|  |
| --- |
|  |
| Virtual Node Administrator Guide  For Virtual Exchange Service |
|  |
|  |
|  |
| **03/12/2025** |

|  |
| --- |
| This document provides basic instructions for managing Virtual Exchange Service using the Virtual Exchange Service Administrator. |

Table of Contents

[1 Introduction 3](#_Toc178232600)

[1.1 What is a Virtual Exchange Service (VES)? 3](#_Toc178232601)

[1.2 Virtual Exchange Service Features 3](#_Toc178232602)

[1.3 Virtual Exchange Service Security 3](#_Toc178232603)

[1.4 Major Objects in a Virtual Exchange Service 4](#_Toc178232604)

[2 Inheritance Model 4](#_Toc178232605)

[2.1 Create a Virtual Exchange Service 5](#_Toc178232606)

[2.2 Virtual Exchange Service Endpoints 7](#_Toc178232607)

[2.3 Node Owners 7](#_Toc178232608)

[3 Data Source Management 8](#_Toc178232609)

[3.1 Network Connectivity Options 8](#_Toc178232610)

[3.2 Create Data Source 8](#_Toc178232611)

[3.3 Data Source Security 10](#_Toc178232612)

[4 Dataflow Management 10](#_Toc178232613)

[4.1 Create a Dataflow 10](#_Toc178232614)

[4.2 Import a Dataflow 11](#_Toc178232615)

[5 Service Management 12](#_Toc178232616)

[5.1 Create a Service 13](#_Toc178232617)

[5.2 REST Interface 16](#_Toc178232618)

[5.3 Clone Services 16](#_Toc178232619)

[5.4 Test Services 17](#_Toc178232620)

[5.5 Publish Services 19](#_Toc178232621)

[5.6 Update Services 20](#_Toc178232622)

[6 Task Management 21](#_Toc178232623)

[7 Document Headers 23](#_Toc178232624)

[8 Transaction Management 23](#_Toc178232625)

[9 Appendix 26](#_Toc178232626)

[9.1 Retrieve Transaction Information 26](#_Toc178232627)

[9.2 Information Exchange Patterns and Common Workflows 29](#_Toc178232628)

[9.2.1 Basic Submission Workflow 29](#_Toc178232629)

[9.2.2 Generic Submission Workflow with Validation 30](#_Toc178232630)

[9.2.3 Generic Query 30](#_Toc178232631)

[9.2.4 Local Query and Submit 31](#_Toc178232632)

[9.2.5 Generic Solicit 32](#_Toc178232633)

[9.2.6 TransactionReport 32](#_Toc178232634)

# Introduction

## What is a Virtual Exchange Service (VES)?

VES is a Network Node implementation for the cloud environment, offered Software as a Service (SaaS). While the owner controls its security and functionalities, operations and maintenance will be supported by the hosting party.

The VES is a node that shares the physical implementation with other nodes while functioning distinctly with its own business processes (dataflow and services), security controls, and transaction isolations. In a cloud environment, the VES is deployed as a multi-tenant model.

## Virtual Exchange Service Features

* **Simplicity**: VES drastically simplifies the development of dataflows. In most situations, services can be created with a few button clicks.
* **Maximize Sharing**: VES maximizes sharing of the node implementations and makes a node readily available to all users and partners so that they can focus on building dataflows instead of network nodes.
* **Simplify Management**: All services will be hosted in a centralized and managed environment to ensure the highest quality of services and maximum availability. A dashboard is offered for node administrator to control all aspects of a node.
* **Reduce Costs**: Built into its architecture, all business processes can be shared in VES to significantly reduce development cost of new dataflow. Many expenses associated with node deployment, operation and maintenance can be eliminated.
* **Enhance Quality of Service (QoS)**: With a clustered virtual exchange service in the cloud environment, service interruption by interoperability, server load or regular maintenance will be minimized. The availability of network nodes can be increased significantly.
* **Agile Business Process**: VES provides a workflow platform for orchestrating complex business operations, which greatly reduces the time-to-operation for dataflow development and deployment.
* **High Performance**: VES is based on an optimized Simple Object Access Protocol (SOAP) implementation in Java with minimal overhead.

## Virtual Exchange Service Security

VES is fully integrated with Network Authentication and Authorization Service (NAAS) for user authentication and authorization. When a VES is created, it is assigned to an owner; the service owner has full control over who can access each service using NAAS security policies.

Although hosted in the same environment with other nodes, the VES management interface operates in its own sandbox and allows only the service owner to make changes to the VES properties and configurations. For instance, one node administrator will not be able to create a service for a different node.

A VES has the same access control mechanisms as any Network node. The node administrator’s authorization is required for accessing node services. Please contact the Node Helpdesk at: [nodehelpdesk@epa.gov](mailto:nodehelpdesk@epa.gov) for access.

## Major Objects in a Virtual Exchange Service

A VES has six key objects that its owner can create and manage. They are:

* **Node**: This contains the definition of a VES including its address (endpoint), description, owner, and other properties.
* **Data Source**: This defines an access point where information is supplied. In most of situations, a data source contains database server name, address, login account and other connection information.
* **Dataflow**: This is a logical collection of services that deal with common set of information exchanged between partners (e.g., Resource Conservation and Recovery Act (RCRA)). A VES owner can create a dataflow and set its properties.
* **Service**: A service is a definition of what to be provided. A VES owner adds new features to a network node by creating services. The services are the basic operation unit that a node executes at runtime.
* **Task**: A task is a set of operations to be executed automatically on a scheduled basis. For instance, a task can be created to perform quarterly submissions to a CDX dataflow.
* **Document Header**: A document header is information appended to the heading of a newly created document when a service is executed. This Header information is used at the beginning of a document upon the completion of constructing instance documents, which includes any query results and it’s corresponding eXtensible Stylesheet Language Transformations (XSLT).

The Virtual Exchange Service Administrator (VESA) uses SOAP web services to manage all objects created on a virtual exchange service. These are used by the VESA and they can also be integrated into other applications or automated using scripts. Node owners use VESA to manage node objects from an Internet browser.

The following sections focus on how to manage VES Nodes using VESA, which is currently available at:

|  |  |
| --- | --- |
| Development | <https://vesdev.epacdxnode.net/VESA> |
| Test | <https://vestest.epacdxnode.net/VESA> |
| Production | <https://ves.epa.gov/VESA> |

# Inheritance Model

The VES environment has an inheritance model to maximize sharing of services as shown below:



There is a special node called ‘***VirtualNode,’*** which contains canned implementations of dataflows and services. Other network nodes inherit services that are defined in the ***VirtualNode*** automatically. Currently implemented dataflow and services in the ***VirtualNode*** include:

1. **ADMIN\_v1.0 dataflow and services**: This is a set of services that allows users to retrieve node transaction information. Admin\_v1.0 is a standard Exchange Network (EN) dataflow that has services, such as GetTransactionList and GetTransactionDetail. It is very useful for statistical analysis and transaction tracking.
2. **ENDS\_v1\_0 dataflow and services**: This is a set of services for publishing service information for your node. This implementation automatically publishes services to Exchange Network Discovery Services (ENDS) for you using either the pulling (GetServices) or pushing (service submission).

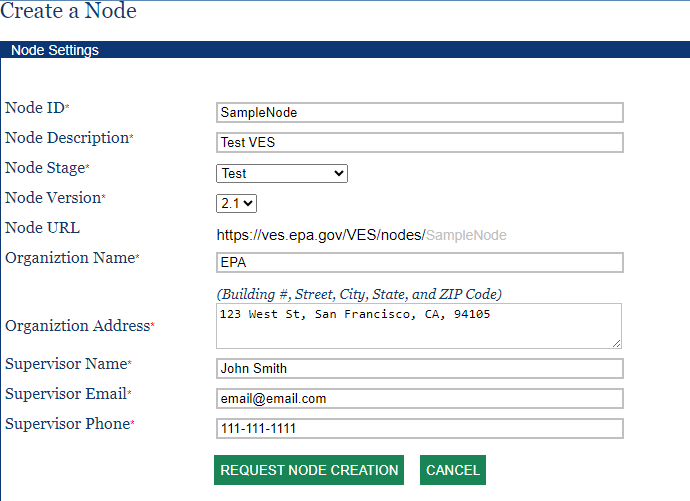
These services, and any additional approved services, are available immediately as soon as the node is created.

**Node Management:**

A node is the top-level object. It can only be created by a node administrator or an authorized user. Each administrator can create two nodes: a test node and a production node. Please contact the Node Helpdesk at: [nodehelpdesk@epa.gov](mailto:nodehelpdesk@epa.gov) if you want to become a VES owner.

## Create a Virtual Exchange Service

The first task of a VES owner is to create a network node. After logging in to the VESA using your NAAS account, you can click on the ‘**Create a Node**” link to add a node. The following screen should be displayed:



The form can contain the following entries.

* **Node ID**: This is a primary identifier of the network node. It should contain only alphanumeric characters without whitespaces. It is a good practice to use an abbreviated organization name, suffixed by the environment, as the node ID. Node ID must be unique and cannot be changed after creation. (e.g., VATest)
* **Node Description**: This is a brief description of your node.
* **Node Stage**: The environment of the node, it can be either ‘Dev’, ‘Test’, or ‘Production’.
* **Node Version**: This is the version of a node. The current version should always be selected unless there are other special reasons.
* **Node URL**: This is your node address (also referred to as the endpoint). It is constructed automatically based on the hosting environment, domain name and node identifier. It is the address to which all requests to your node should be sent.
* **Owner**: An email address or list of email addresses (separated by a semicolon) who will have access rights to the node and any associated entities related to the node, such as dataflows and services.
* **Organization Name**: The name of your organization. The information is used for node approval and service publishing to the ENDS.
* **Organization Address**: This should be the complete address of your organization.
* **Supervisor Name and Supervisor Phone**: This should be the person in your organization who can be contacted for verification.

When the ‘**Request Node Creation**’ button is clicked, a new node will be created, but it will be marked as "Pending" approval. An email will be sent to Environmental Protection Agency (EPA) for review / approval.

The user who created the node will be assigned as the initial owner (administrator) for the virtual exchange service.

## Virtual Exchange Service Endpoints

When a network node is created, it is automatically assigned an address where SOAP requests are accepted. The address is the node ID prefixed with a base URL, depending on the environment, the base URLs are:

|  |  |
| --- | --- |
| Development | https://vesdev.epacdxnode.net/VES/nodes/ |
| Test | https://vestest.epacdxnode.net/VES/nodes/ |
| Production | https://ves.epa.gov/VES/nodes/ |

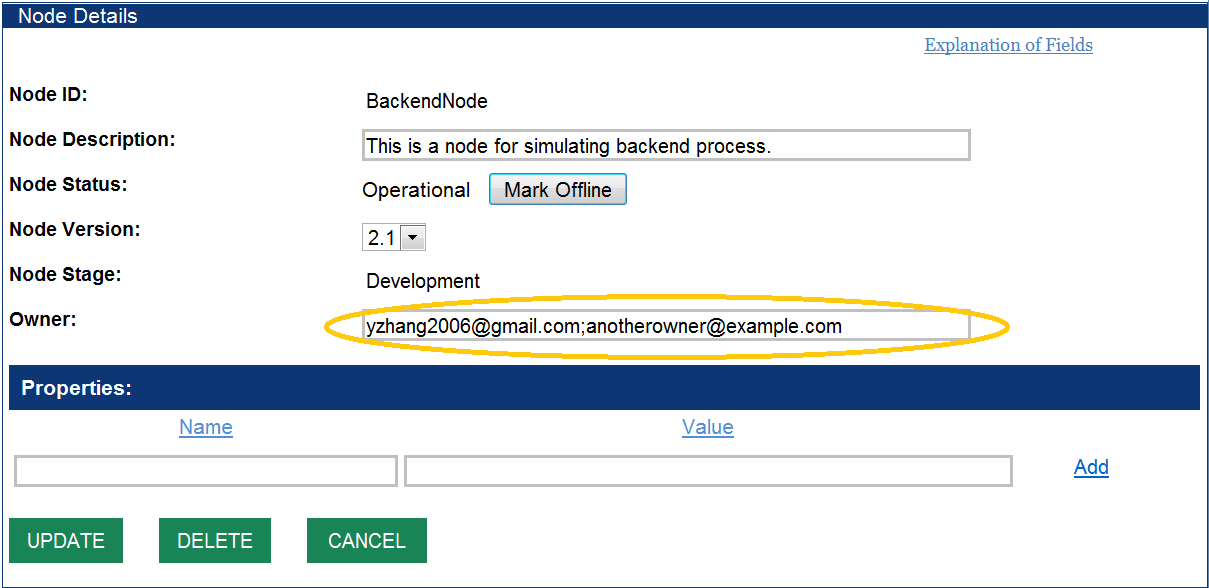
For example, if your node ID is MyNode, the production node address would be:

https://ves.epa.gov/VES/nodes/MyNode

The URL is case insensitive. The production VES and VESA are hosted in Microsoft Azure Cloud environment.

## Node Owners

A node may have multiple owners, each can perform node management tasks independently. The current owner can add additional owners in the Update Node screen as shown below:



Multiple owners should be separated by semicolons. The new owner must be a node administrator and be authorized by the Node Helpdesk to manage the node.

# Data Source Management

In the optimized VES architecture, only the network node is migrated to the VES in the cloud. The application databases and staging databases can remain in the current enterprise network and have the VES connect to them via secure channels. This is the simplest implementation and represents the lowest burden to partners. Alternatively, staging databases could be copied and maintained in the cloud along with the network node, but they would still require a data source definition for that location. So, how can your node retrieve environmental information from your local database servers? The Data Source provides the basic information for securely connecting to external databases from your VES.

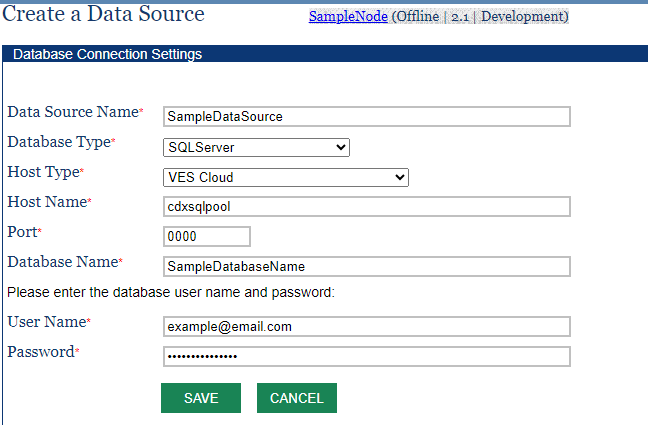
## Network Connectivity Options

Before creating a Data Source, there must be network connectivity from the VES to the database server. To create this connection, we utilize the Virtual Exchange Service Connector (VES Connector).The service connector is provided to relay network traffic between network node and your local database server once authenticated. This provides secure network connectivity without changing inbound firewall rules. Additional outbound firewall rules may be required.

It is the responsibility of a VES Node owner to establish network connectivity. The VES engineer and EN Helpdesk will provide assistance as needed.

## Create Data Source

A new data source can be created by clicking on the ‘**Data Sources**’ link on the left panel of VESA, and then the ‘**Add New**’ button at the top right. A form will be displayed as below:



Elements for data sources are:

* **Data Source Name**: A unique name for your data source, it should contain only alphanumeric characters without any white spaces. The same name will be used in node service definitions.
* **Database Type**: The type of database servers. Currently supported database servers are Structured Query Language (SQL) Server, SQLAzure, and Oracle.
* **Host Type**: The type of host, choices include VES Cloud, Remote Database via VESConnector, and Other Remote Database.
* **Host Name**: For VES Cloud and other, the name of a database server. It can also be an Internet Protocol (IP) address. If the data source is for a remote database with a VESConnector, please use ***localhost*** as the Host Name.
* **Port**: The database listening port. The default ports for SQL Server and Oracle are 1433 and 1521, respectively. Please contact your Database Administrator (DBA) for port information.
* **VES Connector Port**: If the VESConnector is used, a private database port will be assigned to your database. Please contact [nodehelpdesk@epa.gov](mailto:nodehelpdesk@epa.gov) if you don’t know your port number.
* **Database Name**: This is the name of the database for SQL Server. For Oracle, this should be the Service Name, not the System Identifier (SID) (Oracle database instance name) – Note: Oracle service name is case-sensitive. For Oracle server with Virtual Exchange Service Connector, this should be the Oracle service name suffixed with the network domain name. For instance, if your service name is **dstore** and domain is example.com, then the Database Name should be **dstore.example.com**.
* **User Name**: This is the user ID for logging on to the database server.
* **Password**: The database account password. The password will be encrypted on the server.

When the “**Save**” button is clicked, the data source information will be saved under the corresponding node found at the top of the page. Services that encounter data source errors should return to the created data source to ensure the correct information is associated with this data source such as database type, port number, database name, and credentials.

## Data Source Security

A data source, unlike other assets of a node, is dedicated to the node it belongs to, and not shared with other nodes. At runtime, the VES engine ensures the data source can only be referenced by the owner’s node.

Database account information is encrypted using a strong algorithm. The runtime engine uses the account information when making a database connection (Only the runtime engine can decrypt this information).

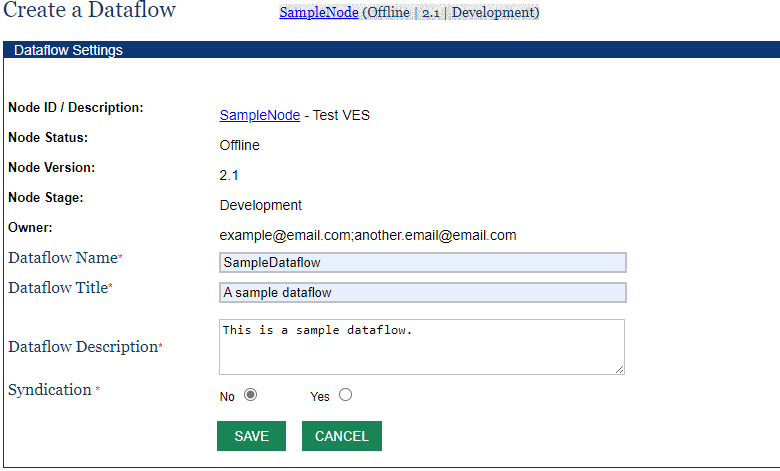
Data sources are your private information. They are not published or disclosed to service registries or anyone else.

# Dataflow Management

Dataflow is considered as a collection of the same, or similar, types of information exchanges. In the Node Functional Specification, dataflow is also a container of service definitions and is a required parameter that must be supplied to Submit, Query, Solicit, and Execute methods.

## Create a Dataflow

A dataflow must be defined before creating any services. In VESA, click the ‘**Dataflows**’ link on the left panel, and then the ‘**Add New**’ button on the dataflow list page. The following HTML form will be displayed:



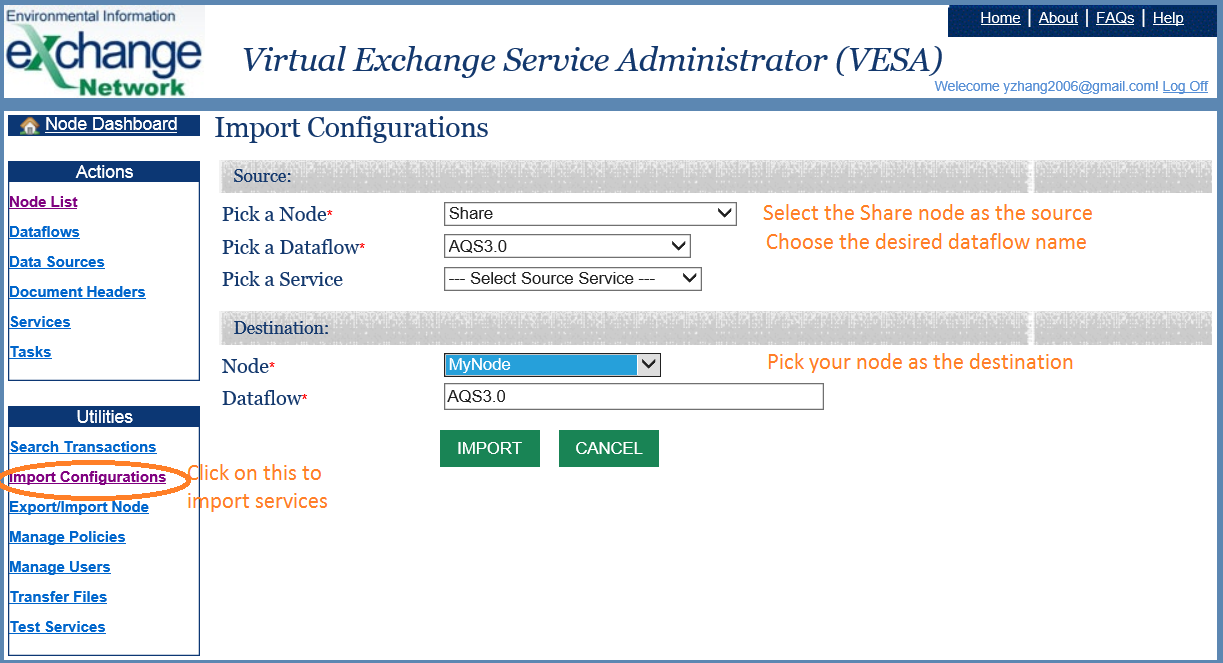
The entries of the form are very simple and straight forward:

* **Dataflow Name**: This is the name of a dataflow. It must be a unique name in VES and contain only alphanumeric characters without any whitespaces. Note that the EN defines standard dataflow names for major data exchanges. Please refer to flow configuration documents on the [Exchange Network website](http://www.exchangenetwork.net/data-exchange/).
* **Dataflow Title**: This is a short, user-friendly name of the dataflow.
* **Dataflow Description**: This is additional description of the dataflow.
* **Syndication**: This Boolean value indicates whether transactions of the dataflow should be published as Really Simple Syndication (RSS) or Atom Syndication Format (ATOM) news feeds or not. If Syndication is Yes, transactions of the dataflow is published as news feeds at:  
   *https://{virtualNodeDomain}/getnewsfeed?node={nodeId}&dataflow={dataflowName} &token={securityToken}&format=RSS|ATOM*  
    
  A security token is required for protected services.

## Import a Dataflow

The VES supports almost all major data flows. They can be imported into your node with the following procedure:

1. Click on the ‘Import Configuration’ from the left panel.
2. On the Import Configuration screen (see screenshot below), select ‘**Share’** as the source node.
3. Choose the desired Dataflow from the dropdown list. (Do not choose a service).
4. Pick your node name as the destination.
5. Click on the ‘**Import’** button to import all services associated with the dataflow.



This is the easiest way to get up and running with the common dataflows. The operation will import dataflow definitions, service definitions/parameters, document headers, XSLT stylesheets and SQL statement if any.

**Important**: *It will not import data sources associated with the services. You must manually edit the services to set your data source for the service to work properly.*

# Service Management

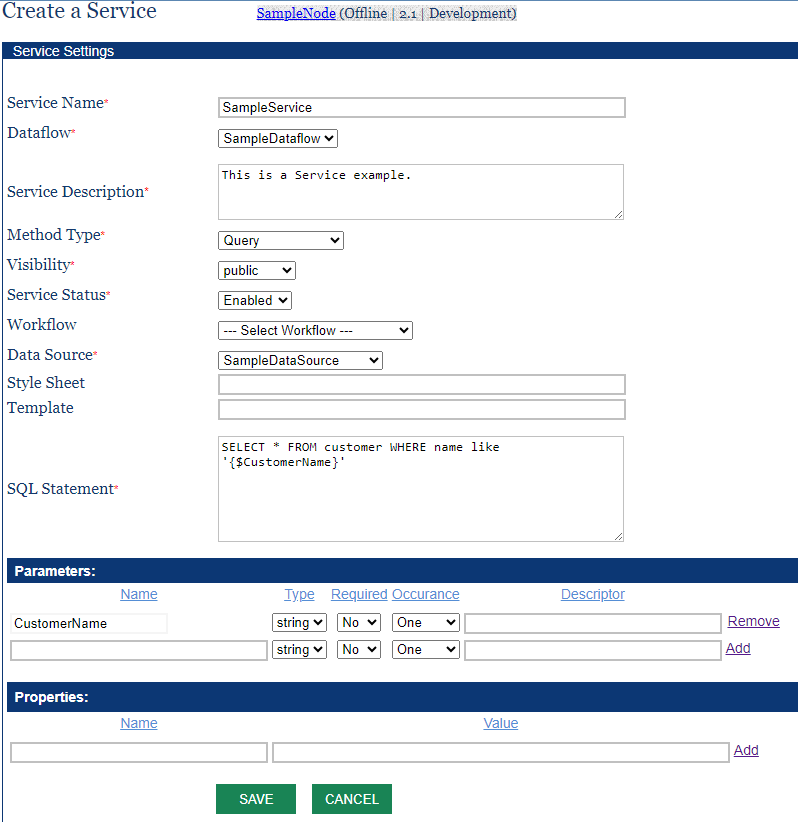
The Network Node functional specifications define a very generic and flexible framework for creating new services without changing the node interface and service definitions (Web Services Description Language (WSDL) remains the same). There are three main categories of web services an owner can create:

1. **Incoming Services**: Services that allow users to send information to your node. The services are handled by the Submit method and implemented as workflows in network node. Submit\_v1.0 and ENDS\_v2.0 are examples of incoming services.
2. **Outgoing Services**: These are services for publishing information to others. Outgoing services are supported by Query and Solicit methods, implemented through a generic database layer with dynamic XSLT transformation or template instantiation in the VES GetFacilityList and GetServiceList are examples of outgoing services.
3. **Executable Services**: Executable services are mechanisms for adding utility services that are not necessarily related to data submission or data publishing. Executable services are implemented with the capability to extend node functionality. Any new service that could not be offered through other node methods (Query, Submit) can be offered through Execute. Execute services are found useful for scheduled tasks. Some examples for Executable services for scheduled tasks are ReportTransactionSummary and BatchUpdateStatus.

As will be discussed in the following subsections, the VES has a set of common workflows that can be reused for processing data submissions so that a node owner can just configure and select the right workflow for incoming services. For outgoing services, the VES can publish information as an Extensible Markup Language (XML) Dataset given an SQL statement. An XSLT stylesheet or a template can then be used to convert the resultant XML into other forms if needed.

## Create a Service

To create a Query/Solicit service, you must have two items: a dataflow where the service resides and a data source where the information resides. In VESA, click on the ‘**Services**’ link in the left panel and then the ‘**Add New**’ button on top right of the page, the following page will be displayed for a new service:



The entries displayed in the form are:

* **Service Name**: The unique name of a service in your node. It must contain only alphanumeric characters without any whitespaces. We suggest using the camel convention for service names. For example, GetFacilityList and GetServiceDetails are valid service names.
* **Service Description**: A brief description of the service.
* **Method Type**: This is the node method under which the service is published. It can be Query, Solicit, Submit, or Execute. Select Query for data services.
* **Visibility**: The scope of the service audience. It can be either public (for public users), protected (for authenticated and authorized users only), or private (unpublished service for authenticated and authorized users only). Authentication is not required on the representational state transfer (REST) interface when a service is marked public. A private service will not be published to ENDS.
* **Service Status**: This is a flag that determines whether the service will be accessible or not. It can be either **Disabled** or **Enabled**. VES will reject service requests to a disabled service.
* **Workflow**: This is an optional filename of a workflow (XML file). The workflow will be executed when a request for the service is received. VES has a powerful default handler for Query and Solicit, so leave this empty if no custom processing is necessary.
* **Style Sheet**: This is an optional filename of an XLST stylesheet for transforming results. VES constructs a dataset XML document if no stylesheet is supplied. Otherwise, it will transform the dataset XML instance document using the supplied file. The parameter is not needed for Submit and Execute services.
* **Template**: This is a template file for constructing XML instance documents. VES will create XML documents based on the template, which contains data mapping and structural information, instead of the default XML dataset. The parameter is not needed for Submit and Execute services.
* **Data Source**: This is the name of a data source for the database server (See Section 5). The entry is required for Query and Solicit services but optional for Submit and Execute services.
* **SQL Statement**: The SQL query statement to be executed at runtime. A query statement may have parameters wrapped by ‘{$’ and ‘}’. An SQL statement must be supplied for Query and Solicit services but optional for Submit and Execute services. For example, using the image above, the query service has the following SQL Statement:

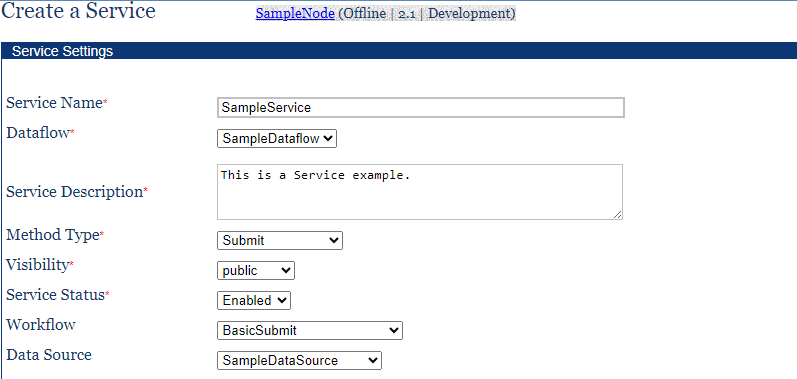
*SELECT \* FROM customer WHERE name like '{$CustomerName}’*

along with an associated parameter “CustomerName” in the Parameters section. When testing the service, a value can optionally be given to this variable.

* **Common Properties:** 
  + **HeaderTemplate**: The name of a document header within VESA to be used at the beginning of any document generated during the service request process.
  + **NotifyUsers**: Specifies email addresses to receive status updates about a service request, including notifications on completion, or errors.
  + **EmailTemplate**: Name of email template file. By default, "NotifyStatus" email template is used if no template stated.
  + **Compress**: Convert the generated document into a ZIP file (e.g., yes).
  + **LocalService**: Commonly found in execute services to retrieve information from other services, such as query services, gathering SQL statements to execute and gather requested data.
  + **Recipient**: Generally found for service requests that invoke Submit web methods. A user can submit a document to another user by specifying the recipient’s email address or node endpoint URL.
  + **Recipients**: Generally found for the Service/Task Report Transaction Summary to notify users upon completion that documents are ready for download.
  + **TargetEndpoint**: A URL that specifies the destination for routing requests within the VESA application.
  + **TargetDataflow**: The name of a valid dataflow.

As soon as the service is created and enabled, it will be made available to users. You can test the service using any EN clients or test tools.

To define a service for data submissions, the Method Type should be Submit. The following screenshot shows an incoming service that accepts any forms of document delivered to a VES:



The key element in the service definition is the workflow file name: BasicSubmit. This workflow performs the following operations when a document is received:

1. Create transaction and add audit records.
2. Sends a receipt to the submitter.
3. Archives the document and extracts the document headers.
4. Delivers the payload to a remote backend node if the recipient is a node URL address.
5. Notifies recipients of the document availability with transaction information.
6. Finalizes the transactions and informs the submitter of the transaction status.

You can use the workflow for any dataflow and accept submissions in any format. However, it does not load the payload XML document into a database server. The loading and transforming of data received to a database server is a separate task. Please see Appendix 9.1 for additional workflows and features.

## REST Interface

Once a service is created, it is published as a SOAP service and a REST service. The URL template is compliant with the Exchange Network recommendations. It is used for accessing query services as seen below:

https://ves.epa.gov/VES/nodedataservice?node={nodename}&dataflow={dataflowname}&request={serviceName}&paramName=paramValue&rowId={rowId}&maxRows={maxRows}&format=json|xml &token={token}

where:

* {nodename}: Is the name of a network node (i.e., the Node Identifier).
* {dataflowname}: Is the name of a dataflow.
* {serviceName}: Name of a service.
* All parameters are formatted as name-value pairs separated by &.
* {token}: A security token issued by NAAS. Token is optional for services that are marked as public.
* {rowId}: The starting row ID for fetching the result set.
* {maxRows}: The maximum number of rows to be fetched.
* The result format can be either JavaScript Object Notation (JSON) or XML.

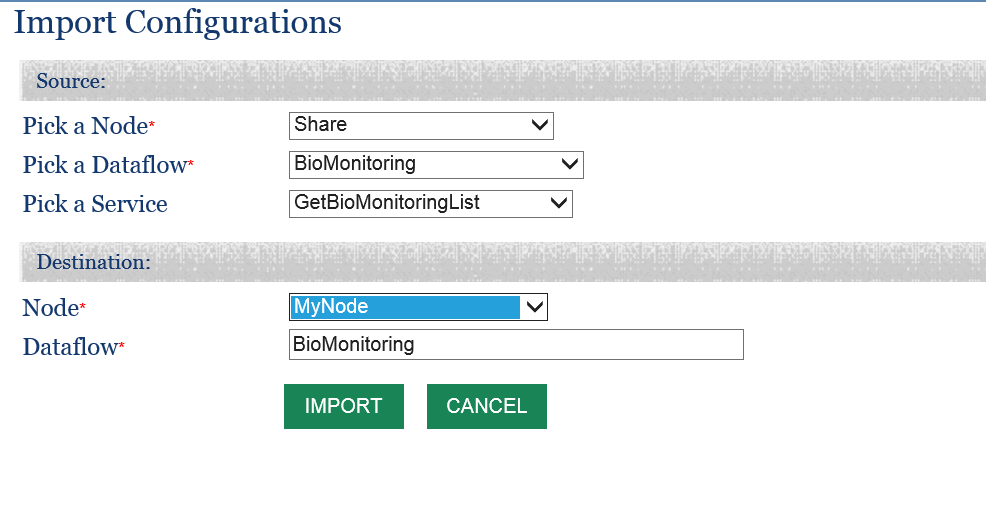
For example, the SampleService service, shown in Section 5.1, would have a REST URL similar to the following:

https://ves.epa.gov/VES/nodedataservice?node=SampleNode&dataflow=SampleDataflow&request=GetCountyCode&format=json&token=...

## Clone Services

To further simplify creation of services, the VESA provides a utility for copying services from other nodes. You can clone all services under a dataflow, or just a single service from another node.

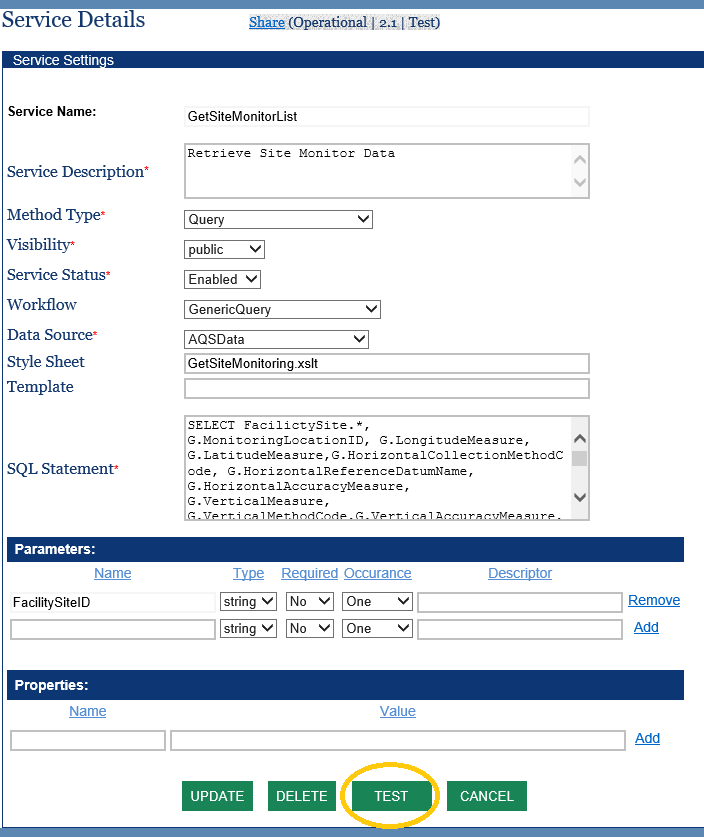
To access the tool, click on the ‘**Import Configurations**’ link, the following screen will be displayed:



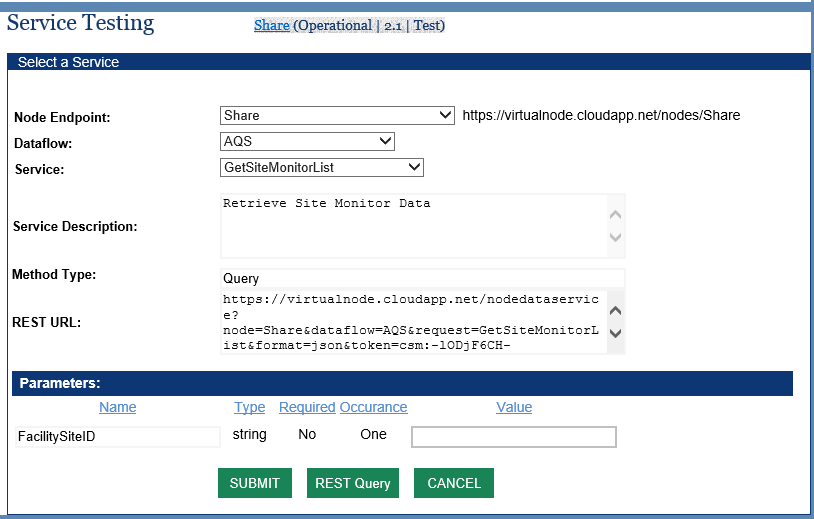
Please select a source node, dataflow, and a service name from the dropdown boxes, and then click the ‘**Import**’ button. The service will be copied to your node. If the name of a service is not selected from the source node, the tool copies all services under the dataflow. Note that the clone utility will not copy Data Sources from the source node for security reasons; the copied services need to be edited to use your own data source.

## Test Services

On the service detail screen (see screenshot below), there is a ‘**Test**’ button on the bottom of the page, which allows administrators to run a service directly and check out its status and results.



When clicking on the “**Test**” button, the service test page will be displayed as shown below:



Please make sure proper parameter values are provided in the Parameters Section and then click the “**Submit**” button. For Query services, the results will be displayed immediately. For Solicit, Submit, or Execute services, a transaction ID will be returned in a normal situation, which can be used to check status of the asynchronous operation. Note that even though VES returns a transaction ID successfully, the transaction may fail due to validation or business logic errors from CDX and EPA backend systems. You should check the transaction status and view final reports on the transaction detail page in a later time. (See Transaction Management in Section 8).

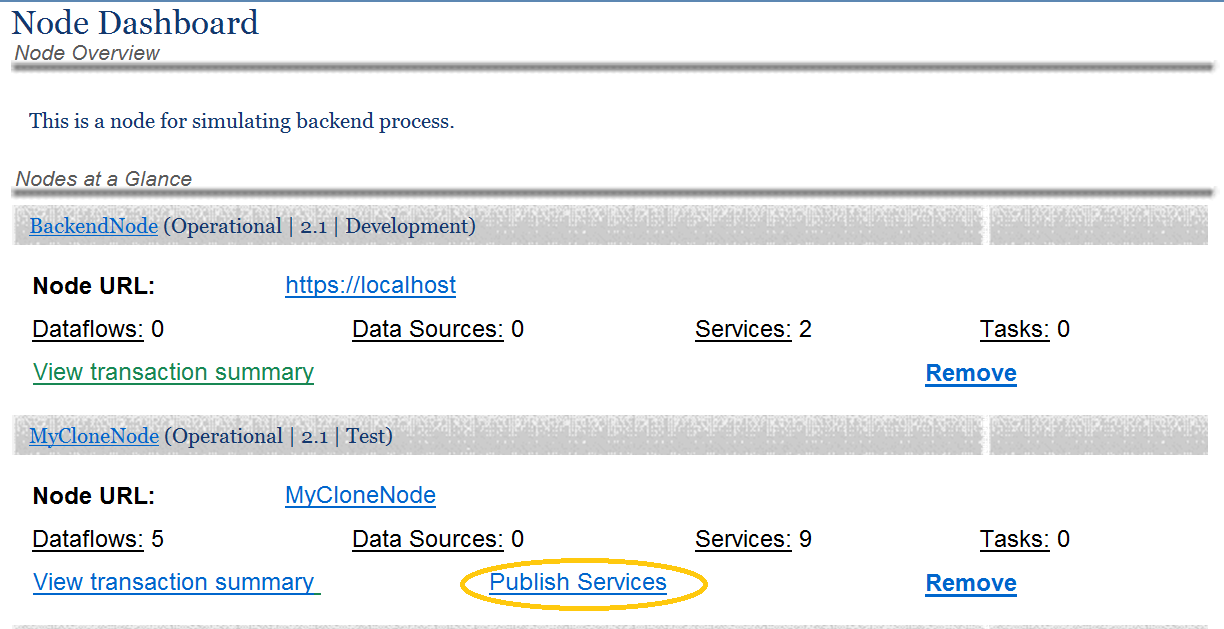
An error message will be displayed if there is something wrong in the service configuration.

## Publish Services

The VES is fully integrated with the ENDS. It supports both the push model (submit service definitions) or the pull model (GetServices from ENDS). However, pulling occurs on a scheduled basis at night. The owner of a VES can make services visible immediately by publishing them from VESA.

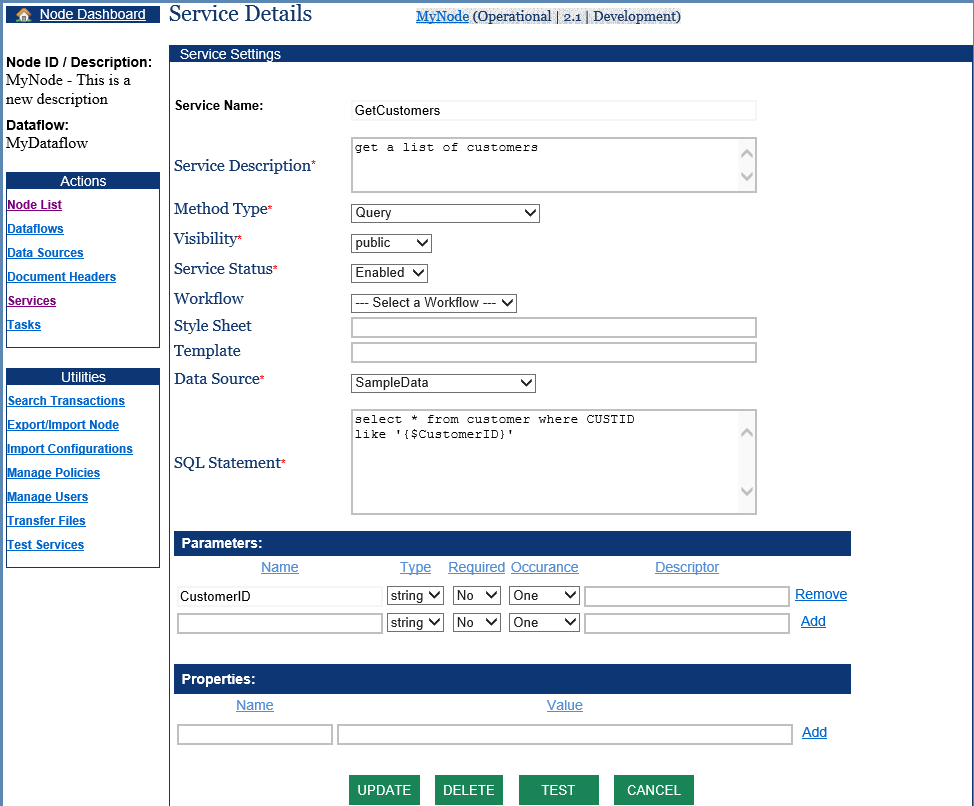
In order to publish node services into ENDS, you must be specifically authorized to load service definitions for a particular node. This is to prevent the administrator of one node from changing service definitions of another node. Please contact the Node Helpdesk at [nodehelpdesk@epa.gov](mailto:nodehelpdesk@epa.gov) for help setting up the publishing of services.

After logging on to VESA, the node dashboard will be displayed as shown below. The “**Publish Service**” link will be available if a node is enabled (not pending) and has a dataflow with at least one service under it. Clicking on the “**Publish Service**” link will trigger submission of service definitions to the ENDS server.



## Update Services

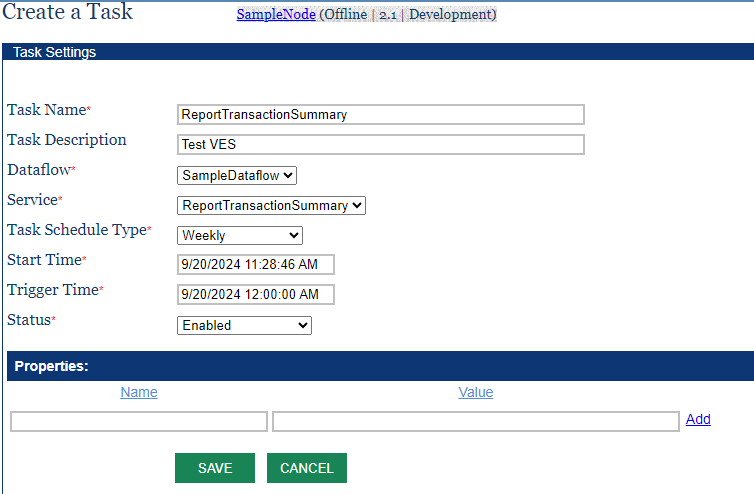
It may be necessary to make changes to your service definition after it is created. This is done by selecting Services on the left panel and then clicking on the service to be modified. The Service Detail screen, like the screenshot below, will be displayed:



# Task Management

A task is a scheduled operation that is executed by VES runtime engine. Tasks are commonly used to automate routine works. In VES, a scheduled task is always associated with an Execute service. When a task is triggered, the service will be executed.

A task can be created by clicking on the ‘**Tasks**’ link on the left panel of VESA. A screen like the following will be displayed:



The entries on the task form are explained below:

* **Task Name**: A unique name of the task. This should contain only alphanumeric characters without any whitespaces.
* **Task Description**: A brief description of the task.
* **Dataflow**: This is the name of an existing dataflow defined in the node.
* **Service**: The service to be executed. It should be a service for the Execute method.
* **Schedule Type**: This is the frequency of task execution. It can be Yearly, Quarterly, Monthly, Daily, Hourly, or Minutely.
* **Start Time**: This defines the starting boundary of the task.
* **Trigger Time**: This is time when the task will be executed.
* **Status**: This is a Boolean flag to enable or disable the task.
* **Properties**: A list of properties of the task. The properties are usually passed as parameters to the service. (See Transaction Report in Appendix 9.2.6)

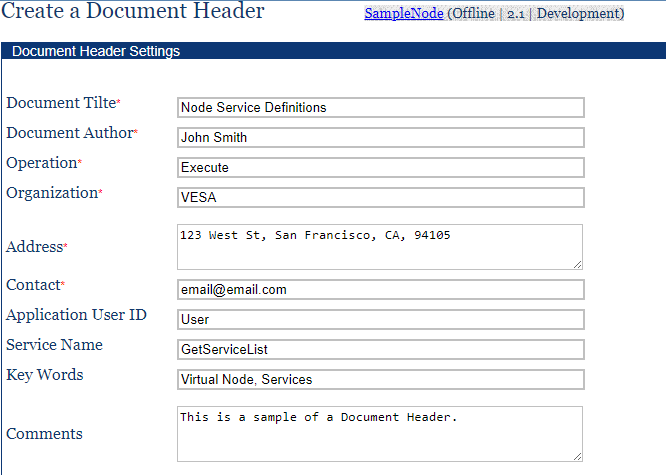
The example task shown above retrieves a list of transactions and constructs an XML document to be triggered at midnight (12:00AM).

When a task is triggered, the VES creates a transaction associated with service execution so that it can be audited and verified.

# Document Headers

Many of the EN dataflows require a document header for submissions. The VES offers a table to store document header information for each dataflow and uses the header information when constructing instance documents.

A document header can be created by clicking on the “**Document Headers**” link in the left panel of VESA, and then the “**Add New**” on the document header list page. The following screen will be displayed:



Please refer to the EN Document Header Specification if you need additional information on the fields.

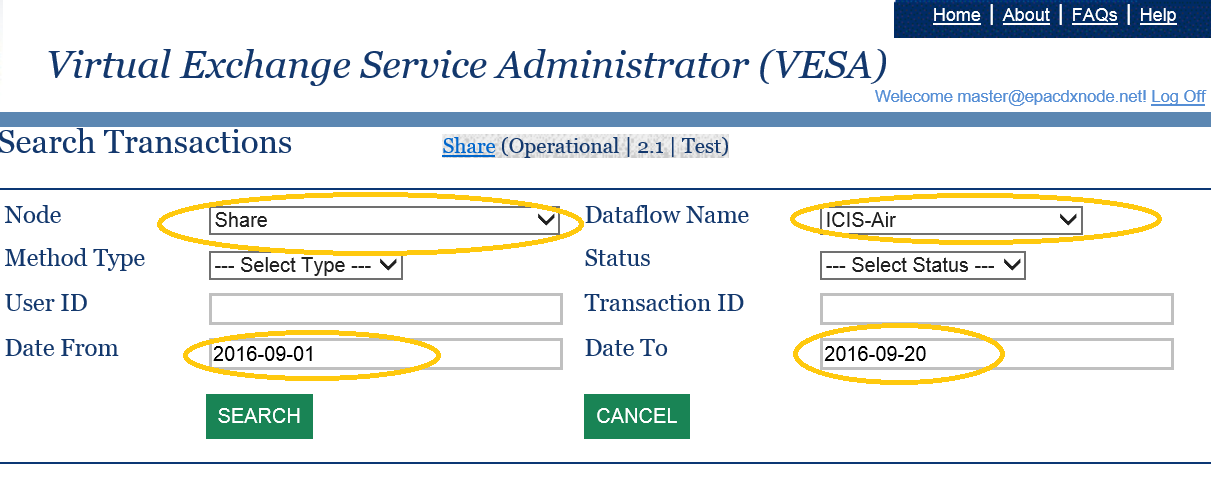
Once the header is saved, it can be used in the creation of constructing an XML payload. VES supplies a header template file which will be populated using the information supplied here.

A custom header template could also be used. This is done by adding a HeaderTemplate property with the name of the template file in the service definition. (See Create a Service in Section 5.1)

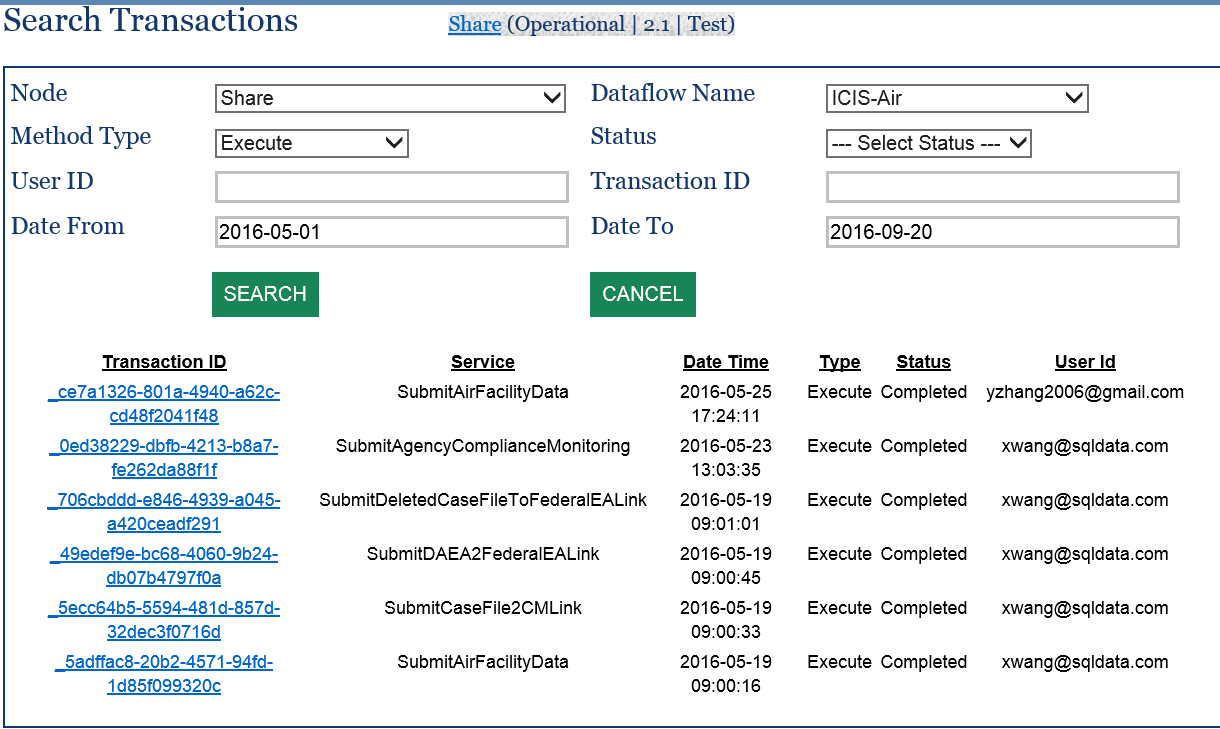
# Transaction Management

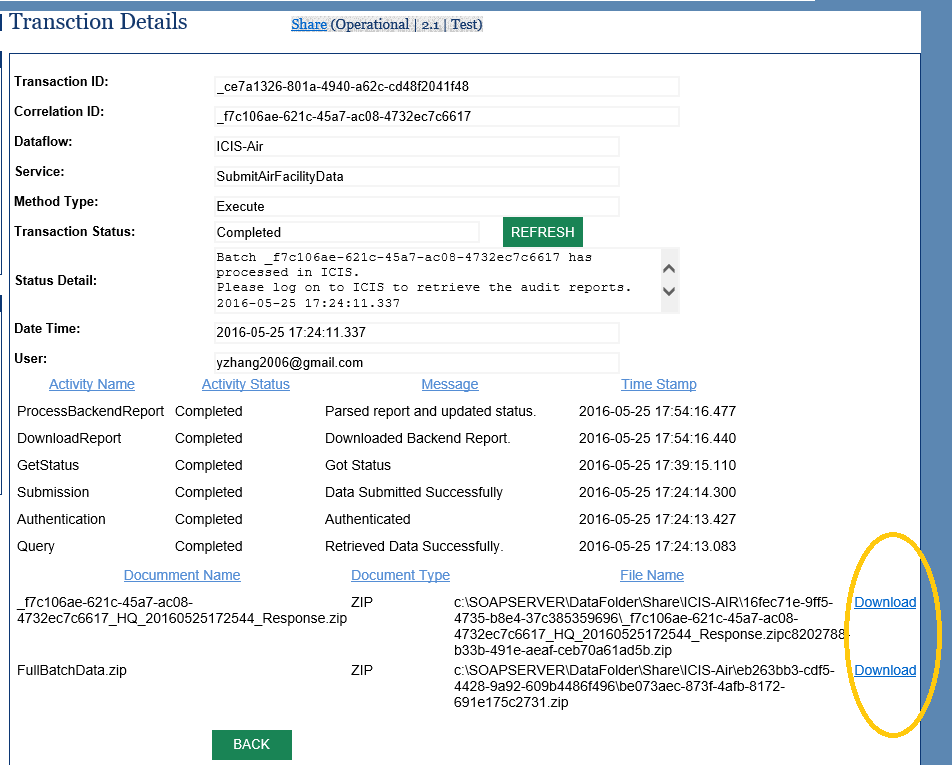
A node owner may check transaction status, view validation report and processing reports, or download the original documents. Here is how this can be done:

1. Click on the ‘**Search Transaction**’ link in the left panel. The transaction search form will be displayed.
2. Select the node and dataflow from the dropdown list.
3. Choose a period of time (“Date From” and “Date To”).
4. Click the ‘**Search**’ button.



A list of transactions meeting the search criteria will be displayed as shown in the screenshot below. Change your search criteria and try again if no result set is returned.





On the lower part of the screen, is a list of files (processing reports, validation results, and original documents) that can be downloaded and examined if something went wrong.

If the transaction status is still ‘Pending’ after a long period of time. You may click on the ‘**Refresh**’ button next to the transaction status. This will force the system to retrieve transaction status from CDX again and process the results if available.

It is possible for a transaction to fail when working with a large amount of data at a given time. The general recommendation is to keep these transactions below a few 175MB to ensure data flows smoothly and without any risk of a timeout occurring.

If you have a large amount of data to submit and suspect this may be the reason a transaction is failing, there are a few ways you can reduce the size within VES. One method is to specify parameters in the services used by your node. By restricting your transactions to specific dates or other criteria such as Transaction ID, Submission ID or Facility ID, it can reduce the size of the dataset. For services without parameters, partners can modify the SQL statement in the service by adding "WHERE" clauses to limit the data returned by the query.

These are just a few suggestions that can be used to reduce the size of a large dataset in VES. If there is any question regarding these recommendations, you can reach out to nodehelpdesk@epa.gov to request additional guidance.

# Appendix

## Retrieve Transaction Information

Node administrators may obtain transaction information using the VES administrative platform at <https://ves.epa.gov/VESA>. However, if you wish to integrate the VES transaction information with your application through web services, this section provides detailed information on how this could be accomplished.

Your node supports two web services for retrieving transaction information by default: GetTransactionList and GetTransactionDetail. These are services under the Query method defined in the latest Network Node Functional Specification.

The query method is defined as follows:

<element name="Query">

<complexType>

<sequence>

<element name="securityToken" type="xsd:string"/>

<element name="dataflow" type="xsd:NCName" />

<element name="request" type="xsd:string" />

<element name="rowId" type="xsd:integer"/>

<element name="maxRows" type="xsd:integer"/>

<element name="parameters" type="typens:ParameterType" minOccurs="0" maxOccurs="unbounded"/>

</sequence>

</complexType>

</element>

<element name="QueryResponse" type="typens:ResultSetType"/>

It requires the following parameters as shown in the XML schema:

* **securityToken**: A security ticket issued by the service provider or a trusted security provider. This is the token received from NAAS after successful authentication using your account.
* **dataflow**: The name of the dataflow. It should be **admin\_v1\_0** for transaction information retrieval.
* **request**: The database query to be processed. This could be **GetTransactionList** or **GetTransactionDetail**.
* **rowId**: The starting row for the result set - it is a zero-based index to the current result set. The value of rowId must be “0” if paging is not requested.
* **maxRows**: The maximum number of rows to be returned. Valid values are any number greater than zero. Please use 200 for maximum efficiency.
* **parameters**: An array of zero or more ParameterType structures for the information request. For GetTransactionDetail, a transaction ID (string type) is required. The GetTransactionList service accepts the following parameters (filters):

| **Parameter Name** | **Parameter Type** | **Comments** |
| --- | --- | --- |
| UserName | String | The user who initiated the transaction. |
| Dataflow | String | The name of the dataflow. |
| Status | String | The transaction status, it may be Completed, Failed, or Pending. |
| Method | String | The web method used for the transaction. Common web methods are Execute, Submit, Solicit, or Query. |
| fromDate | Date | Starting date for search transaction in a time period. |
| endDate | Date | Ending date. For transaction search in a time period, but fromDate and endDate must be supplied. |
| NodeId | String | Your node identifier. |

The request message should be sent to your node. For example, in Production, it would be at the following address:

https://ves.epa.gov/VES/nodes/{NodeId}

where {NodeId} is your node’s unique identifier. If successful, the VES node will respond with an XML containing the requested transaction information.

Here is an example message for GetTransactionDetail service:

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope" xmlns:tns="http://www.exchangenetwork.net/wsdl/node/2" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:typens="http://www.exchangenetwork.net/schema/node/2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" >

<SOAP-ENV:Body>

<typens:Query xmlns:typens="http://www.exchangenetwork.net/schema/node/2">

<typens:securityToken>csm:-lODjF6CH-uOjxQh\_qAr3IjOfVTJkgx5h\_,,.</typens:securityToken>

<typens:dataflow>admin\_v1\_0</typens:dataflow>

<typens:request>GetTransactionDetail</typens:request>

<typens:rowId>0</typens:rowId>

<typens:maxRows>10</typens:maxRows>

<typens:parameters parameterName="TransactionId" parameterType="String" parameterEncoding="None">\_f92830f7-88d5-44c4-93c3-798dbcf72f56</typens:parameters>

</typens:Query>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

A positive response would be something like this:

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

<SOAP-ENV:Body>

<QueryResponse xmlns="http://www.exchangenetwork.net/schema/node/2">

<rowId>0</rowId>

<rowCount>1</rowCount>

<lastSet>true</lastSet>

<results format="XML">

<TransactionDetail xmlns="http://www.exchangenetwork.net/schema/tts/1">

<TransactionId>\_f92830f7-88d5-44c4-93c3-798dbcf72f56</TransactionId>

<NodeId>share</NodeId>

<DataflowName>BeachNotification</DataflowName>

<ServiceName>ProcessBeachNotificationData</ServiceName>

<TransactionType>Execute</TransactionType>

<TransactionStatus>Pending</TransactionStatus>

<StatusDescription>Submit processed successfully. 2016-08-19 10:38:43.480</StatusDescription>

<CorrelationId>\_071a885d-ed20-4afe-8cbf-aaeb6f359cc7</CorrelationId>

<CreationDateTime>2016-08-19 10:38:43.480</CreationDateTime>

<UserId>jdoe@example.com</UserId>

<ActivityList>

<Activity>

<ActivityName>Submission</ActivityName>

<ActivityStatus>Completed</ActivityStatus>

<ActionDescription>Data Submitted Successfully</ActionDescription>

<Message>Data Submitted Successfully</Message>

<TimeStamp>2016-08-19 10:38:47.750</TimeStamp>

</Activity>

<Activity>

<ActivityName>Authentication</ActivityName>

<ActivityStatus>Completed</ActivityStatus>

<ActionDescription>Authenticated</ActionDescription>

<Message>Authenticated</Message>

<TimeStamp>2016-08-19 10:38:46.440</TimeStamp>

</Activity>

</ActivityList>

<DocumentList>

<Document>

<DocumentId>\_ad09ad80-18ac-498a-b54d-f205ee504875</DocumentId>

<DocumentName>Data.zip</DocumentName>

<FileName>share\BeachNotification\fc32a6cb-4472-4fc4-bf4a-4391a885850c\cec215d5-62e6-4bd7-a5d8-f071e7823758.zip</FileName>

<DocumentType>ZIP</DocumentType>

</Document>

</DocumentList>

</TransactionDetail>

</results>

</QueryResponse>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

## Information Exchange Patterns and Common Workflows

VES leverages a workflow foundation technology for orchestrating complex business processes from a very high level. The VES workflows are constructed based on typical data exchange patterns commonly used in the EN. Each workflow is customized to address a set of processes for partners when executing a service.

The following subsections describe existing workflows, what they do, and when to use them. We welcome suggestions for new assembly of workflows for common application patterns. Please contact the Node Helpdesk with your business case and requirements; we will create new workflows as needed.

### Basic Submission Workflow

#### Description

This is a very general workflow, which allows submissions of data to any dataflow in any format. The workflow name is **BasicSubmit**.

#### Operations

The workflow performs the following operations:

1. Create transaction and add audit records.
2. Send receipt email to the submitter.
3. Archive the document and extract document headers.
4. Deliver the payload to the remote backend node if the value in recipient is a node URL address.
5. Notify recipients of the availability of the document with transaction information.
6. Set transaction status to “COMPLETED” and inform the submitter of the new status of the transaction.

#### Parameters

None.

#### Usage

The workflow can be used in situations where data validation is either unnecessary or infeasible (non-XML). The payload of data submissions may have undetermined format. The workflow that is selected from the drop-down list on the service form is **BasicSubmit.**

#### Variants

* **Submission service for loading Toxics Release Inventory (TRI) data from CDX**. The workflow name is **LoadTRIDoc**. Responsible for loading a TRI document and submitting to a target Network Node.
* **Submission service for loading BUOY data**. The workflow name is **LoadBUOYData**. Responsible for the loading and submission of a BUOY Data Report via a comma separated values (csv)/text (txt) file.

### Generic Submission Workflow with Validation

#### Description

This is a workflow that not only handles the regular submission processing, but also tries to validate the document using the QA services. The workflow name is **GenericSubmit**.

#### Operations

The workflow performs the following operations:

1. Creates a transaction and adds audit records.
2. Sends a receipt to submitter.
3. Archives the document and extracts the document headers.
4. Validates the document using the QA services, terminates the transaction if validation fails, otherwise proceeds to next step.
5. Delivers the payload to remote backend node if the recipient value entered in the form is a node URL address.
6. Notifies the recipients of the availability of the document including the transaction information.
7. Finalizes the transactions and informs the submitter of the status of the transaction.

#### Parameters

None.

#### Usage

The workflow can be used for processing XML submissions where schema or schematron data validation is necessary.

### Generic Query

#### Description

This workflow invokes a Query service on a VES and then transforms to XML document with specific XML schema defined with dataflow. The workflow name, which is selected from the drop-down list on the service form, is **GenericQuery**. This query service utilizes Java Database Connectivity (JDBC).

#### Operations

The workflow performs the following operations:

* Invokes the Query Service against a network node.
* Transform query results into a strong typed XML file, either complied with specific XML schema defined in dataflow or EN dataset.

#### Parameters

* User defines query parameters based on query conditions in Service Configuration User Interface (SCUI).

#### Usage

The workflow can be used for delivering your database information to another network node in XML.

### Local Query and Submit

#### Description

This workflow performs a database query operation against the local node and then submits the resultant XML document to a remote node. The workflow name that is selected from the dropdown list in the service form is **QueryThenSubmit**.

#### Operations

The workflow performs the following operations:

* Performs the database query and retrieves result set.
* Constructs an XML instance document.
* Submits the XML document to the target node, retries at least three times if failures occur.
* Finalizes the transaction status.
* Notifies the owner of the status of the transaction.

#### Parameters

* User defines query parameters based on query conditions in SCUI.

#### Usage

The workflow can be used for delivering database information listed in your data sources to a remote node.

#### Variants

There is a couple of specialized workflows based on QueryThenSubmit. They are:

* **SubmitAQSData:** The workflow is customized for supporting **Air Quality System (AQS), Water Quality Exchange (WQX), Assessment and Total Maximum Daily Load Tracking and Implementation System (ATTAINS),** and others. It performs a local query operation (constructing an XML instance document), submitting to a target Network Node and then download processing report.
* **SubmitMultiPayloadICIS:** It is specifically designed for ICIS-AIR data submissions, which performs a local query, data submission, backend report processing.

### Generic Solicit

#### Description

This workflow handles solicit request to the node. It performs specified database services in an asynchronous mode and then updates transaction status as needed. The workflow name that is entered into the service form is **GenericSolicit**.

#### Operations

The workflow performs the following operations:

* Performs the database query and retrieves the result set.
* Constructs an XML instance document.
* Updates the transaction status (Completed or Failed).
* Notifies the user about the status of the transaction.

#### Parameters

* User defines query parameters based on query conditions in SCUI.

#### Usage

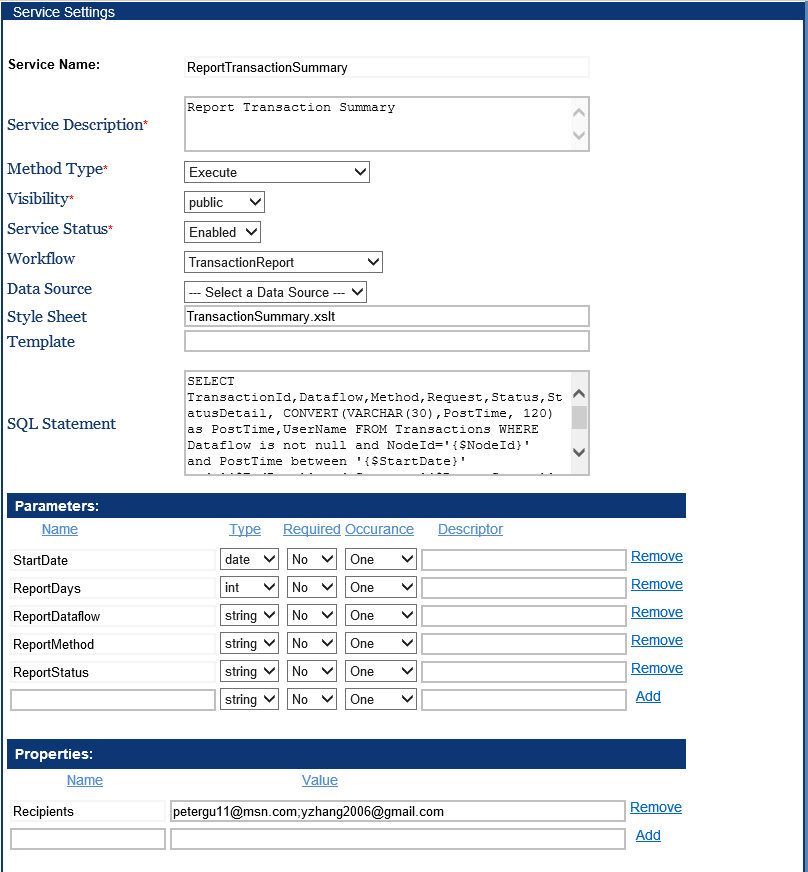
* The workflow can be used for processing any Solicit requests.

### TransactionReport

#### Description

This workflow retrieves a list of transactions, formats the results into Hypertext Markup Language (HTML) format and sends a document notification to specified recipients to visit and download at the specified URL. This is typically used with a scheduled task to send transaction summaries to flow or node administrators on an hourly, daily, weekly, monthly or yearly basis.

The node administrators should use (import) the ReportTransactionSummary service under the Report dataflow from the Share node. The following screenshot shows the definition of the service:



#### Operations

The workflow performs the following operations:

* Perform a query operation of the specified node using supplied parameters.
* Transform the XML results into HTML document.
* Send the formatted HTML document to specified recipients.

#### Parameters

The ReportTransactionSummary service accepts the following parameter:

* StartDate: The beginning date for the report period. This parameter should be supplied either manually or by a Task scheduler.
* ReportDays: How many days to be included in the report starting from the StartDate.
* ReportDataflow: The name of the dataflow for the report. All data flows will be included if the parameter is not supplied.
* ReportMethod: The node method associated with transactions. For example, if you are interested in data submitted to the node, ReportMethod should be Submit. Transactions of all method will be included in the report if ReportMethod is not supplied.
* ReportStatus: The transaction status to be used in filtering transactions. For example, if you wish to show only failed transactions, ReportStatus should have a value of “Failed”.

#### Usage

The workflow can be used for generating transaction reports automatically. To use the workflow in automatic reporting:

1. Create a ReportTransactionSummary service or import the service from the Share node. This greatly reduces the complexity of setting up SQL queries.
2. Create a Task to run the service on a scheduled basis. The following screenshot shows a sample scheduler which runs the service every Monday at 4:00pm starting from April 13, 2015

