URI for Submit File Rest method
     // Example URI: http://VantageServer:8676/SubmitRest/SubmitFile
            ?workflowIdentifier=2234955-4b36-b5fe-c508-e8294312
            &sourceFilename=%5C%5CShare%5Cmedia.mpg&jobName=Job+Submit
    URI uri = new URIBuilder()
    .setScheme("http")
    .setHost("<VantageServer>:8676")
    .setPath("/SubmitRest/SubmitFile")
    .setParameter("workflowIdentifier", "xxxxx-xxxx-xxx-xxxx-xxxxxxxxx")
    .setParameter("sourceFilename","full path to media with double \\")
    .setParameter("jobName","name of this job")
    .build();
     System.out.println("submitJob URI: " + uri);
    //Execute the job submit method
    submitJob(uri);
  }
  private static String submitJob(URI uri) throws IOException {
    String result = null;
    // create an HTTP client
    CloseableHttpClient httpClient = HttpClients.createDefault();
    try { // set up a Post
       HttpPost httpPost = new HttpPost(uri);
       // execute the post
       CloseableHttpResponse response = httpClient.execute(httpPost);
    }
    catch(Exception e) {}
    //release the connection and return
    httpClient.close();
    return result;
  }
}
```

The Vantage *SubmitFile* method submits a new job using the file referenced by *sourceFilename* to the workflow specified by *workflowIdentifier*.

Exceptions for this method:

- *WorkflowDoesNotExistException*—a workflow with the specified identifier does not exist.
- *WorkflowInvalidStateException*—the workflow with the specified identifier is not running.

The *submitJob* method creates an HTTP client connection, executes an HTTP Post, receives the response, and closes the connection.

Note: In this sample program and others in the Vantage SDK—other than the Vantage SDK methods—all code is just an example. The code in these programs is often the simplest, but not the best, safest, or most efficient way to perform a given task, all issues being considered. These programs are provided for tutorial purposes only; you should never use these programs in production, and Telestream disclaims any responsibility in doing so.

Updating the Program for Your Domain

To update the sample program to work with your domain, you should modify the URI code to identify your Vantage domain, identify a short media file, and specify a Vantage workflow you've imported or created.

The main class constructs the URI using *URIBuilder*, and then executes the *submitJob* method.

```
    URI uri = new URIBuilder()
    .setScheme("http")
    .setHost("VantageDomainServer:8676")
    .setPath("/SubmitRest/SubmitFile")
    .setParameter("workflowIdentifier", "295045-4b76-b5fe-c508-e83012")
    .setParameter("sourceFilename","\\\\VantageServer\\src\\media.mpg")
    .setParameter("jobName","Job Submit Test")
    .build();
```

The result of *URIBuilder* is a properly formed, UTF-8-encoded URI:

```
http://VantageDomainServer:8676/SubmitRest/SubmitFile
?workflowIdentifier=295045-4b76-b5fe-c508-e83012
&sourceFilename=%5C%5CVantageServer%5Csrc%media.mpg
&jobName=Job+Submit+Test
```

To update the *URIBuilder* method to submit a job to your domain, specify the following information in the URI:

- The name of the host Vantage domain server—see Specifying the Vantage Domain Server.
- The workflow that you are submitting the job to—see Providing the Workflow Identifier.
- The full path to the media to process—see Specifying the Fully-qualified Path to Media.
- The name of the job you are submitting—see Providing a Job Name.

Specifying the Vantage Domain Server

The Vantage domain server identity can be the computer name, the IP address of the computer, or *localhost* when your program is running on the same computer (which probably is not the norm or the ideal situation).

In an All-in-One domain, the host name of the server on which Vantage is running. However, in a Vantage Array (multiple servers, each running various Vantage services and a Microsoft SQL Server database), the Vantage server to target is the one running the SDK Service.

The host name is specified as a URL, including the Vantage SDK Service port (default 8676).

Examples:

```
http://localhost:8676
http://Vantage_Domain_Seven:8676
http://017.121.001:8676
```

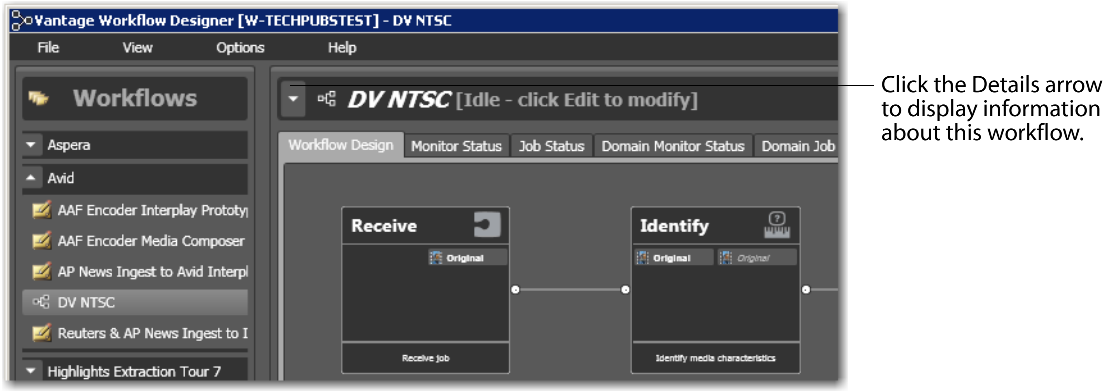
Providing the Workflow Identifier

The workflow identifier is a GUID. When you import or create a workflow (workflows are always created in Workflow Designer), Workflow Designer automatically assigns a GUID. You can obtain a workflow's GUID using the Vantage SDK REST method */DomainRest/GetWorkflows*. You can also obtain it directly in Workflow Designer.

To obtain a workflow's GUID for use in your program, follow these steps:

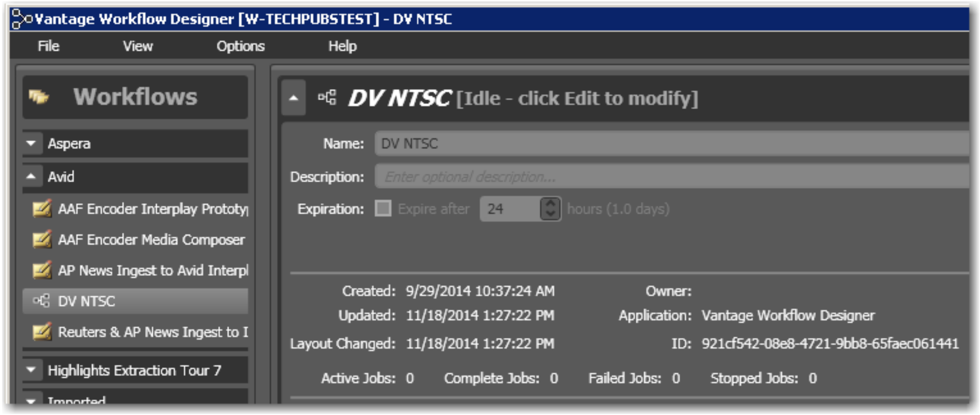
1. Open Workflow Designer.
2. Display the workflow you are targeting, as shown in the figure following:

Figure 4. Workflow in Workflow Designer


Vantage Workflow Designer [W-TECHPUBSTEST] - DV NTSC
File
View
Options
Help
Workflows
Aspera
Avid
AAF Encoder Interplay Prototy
AAF Encoder Media Composer
AP News Ingest to Avid Interpl
DV NTSC
Reuters & AP News Ingest to I
Highlights Extraction Tour 7
DV NTSC [Idle - click Edit to modify]
Workflow Design
Monitor Status
Job Status
Domain Monitor Status
Domain Job
Receive
Original
Receive job
Identify
Original
Original
Identify media characteristics
Click the Details arrow to display information about this workflow.

3. Click the details arrow just to the left of the workflow's title, to display the workflow details panel, as shown in the figure following:

Figure 5. Displaying Workflow Details (Including GUID)


Vantage Workflow Designer [W-TECHPUBSTEST] - DV NTSC
File
View
Options
Help
Workflows
Aspera
Avid
AAF Encoder Interplay Prototy
AAF Encoder Media Composer
AP News Ingest to Avid Interpl
DV NTSC
Reuters & AP News Ingest to I
Highlights Extraction Tour 7
Imported
DV NTSC [Idle - click Edit to modify]

Name: DV NTSC
Description: Enter optional description...
Expiration: Expire after 24 hours (1.0 days)

Created: 9/29/2014 10:37:24 AM
Owner:
Updated: 11/18/2014 1:27:22 PM
Application: Vantage Workflow Designer
Layout Changed: 11/18/2014 1:27:22 PM
ID: 921cf542-08e8-4721-9bb8-65faec061441
Active Jobs: 0 Complete Jobs: 0 Failed Jobs: 0 Stopped Jobs: 0

4. Copy the GUID, and paste it into your code.

Specifying the Fully-qualified Path to Media

The path to the media being submitted must be fully-qualified (and accessible to Vantage services). If the media is on the same server as all of the services that must access it, you can use a drive-letter path. However, in almost all circumstances, it is always safer to place the media on a share, which is accessible by every server in the Vantage domain.

Note: Back-slashes are doubled—they are escape characters in the context of a Java string.

Examples:

```
C:\\Media\\Ginger_Ale_Project\\Sierra_snow.mpg
\\\\Medusa_Server\\Ginger_Ale_Project\\Sierra_snow.mpg
```

Providing a Job Name

Create a practical job name, which should be unique for every job. The job name is displayed in the job status tabs in Workflow Designer, during and after job execution.

Submitting the Job to Your Vantage Workflow

To submit a job using your program, follow these steps:

1. In Workflow Designer, make sure that your workflow is active.
2. Run your Job Submit program (determining that it executed successfully).
3. In Workflow Designer, select the target workflow and display the Job Status tab to view your job running to completion.

Submitting a Job with Input Variables

When you submit a job to a workflow, you can optionally provide values for variables defined in the workflow actions, to control the job at run time.

Note: Only variables that are already bound to a parameter in an action can be provided with a value during job submission. If you don't supply bind a variable to a parameter, the variable will not be included in the context XML file.

Note: This task is implemented in the *JavaRestAppNoteSample* program directly in main. It executes the *submitJob* method as well.

The general process for submitting variables with runtime values involves these tasks:

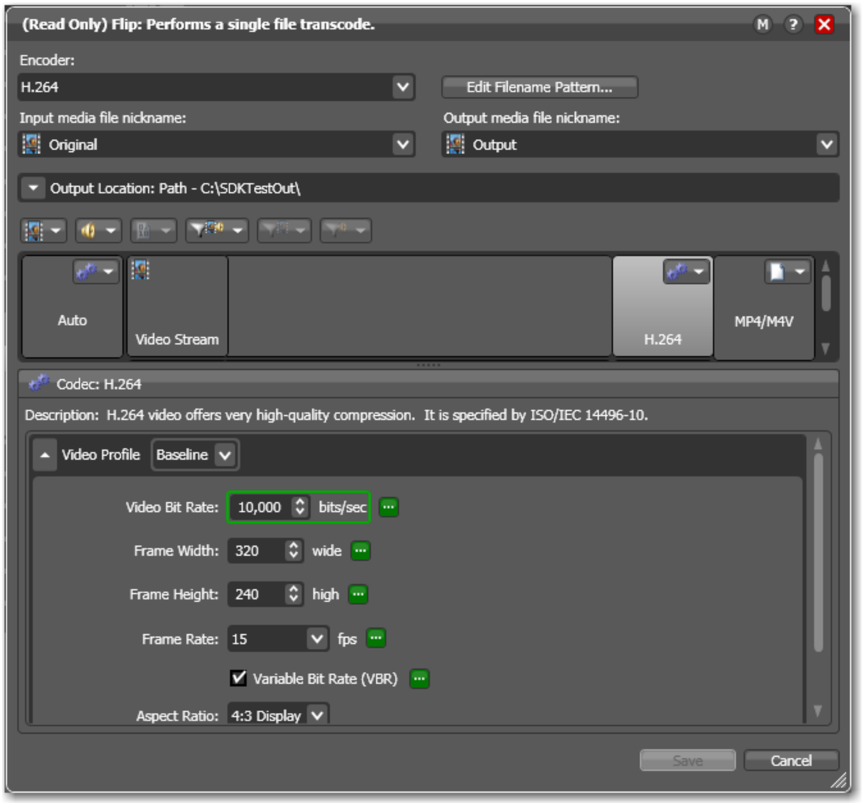
- Adding Variables to a Workflow
- Obtaining the Input Variables in a Workflow
- Updating the Variable Values
- Including Variables When You Submit a Job
- Verifying that Variables were Submitted Properly

Adding Variables to a Workflow

To add a variable to a workflow and bind it to an action's parameter, follow these steps:

1. Open Workflow Designer and select (or create) the workflow.
2. In the action where you want to use the variable, display the Inspector.
3. Identify the parameter and click the green Browse button, as shown in the image below (shown completed—the value field (in this example, Video Bit Rate) is bordered in green).

Figure 6. Binding a Variable to a Parameter


(Read Only) Flip: Performs a single file transcode.

Encoder: H.264

Input media file nickname: Original

Output media file nickname: Output

Output Location: Path - C:\SDKTestOut\

Codec: H.264

Description: H.264 video offers very high-quality compression. It is specified by ISO/IEC 14496-10.

Video Profile: Baseline

Video Bit Rate: 10,000 bits/sec

Frame Width: 320 wide

Frame Height: 240 high

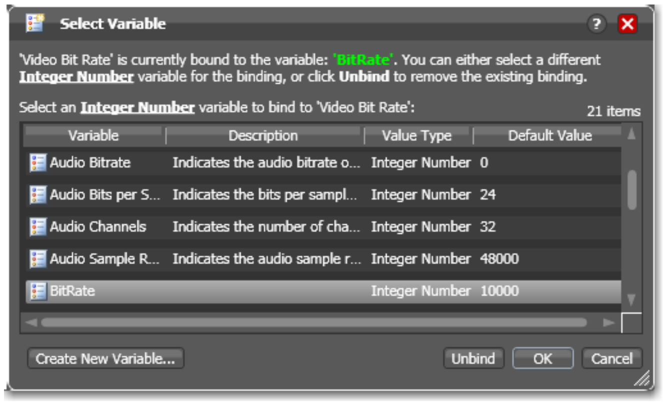
Frame Rate: 15 fps

☑ Variable Bit Rate (VBR)

Aspect Ratio: 4:3 Display

Workflow Designer displays the Select Variable dialog where you can browse and select a variable or create a new one to use.

Figure 7. Selecting a Variable


Select Variable

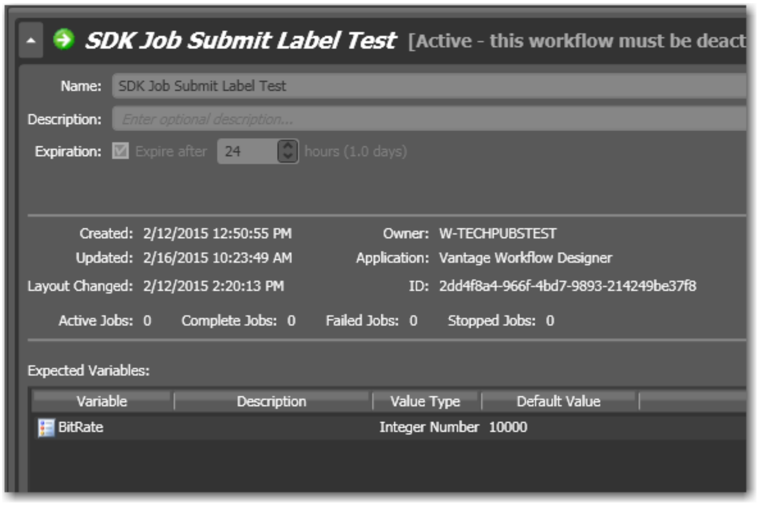
'Video Bit Rate' is currently bound to the variable: 'BitRate'. You can either select a different **Integer Number** variable for the binding, or click **Unbind** to remove the existing binding.

Select an **Integer Number** variable to bind to 'Video Bit Rate': 21 items

Variable	Description	Value Type	Default Value
Audio Bitrate	Indicates the audio bitrate o...	Integer Number	0
Audio Bits per S...	Indicates the bits per sampl...	Integer Number	24
Audio Channels	Indicates the number of cha...	Integer Number	32
Audio Sample R...	Indicates the audio sample r...	Integer Number	48000
BitRate		Integer Number	10000

1. Select (creating, if necessary) the variable (BitRate in this example) you want to bind to the action's parameter and click OK to bind the variable to the parameter.

Figure 8. Expected Variables in this Workflow


SDK Job Submit Label Test [Active - this workflow must be deact

Name: SDK Job Submit Label Test

Description: Enter optional description...

Expiration: ✓ Expire after 24 hours (1.0 days)

Created: 2/12/2015 12:50:55 PM Owner: W-TECHPUBSTEST

Updated: 2/16/2015 10:23:49 AM Application: Vantage Workflow Designer

Layout Changed: 2/12/2015 2:20:13 PM ID: 2dd4f8a4-966f-4bd7-9893-214249be37f8

Active Jobs: 0 Complete Jobs: 0 Failed Jobs: 0 Stopped Jobs: 0

Expected Variables:

Variable	Description	Value Type	Default Value
BitRate		Integer Number	10000

2. Now, click the details arrow just to the left of the workflow's title to display the workflow details panel, as shown in the figure above.

Note: Notice that Workflow Designer lists the variable (in this case, *BitRate*) as an expected variable, and the same variable is listed in the list of variables, in the topic immediately following.

Obtaining the Input Variables in a Workflow

To obtain the list of variables defined in a workflow, execute GET /WorkflowRest/GetWorkflowVariableRequirements. You'll need to supply the workflow GUID (Providing the Workflow Identifier).

Here's an example URI:

```
http://VantageServer:8676/WorkflowRest/GetWorkflowVariableRequirements
?identifier=914fdeac-1121-42fd-bce0-2f377cd1b382
```

GetWorkflowVariableRequirements returns a *Context* XML. In the context of the Vantage SDK, variables are referred to as *conditions*. The *Context* XML is comprised of each variable defined in the workflow and bound to a parameter in some action.

Updating the Variable Values

To update a variable's value, you'll parse the *Context* XML to identify the variable, and provide the run-time value you want. Here is a snippet of the *Context* XML, showing one variable (presented as a *Condition* element). The variable is *BitRate*, with a default value of `10000`:

```
<Context...>
...
<a:Condition>
  <Identifier>5b795b51-7c32-4e3e-aa8c-0018ca59d7e0</Identifier>
  <Name>BitRate</Name>
  <ParameterSelections/>
  <ParameterSetCollections/>
  <Parameters/>
  <Summary i:nil="true"/>
  <Instance>70d675c0-166c-4097-846c-b37e821e6abe</Instance>
  <ConditionValue>
    <ComplexValue i:nil="true"/>
    <Text i:nil="true" xmlns:b="..."/>
    <Default>
      <ComplexValue i:nil="true"/>
      <Text xmlns:b=...>
        <b:string>10000</b:string>
      </Text>
    </Default>
  </ConditionValue>
  <TypeCode>Int32</TypeCode>
</a:Condition>
...
```

Best practice suggests some form of XML parsing method to identify the variable, then update the value of the variable's default value, in the *<b:string>* element. When you are developing the program, you can execute *GetWorkflowVariableRequirements* and copy the returned *Context* XML into a text editor to view the results. Then, write your code to modify each variable's value as required.

Note: There is no guarantee that the *Context* XML will present the variables in the same order each time.

Including Variables When You Submit a Job

To submit a job to a workflow with run-time variables values, you add the optional *context* parameter and supply the *Context* XML in the job submit method. For example, POST / SubmitRest/SubmitFile.

Here is an example URI with the *context* parameter highlighted in bold—the bulk of the *Context* XML object is deleted for brevity.

```
http://VantageServer:8676/SubmitRest/SubmitFile
?workflowIdentifier=914fdeac-1121-42fd-bce0-2f377cd1b382
&sourceFilename=%5C%5CVantage_Media%5Cmedia%5C720x480.mpg
&context=%3CContext+xmlns%3D%22http%3A%2F%2FTelestream.Vantage.Sdk...
&jobName=Job+Submit+Test
```

SubmitFile (and the other job submit methods) returns a job *guid* XML. Here's an example:

```
<guid xmlns="http://schemas.microsoft.com/2003/10/Serialization/
">ccb1ef3c-15e6-43ea-ac89-c39ea8bf63c3</guid>
```

You should extract the job GUID and save it in a variable, for use when you access the same job to perform other tasks.

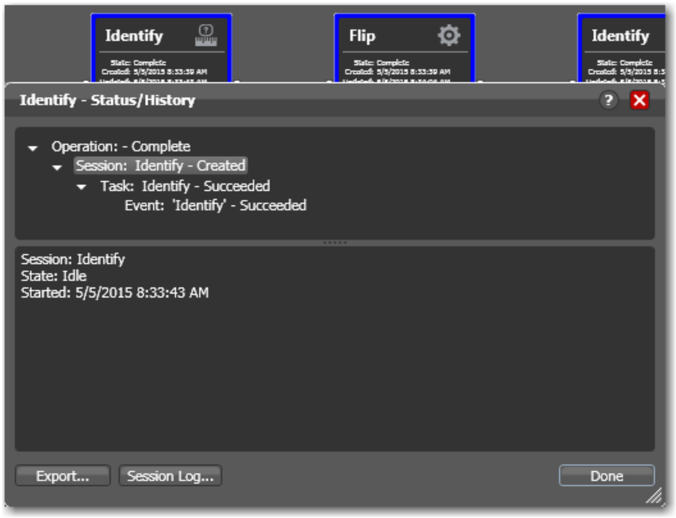
Note: To review other job submission methods, refer to the Vantage SDK library documentation CHM file (see What's in the SDK) and search with `submit`. Or refer to the Vantage REST Utility.

Verifying that Variables were Submitted Properly

To verify that the variables and runtime values were supplied correctly in your job, follow these steps:

1. Submit a job by running your program.
2. In the Workflow Designer's Job Status panel, select the job and watch it execute to completion.
3. Double-click on the 2nd action (the action immediately following the Receive action) in your workflow, to display the Status inspector. (This action is the first action to gain access to your input variables.)

Figure 9. Viewing the Action Inspector in Complete Jobs

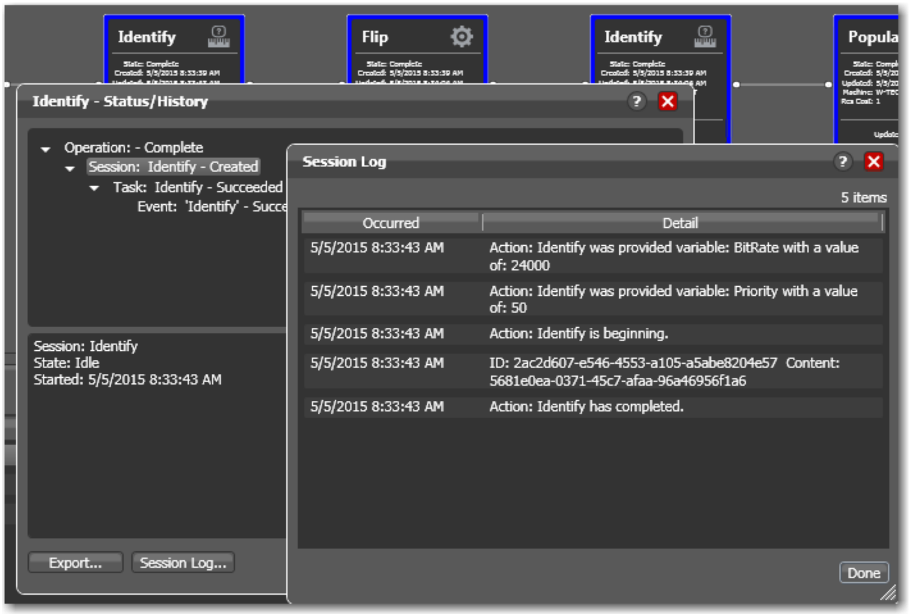


Operation: - Complete
Session: Identify - Created
Task: Identify - Succeeded
Event: 'Identify' - Succeeded

Session: Identify
State: Idle
Started: 5/5/2015 8:33:43 AM

Select the second line (where the session is created), and click Session Log.

Figure 10. Viewing the Action's Session Log to Verify Runtime Variable Values


Identify
Flip
Identify
Popula
Identify - Status/History
Operation: - Complete
Session: Identify - Created
Task: Identify - Succeeded
Event: 'Identify' - Succe
Session Log
5 items
Export...
Session Log...
Done

Session: Identify
State: Idle
Started: 5/5/2015 8:33:43 AM

Occurred	Detail
5/5/2015 8:33:43 AM	Action: Identify was provided variable: BitRate with a value of: 24000
5/5/2015 8:33:43 AM	Action: Identify was provided variable: Priority with a value of: 50
5/5/2015 8:33:43 AM	Action: Identify is beginning.
5/5/2015 8:33:43 AM	ID: 2ac2d607-e546-4553-a105-a5abe8204e57 Content: 5681e0ea-0371-45c7-afaa-96a46956f1a6
5/5/2015 8:33:43 AM	Action: Identify has completed.

Note the two variable events in this example: The BitRate and Priority variables were received, with runtime values of 24000 and 50, respectively.

Note: If the variables and their values do not correspond with the functionality implemented in your program, it needs to be resolved.

Setting the Job Priority During Job Submission

Note: This task is implemented in the *JavaRestAppNoteSample* program directly in main.

You can set the job priority (which is really setting action (*session)* priority on the first action in the job) when you submit a job. Include the optional *context* parameter in the job submission URI, and insert the Priority *condition* element into the context XML.

Note: In the context of the Vantage SDK, a *session* is the execution of an action. Actions exist in the context of a workflow; sessions exist in the context of a job. This is an important distinction.

For details, see Submitting a Job with Input Variables.

Note: Actions which have a Priority variable set in Workflow Designer (you right-click on an action and set the Priority) will always use the explicitly-specified value, whether an upstream Priority is passed from an earlier action in the workflow, or from an SDK submission. The only way to guarantee that the Priority value will be used is to insure that no Priority values are explicitly set.

Insert the following *Condition* element into the *Context* XML, specifying the Priority value you want for the job in the <a:Condition><ConditionValue><Default><Text><b:string> element (set as `50` in this example):

```
<a:Condition>
  <a:xmlns.../>
  <Identifier>FF261686-8408-46b9-B6E7-D447BD5BCD82</Identifier>
  <Name>Priority</Name>
  <ParameterSelections/>
  <ParameterSetCollections/>
  <Parameters/>
  <Summary i:nil="true"/>
  <ConditionValue>
    <ComplexValue i:nil="true"/>
    <Text xmlns:.../>
    <Default>
      <ComplexValue i:nil="true"/>
      <Text xmlns:...>
        <b:string>50</b:string>
      </Text>
    </Default>
  </ConditionValue>
  <TypeCode>Priority</TypeCode>
</a:Condition>
```

One method of inserting this element in the *Context* XML string is to assign this element to a string (escaping the double quote marks), and then replace the end `</Conditions>` tag with the Priority variable string concatenated with the `</Conditions>` tag. Another is to use X Path processing.

Note: You can also explicitly set the session priority (via *SetSessionPriority*) of a running action; this will set or override the priority of the session. This is similar to setting the priority on a running or waiting to run action directly in Workflow Designer.

Obtaining the Workflows in a Domain

You may want to design a program that provides a list of workflows in a Vantage domain, so that a user can select which workflow they want to work with.

To obtain the list of workflows in a given Vantage domain, execute GET /DomainRest/GetWorkflows. Here's an example URI:

```
http://VantageServer:8676/DomainRest/GetWorkflows
```

There are no parameters. Just provide the name of your Vantage domain server (Specifying the Vantage Domain Server).

GetWorkflows returns a Procedures XML. In the context of the SDK, workflows are referred to as *procedures*. When the Procedures XML is returned, Typically, you'll design the method to parse it and dynamically build a list of workflows (each identified as a Procedure element in the XML) from which the user can select.

Here's a snippet from a *Procedures* XML to illustrate the instance of a workflow:

```
<Procedure>
  <xmlns.../>
...
  <Identifier>b552d26e-fb18-464a-be8a-02ce12da2ead</Identifier>
  <Conditions xmlns:a=... />
  <Description />
  <LockState>Unlocked</LockState>
  <Name>workorder conform proxy</Name>
</Procedure>
...
```

Obtaining the Job GUID

When you submit a file for processing, *SubmitFile* (and other job submission methods) returns the *guid* XML. Here's an example:

```
<guid xmlns="http://schemas.microsoft.com/2003/10/Serialization/">
ccb1ef3c-15e6-43ea-ac89-c39ea8bf63c3</guid>
```

You'll need to extract the job GUID as a string, for use when you access this job in your programs for any reason. You can use XML parsing or string processing on this simple XML.

Determining Job Status

Note: This task is implemented in the *JavaRestAppNoteSample* program in *waitForJobToComplete*.

A common requirement of job submission programs is to determine the status of the job.

You'll need to know when the job is complete to retrieve end-of-job variable values, or access job media or metadata labels. Or, you may be reporting job status on one or more jobs to another monitoring system or a job management database.

To obtain the status of a job, you execute GET /JobRest/GetJobState. You'll need to supply the job GUID, as explained below. Here's an example of the URI:

```
http://VantageServer:8676/JobRest/GetJobState
?identifier=99632a8e-d719-400e-8cf8-7a7ca2cd2f1b
```

Typically, you'll design the method that executes *GetJobState* with a do-while loop until it determines that the job is complete, as specified by the reserved string *Complete*. Then, you can continue your final job processing.

Obtaining the Job State

To obtain the transaction state of the job, execute GET /JobRest/GetJobState, supplying the job's GUID in the *identifier* parameter.

Here is an example of the URI:

```
http://VantageDomain:8676/JobRest/GetJobState
?identifier=ccb1ef3c-15e6-43ea-ac89-c39ea8bf63c3
```

Here is the *WorkflowJobState* XML returned from *GetJobState*. These three examples illustrate various job states: *Waiting*, *Active*, and *Complete*.

```
<WorkflowJobState xmlns="http://Telestream.Vantage.Sdk/2010/
07">Waiting</WorkflowJobState>
<WorkflowJobState xmlns="http://Telestream.Vantage.Sdk/2010/07">Active</
WorkflowJobState>
<WorkflowJobState xmlns="http://Telestream.Vantage.Sdk/2010/
07">Complete</WorkflowJobState>
```

This the list of job state values that can be returned: *Active* | *Failed* | *StoppedByUser* | *Complete* | *WaitingToRetry*.

Retrieving Variable Values From Jobs

Note: This task is implemented in the *JavaRestAppNoteSample* program in *retrieveTimeCodeVariable*.

When a job is complete, you can obtain the final value of variables that were used in the job. The most effective strategy is to target a specific target that you know—by workflow design—will have the correct value at job's end, and extract the target variable's value from it.

Note: This is only one method of accessing runtime values from a job. The other method is to use labels, which are generated in a workflow by using a Populate action, where you assign variables to a given parameter in the label you specify (see Accessing Variable Values from Labels in Jobs).

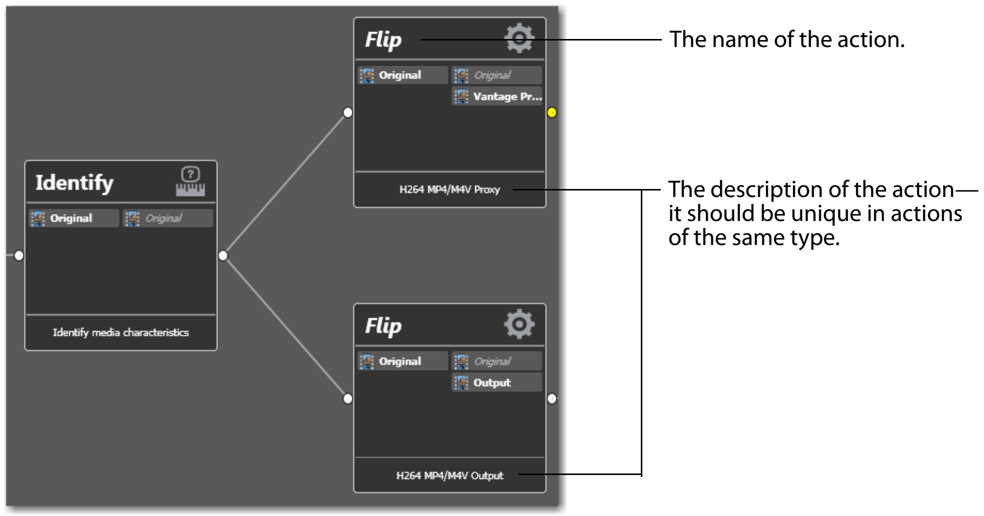
In straight-line (non-branching) workflows, you can obtain all variables from the last session that executes. However, in branching workflows, the last session that executes is non-determinant. Its best to target a specific action that you know will always execute last.

To process variables, you need to know the action and the variables you want to obtain values from. In Workflow Designer, collect these two pieces of information:

- The name or definition of the target action.
- The name variable that you are targeting. You can view the variables that comprise a work-flow by displaying the workflow in Workflow Designer and selecting the Variables display option (at the bottom right corner of the workflow design panel).

Here's how to identify the action in Workflow Designer. The name and description are displayed directly on the action in a workflow:

Figure 11. Identifying the Name and Description of Actions in a Workflow



The name of the action is displayed at the top of each action. The description of the action is displayed at the bottom of the action. To modify the default description, double-click the text and type in the new description to make it unique. To make it easy to use, keep it short.

Note: In the context of the Vantage SDK, a *session* is the execution of an action. Actions exist in the context of a workflow; sessions exist in the context of a job. This is an important distinction.

Extracting variable values from a job involves these general tasks, which are described in detail in the topics following:

1. Using the job GUID (see Obtaining the Job GUID, described previously), obtain the list of sessions—action execution instances—that comprise the job. See Obtaining the Sessions List.
2. Using the action's name or description, extract it's session GUID from the list of sessions. See Extracting the Target Session's GUID.
3. Using the session GUID, obtain the session details. See Obtaining the Session Details.
4. Extract the variable's value (and other information you need) from the session details. See Obtaining the Variable's Value.

Obtaining the Sessions List

To obtain the list of sessions in a job, execute GET /WorkflowRest/GetSessionsForJob. You need to provide the GUID of the particular job (see Obtaining the Job GUID).

Here's an example:

```
http://VantageServer:8676/JobRest/GetSessionsForJob
?identifier=ccb1ef3c-15e6-43ea-ac89-c39ea8bf63c3
```

The returned XML document is an *ArrayOfSessionType*. It is comprised of a series of *SessionType* elements—one for the execution of each action in the workflow. You can parse it to identify the target session and extract its GUID.

Here's a snippet from an *ArrayOfSessionType* XML with only a single *SessionType* (instance of action execution) to illustrate what `GetSessionsForJob` returns:

```
<ArrayOfSessionType xmlns=...>
  <SessionType>
    <Identifier>32ab0163-fb11-4aab-8f78-647f2c90c6a4</Identifier>
    <Description>DurationVal</Description>
    <Name>Copy</Name>
    <Started>2015-01-27T07:36:16.49</Started>
    <State>Complete</State>
    <Updated>2015-01-27T07:36:47.55</Updated>
  </SessionType>
...
<ArrayOfSessionType>
```

Bear in mind that all jobs will have at least two sessions, because a valid workflow must contain at least 2 actions. Jobs will have a session for every action that executed in the job.

Extracting the Target Session's GUID

Using the name or description of the target action, you can identify its session in the *ArrayOfSessionType* XML that was returned and extract it for use in the next task. The session name is specified in the Name element; the description is in the *Description* element.

Parse the *ArrayOfSessionType* XML to identify the target session and extract its GUID from the *Identifier* element.

Here is a sample X Path expression that selects the target session's *Description* node by the string value of its *Description* element:

```
//Identifier[ancestor::SessionType/Description[text()='DurationVal']].
```

Obtaining the Session Details

Using the session's GUID, you can obtain the context—the details of a given session—by executing GET /JobRest/GetContentsForJob, using the session GUID.

Here's an example:

```
http://VantageServer:8676/SessionRest/GetSessionContext
?identifier=32ab0163-fb11-4aab-8f78-647f2c90c6a4
```

This returns a *Context* XML—the same XML returned when you execute GET /WorkflowRest/GetWorkflowVariableRequirements for submitting input variable values.

The *Context* XML is comprised of a series of *Condition* elements—one for each variable in the session. You can parse it to identify each variable and extract the required value or other data.

Here's a snippet with two *Condition* (elements) to illustrate how variables are represented:

```
<Context... >
  <a:Condition>
    <a:xmlns... />
    <Identifier>4949b7da-3253-490d-96b2-d0409ac406ad</Identifier>
    <CustomEditorType i:nil="true" />
    <Name>Content Duration</Name>
    <ParameterSelections />
    <ParameterSetCollections />
    <Parameters />
    <Summary i:nil="true" />
    <Instance>247afdf9-e243-4ead-be1d-7317aecc9273</Instance>
    <ConditionValue>
      <ComplexValue i:nil="true" />
      <Text xmlns:b=...>
        <b:string>00:00:30:00@29.97</b:string>
      </Text>
      <Default>
        <ComplexValue i:nil="true" />
        <Text xmlns:b=...>
          <b:string>00:00:00:00@29.97</b:string>
        </Text>
      </Default>
    </ConditionValue>
    <TypeCode>TimeCode</TypeCode>
  </a:Condition>
  <a:Condition>
    <a:xmlns i:nil="true" xmlns:b=... />
    <Identifier>432820a7-d6f2-42f0-877e-f5b6b16c6081</Identifier>
    <CustomEditorType i:nil="true" />
    <Name>SdkBitRate</Name>
    <ParameterSelections />
    <ParameterSetCollections />
    <Parameters />
    <Summary i:nil="true" />
    <Instance>5b41dbd5-5d91-4c6a-9d00-a5c92411397a</Instance>
    <ConditionValue>
      <ComplexValue i:nil="true" />
      <Text xmlns:b=...>
        <b:string>9943</b:string>
      </Text>
    <Default>
      <ComplexValue i:nil="true" />
      <Text xmlns:b=...>
        <b:string>0</b:string>
      </Text>
    </Default>
    </ConditionValue>
    <TypeCode>Int32</TypeCode>
...
```

Obtaining the Variable's Value

Using the variable's name, you can extract its value. The name of the variable is in the <Condition><Name> element; its value is in the <Condition><ConditionValue><Text><String> element.

This X Path expression selects the text string of the *text* node of the *Content Duration* variable:

```
/Context/Conditions/Condition[Name='Content Duration']/ConditionValue/Text/
string/text()
```

The duration of the encoded media in this example is 00:00:30:00@29.97.

This sample X Path expression selects the text string of the *text* node of the *SDKBitRate* variable:

```
/Context/Conditions/Condition[Name='SDKBitRate']/ConditionValue/Text/string/
text()
```

The bit rate of the encoded media in this example is 9943 bits per second.

Accessing Variable Values from Labels in Jobs

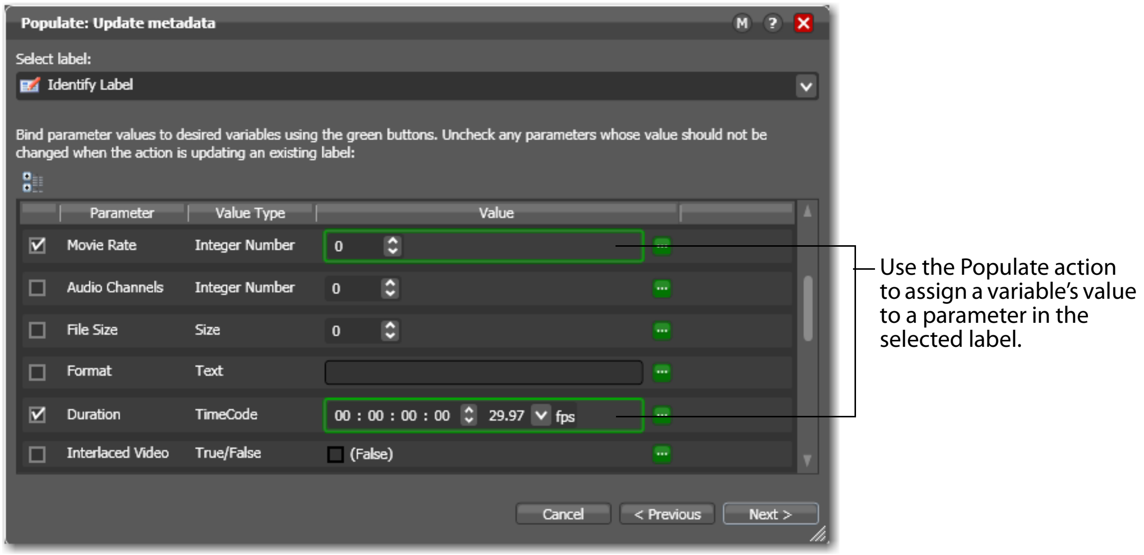
Note: This task is implemented in the *JavaRestAppNoteSample* program in *retrieveLabel*.

You typically access a metadata label to gain access to individual metadata, identified as a parameter in the label. When a job is complete, metadata labels specified in the workflow are saved with the job.

Note: Metadata label processing is a specially-licensed feature. Contact Telestream Customer Service or your Telestream representative for more information.

Metadata labels are generated in a workflow by using a Populate action, where you assign a variable's value to a specified parameter in the label you specify. This is one method of accessing runtime values from a job. The other method is to use variables (see Retrieving Variable Values From Jobs).

Figure 12. Binding Variables to Label Parameters in the Populate Action



Populate: Update metadata

Select label:

Identify Label

Bind parameter values to desired variables using the green buttons. Uncheck any parameters whose value should not be changed when the action is updating an existing label:

	Parameter	Value Type	Value
✓	Movie Rate	Integer Number	0
	Audio Channels	Integer Number	0
	File Size	Size	0
	Format	Text	
✓	Duration	TimeCode	00 : 00 : 00 : 00 29.97 fps
	Interlaced Video	True/False	(False)

Cancel
< Previous
Next >
Use the Populate action to assign a variable's value to a parameter in the selected label.

To process parameters in a label, you need to know the label and the parameter you want to obtain values from. In Workflow Designer, collect the label's name and the parameters you need. You can view the label by displaying the workflow in Workflow Designer and displaying the Populate action's inspector.

Extracting variable values stored in parameters in metadata labels involves these general tasks, which are described in the topics following:

1. Using the job GUID (see Obtaining the Job GUID, described previously), obtain the list of contents of the job. See Obtaining the Contents for a Job.
2. Extract the job's content GUID from the contents list. See Extracting the Content GUID.
3. Using the content GUID, obtain the job's labels. See Obtaining Labels.
4. Extract each target variable's value from the label's parameter. See Extracting the Parameter's Value.

Obtaining the Contents for a Job

To obtain the content generated from a job, execute a GET /JobRest/GetContentsForJob method. You need to provide the GUID of the target job.

Here's an example:

```
http://VantageServer:8676/JobRest/GetContentsForJob
?identifier=ccb1ef3c-15e6-43ea-ac89-c39ea8bf63c3
```

The returned XML document is named *ArrayOfContentType*. An *ArrayOfContentType* XML contains the *ContentType* element.

Here's an *ArrayOfContentType* XML to illustrate what *GetContentsForJob* returns:

```
<ArrayOfContentType...">
  <ContentType>
    <Identifier>478b3e6d-eb5e-4dfc-8d70-ce5cce0b008d</Identifier>
    <Created>2015-02-11T15:51:25.917</Created>
    <Name>720x480.422</Name>
    <Updated>2015-02-11T15:51:50.843</Updated>
  </ContentType>
</ArrayOfContentType>
```

Extracting the Content GUID

Parse the *ArrayOfContentType* to extract the *ContentType* GUID (in the *Identifier* element) for use in accessing the content.

Here is a sample X Path expression that selects the *Identifier* node:

```
//Identifier[ancestor::ContentType]
```

Now that you have the content GUID, you can obtain its items, and extract parameter values.

Obtaining Labels

To obtain a job's labels, execute Get /ContentRest/GetContentLabels, using the content GUID.

Here's an example:

```
http://VantageServer:8676/ContentRest/GetContentLabels
?identifier=478b3e6d-eb6e-4dfc-8d70-ce5cce0b008d
```

The returned XML document is *ArrayOfItemType* with a set of *Label* item types—one for each label in the content of the job. You can parse a label to identify each parameter and extract the required data.

Here's a snippet to illustrate how labels and their parameters are represented. This example illustrates the *Move Rate* variable, whose value is 18931, an Int32:

```
<ArrayOfItemType...>
  <ItemType>
    <Identifier>9d319fc9-0e4a-0098-3bbe-ce70c8ad3613</Identifier>
    <Item i:type="Label">
      ...
      <Parameters>
        <Parameter>
```

```
          <Text... >
            <a:string>18931</a:string>
          </Text>
          <Identifier>2a8a4996-65bb-4d0a-9d8a-74aee2959409</Identifier>
          <TypeCode>Int32</TypeCode>
          ...
          <Name>Movie Rate</Name>
          ...
        </Parameter>
...
```

You can parse each parameter in a label to identify and extract its value.

Extracting the Parameter’s Value

Using the parameter’s name, you can extract its value from the *ArrayOfItemType*.

The name of the parameter is in the <*ArrayOfItemType*><*ItemType*><*Item*><*Parameters*><*Parameter*><*Name*> element.

The value of the parameter is in the child <*Text*><*String*> element.

This sample X Path expression selects the text string of the *text* node of the *Content Duration* variable:

Here is a sample X Path expression that selects the text of the parameter’s *string* node:

```
/ArrayOfItemType/ItemType/Item/Parameters/Parameter[Name='Movie Rate']/Text/
string/text()
```

Accessing Media Files from Jobs

Note: This task is implemented in the *JavaRestAppNoteSample* program in *retrieveMedia*.

When a job is complete, the fully-qualified path to all media created in the workflow is saved in the job records.

To obtain the path to the media, you need to know the nickname of the file you want to access. In Workflow Designer, select the workflow and identify the nickname on the action that creates or operates on the file.

1. Using the job GUID (see Obtaining the Job GUID, described previously), obtain the list of contents of the job. See Obtaining the Contents of a Job.
2. Extract the job's content GUID from the contents list. See Extracting the Content GUID.
3. Using the content GUID, obtain its list of media. See Obtaining the List of Media Items.
4. Extract the media item's GUID from the list of media. See Extracting the Media Item's GUID.
5. Using the media item's GUID, obtain the fully-qualified path to the media file. See Obtaining the Media Item's Fully-qualified Path.

Obtaining the Contents of a Job

To obtain the content generated by a job (labels, media, etc.), execute a GET /JobRest/GetContentsForJob method. You need to provide the GUID of the target job.

Here's an example:

```
http://VantageServer:8676/JobRest/GetContentsForJob
?identifier=ccb1ef3c-15e6-43ea-ac89-c39ea8bf63c3
```

The returned XML document is named *ArrayOfContentType*. An *ArrayOfContentType* XML contains the *ContentType* element. You can parse it to obtain the GUID of the content you want to access.

Here's an *ArrayOfContentType* XML to illustrate what *GetContentsForJob* returns:

```
<ArrayOfContentType...>
  <ContentType>
    <Identifier>478b3e6d-ebbe-4dfc-8d70-ce5cce0b008d</Identifier>
    <Created>2015-02-11T15:51:25.917</Created>
    <Name>720x480.422</Name>
    <Updated>2015-02-11T15:51:50.843</Updated>
  </ContentType>
</ArrayOfContentType>
```

Extracting the Content GUID

Parse the *ArrayOfContentType* to extract the *ContentType* GUID (in the *Identifier* element) for use in accessing the content.

Here is a sample X Path expression that selects the *Identifier* node:

```
/ArrayOfContentType/ContentType/Identifier
```

Now that you have the content GUID, you can obtain its items, and extract the file path.

Obtaining the List of Media Items

Once you have the GUID of the content, you can obtain the media items in the content. Execute Get /ContentRest/GetContentMedia, using the content GUID.

Here's an example:

```
http://VantageServer:8676/ContentRest/GetContentMedia?
identifier=32a61634-5f12-4916-8f69-5bb0eff061cf
```

The returned XML document is *ArrayOfItemType*. This XML is comprised of a series of *Item* elements of type *Media*—one for each media file in the job. Each is uniquely identified by its nickname. For example, a Flip action configured to generate a media file nicknamed *Output*.

Here's a snippet of an *ItemType* element, to illustrate how each item is represented:

```
<ItemType>
  <Identifier>fb181780-5501-051f-123e-1f5a36154b62</Identifier>
  <Item i:type="Media">
    <xmlns i:nil="true" xmlns=... />
    <Identifier>c9be01b4-0a13-4c09-9d57-44ead9e52aad</Identifier>
    <CustomEditorType i:nil="true" />
    <Name>Output</Name>
```

There are dozens of detail elements (including parts of path and filenames).

Extracting the Media Item's GUID

Parse the *ArrayOfContentType* to identify the target media file by its nickname (the *Name* element in the *Item* element) and then extract the *ContentType*'s *Identifier* GUID to use in identifying the path.

Here is a sample X Path expression that selects the *Identifier* node of the item with the nickname `Output`:

```
/ArrayOfItemType/ItemType[Item/Name = 'Output']/Identifier
```

Now that you have the item's GUID, you can obtain its details and extract the file path.

Obtaining the Media Item's Fully-qualified Path

Once you have the GUID of the media item, you can obtain the fully-qualified path to the file. Execute Get /ContentRest/GetItemFilePaths, using the *ItemType's* GUID for the target file, as identified by the file's nickname.

Here's an example:

```
http://VantageServer:8676/ItemRest/GetItemFilePaths?
identifier=fb181780-5501-051f-123e-1f5a36154b62
```

Here's the returned *ArrayOfString* element, to illustrate how the file's path is presented:

```
<ArrayOfstring xmlns="http://schemas.microsoft.com/2003/10/
Serialization/Arrays" xmlns:i=...>
  <string>C:\SDKTestOut\720x480.422.27.m4v</string>
</ArrayOfstring>
```

Parse the *ArrayOfString* element with the X Path `//string` to extract the fully-qualified path to the target media file.

Common Vantage REST Methods

This topic describes how to use many of the commonly-used REST methods. Most are Get methods; others are Post methods. All parameters are strings.

Note: For a complete list of all REST methods in the Vantage API, use the Vantage REST utility.

- POST /SubmitRest/SubmitFile
- GET /DomainRest/GetWorkflows
- GET /WorkflowRest/GetWorkflowVariableRequirements
- GET /JobRest/GetJobState
- GET /WorkflowRest/GetSessionsForJob
- GET /JobRest/GetContentsForJob
- Get /ContentRest/GetContentLabels
- Get /ContentRest/GetContentMedia
- Get /ContentRest/GetItemFilePaths

Specifying the Host Name in a URI

The Vantage domain host name is the Vantage domain server identity: computer name | IP address | *localhost*, plus the SDK Service port (default 8676), separated by a colon.

In an All-in-One domain, use the host name of the server on which Vantage is running. In an array, use the server running the SDK Service. When a Vantage SDK client is running on the same server, you can use *localhost*.

Examples

```
http://localhost:8676/SubmitRest/SubmitFile
```

```
http://Vantage_Domain_Seven:8676/SubmitRest/SubmitFile
```

```
http://192.168.1.1:8676/SubmitRest/SubmitFile
```

POST /SubmitRest/SubmitFile

Submits a job to the specified workflow, in the specified domain.

Requires the workflow identifier, the source file, and a job name. Context (input variables) is optional.

Returns the *guid* in XML format.

Parameters

workflowIdentifier (required)

A GUID assigned to the workflow, generated automatically you create a workflow in Workflow Designer.

To obtain a workflow's GUID, open Workflow Designer, select the workflow and display the workflow details panel. You can also access workflows and their GUID by executing */DomainRest/GetWorkflows*.

sourceFilename (required)

The fully-qualified path to the media being submitted. If on the same server as all of the services that must access it, you can use a drive-letter path. Otherwise, reference the media as a share.

Note: Each back-slash in the path string must be doubled up, because the back-slash is an escape character in the context of a Java string.

Examples (a drive-letter path, and a share):

```
C:\\Media\\Ginger_Ale_Project\\Sierra_snow.mpg
```

```
\\\\Medusa_Server\\Ginger_Ale_Project\\Sierra_snow.mpg
```

context (optional)

The *context* XML file with runtime values for each variable defined in the workflow.

jobName (required)

A text string identifying the user-facing name of the job, displayed in the job status tabs in Workflow Designer during and after job execution.

URI Example (Without Variables)

```
http://VantageDomainServer:8676/SubmitRest/SubmitFile?
workflowIdentifier=29504355-6190-4b76-b5fe-c5082e830122
&sourceFilename=%5C%5CVantageServer%5Cmedia%5C720x480.mpg
&jobName=This+Is+My+Job
```

GET /DomainRest/GetWorkflows

Obtains the list of workflows (referred to as *Procedures*) in XML format that are present in the domain.

There are no parameters.

URI Example

```
http://VantageServer:8676/DomainRest/GetWorkflows
```

GET /WorkflowRest/GetWorkflowVariableRequirements

Obtains the list of input variables (known as the *Context*) in XML format defined in the workflow. This is a list of all variables, each of which have a default value and have been bound to a parameter in an action in the workflow. If these conditions are not met, the variable is not present in the *Context*.

Requires the workflow identifier GUID.

Parameters

identifier (required)

A workflow GUID, generated automatically you create a workflow in Workflow Designer.

To obtain a workflow's GUID, open Workflow Designer, select the workflow and display the workflow details panel. OR, execute */DomainRest/GetWorkflows*.

URI Example

```
http://VantageServer:8676/WorkflowRest/GetWorkflowVariableRequirements?
identifier=29504355-6190-4b76-b5fe-c5082e830122
```

GET /JobRest/GetJobState

Obtains the current state of the target job. Returns the *WorkflowJobState* XML. The *WorkflowJobState* provides a set of reserved keywords to represent the state of the job:

Active | *Failed* | *StoppedByUser* | *Complete* | *WaitingToRetry*.

Requires the job identifier GUID.

Parameters

identifier (required)

A job GUID, generated when a job is executed in Vantage.

You can obtain a job GUID when you submit a job, or by executing */JobRest/GetJobsFor-Workflow*.

Full URI Example

```
http://Vantage_Domain_Seven:8676/JobRest/GetJobState?
identifier=29504355-6190-4b76-b5fe-c5082e830122
```

GET /WorkflowRest/GetSessionsForJob

Obtains the sessions in the job. Returns *Context* in XML format. This is a list of all sessions that executed in the job. (A session is an execution of an action instance.)

Requires the job identifier GUID.

Parameters

identifier (required)

A job GUID, generated when a job is executed in Vantage.

You can obtain a job GUID when you submit a job, or by executing */JobRest/GetJobsFor-Workflow*.

URI Example

```
http://localhost:8676/WorkflowRest/GetSessionsForJob?
identifier=29504355-6190-4b76-b5fe-c5082e830122
```

GET /JobRest/GetContentsForJob

Obtains the contents of the target job. Returns an *ArrayOfContentType* in XML format, with details about the job's content object. You can use the content's GUID to access the job's content.

Requires the job GUID.

Parameters

identifier (required)

A job GUID, generated when a job is executed in Vantage.

You can obtain a job GUID when you submit a job, or by executing */JobRest/GetJobsFor-Workflow*.

URI Example

```
http://192.168.1.1:8676/WorkflowRest/GetSessionContext?
identifier=29504355-6190-4b76-b5fe-c5082e830122
```

Get /ContentRest/GetContentLabels

Obtains the *ArrayOfItemType* XML from the specified *Content*. This *ArrayOfItemType* is a list of labels and their parameters and all details.

Parameters

identifier (required)

The Content GUID. To obtain the GUID, execute GET /JobRest/GetContentsForJob.

Get /ContentRest/GetContentMedia

Obtains the *ArrayOfItemType* XML from the specified *Content*. This *ArrayOfItemType* is a list of media (the *ItemType* element) in the contents of the job, plus related details.

Parameters

identifier (required)

The *Content* GUID. To obtain the GUID, execute GET /JobRest/GetContentsForJob.

Get /ContentRest/GetItemFilePaths

Obtains the *ArrayOfItemType* XML from the specified *Content*. This *ArrayOfItemType* is a list of media and all details.

Parameters

identifier (required)

The *Content* GUID. To obtain the GUID, execute GET /JobRest/GetContentsForJob.

Vantage Terms and Concepts

context

The set of variables defined in the various actions that comprise the target workflow, presented in XML format.

condition

The definition of a variable including its value, in the context XML object.

Procedures

The set of workflows (including their name and GUID) defined in the target domain, presented in XML format.

session

An action execution instance in the context of a job.

Sessions

The set of session (action) instances in a given job, presented in XML format.

Copyright and Trademark Notice