



ShowCase

Data Exchange Services (DES)

Copyright

Copyright © 2020
ShowCase
equivant
2290 Lucien Way, Suite 330
Maitland, FL 32751

Legal Disclaimer

The information contained in this document describes confidential information about equivant's services and methodology. Receipt of this document constitutes acknowledgment that this information is proprietary to equivant. Recipients of this document will protect and not disclose the contents herein (and any copies thereof); including said confidential information, using the same procedures and requirements by which they protect their own proprietary and confidential information.

Recipients will not, in whole or in part, disclose any of the proprietary or confidential information to any person, firm, corporation, association, or other entity for any reasons or purpose whatsoever, nor shall they make use of any proprietary or confidential information for their own purposes or benefit, without prior express written consent of equivant. Recipients will not knowingly make this document or the information contained herein available, in whole or in part, to current or potential competition, or to competitors of equivant, or to other organizations unrelated to recipients.

Contents

Introduction	3
Data Exchanges	4
DES Architecture	6
Exchange Point Service (EPS)	6
Exchange Worker Service (EWS)	7
Centralized Log Monitoring	9
DES Workflows	10
Job Scheduling	12
DES Dashboard	13
Summary	15

Introduction

ShowCase Data Exchange Services, also known as DES, is a framework for building services to interface with software applications internal and external to an organization. The tools provided by the framework eliminate the need to explicitly create and manage web services. It alleviates developers and administrators from the task of maintenance and management of web services in an organization.

DES is implemented as a generic system that can be used by any enterprise wide application to efficiently implement and manage a large number of services in an organization. It is built on top of familiar web server technologies - IIS, ASP.NET, Web API and .NET framework. DES provides several ready to use tools, making it very easy to implement and manage a service oriented application.

DES can be used to build any web service. Unlike traditional web services, DES provides a unique way of building the service and has distinct advantages:

- **Powerful Workflow Engine** - Zero or minimal coding required to implement the logic in the service. This is achieved via Windows Workflow Foundation of .NET framework. With WF almost all programming logic can be expressed as xml using an intuitive graphical interface.
- **Centralized Logging** – Any log activity in the service is recorded to a centralized database. This database is used by the DES dashboard to query and analyze service data.
- **Flexible Contracts** – The clients of a service built using DES need not use service contracts. Service methods can be accessed via simple HTTP GET and POST calls. The data that is sent to and from a service is in XML format.
- **Scheduling** – Service jobs can be scheduled to run at specific times or on demand.
- **Reprocessing** - A service execution can be set to automatically reprocess any number of times when it fails. It can also be reprocessed manually.
- **Process isolation** - A number of different services can run in parallel in separate processes.
- **Scaling** – Services in DES can be scaled dynamically to handle various load patterns.
- **Notifications** – DES provides automatic email notifications to personnel on life cycle events of a service execution.
- **File Transfers** – DES provides file transfer capabilities using FTP and SFTP.
- **Dashboard** – DES provides a dashboard for live monitoring of service executions, query logs, manage services etc.
- **Security** – DES services can utilize ShowCase authentication system to control access. Alternately, a DES service can be configured to run in Anonymous mode.
- **Fire and Forget** – A service can be configured to run in the background. Clients can start the service execution, and later receive notifications of the execution result. Clients can also receive pre-defined status updates during the execution of a long running service.
- **High Reliability** – Service executions inside DES are highly reliable and fault tolerant. Services are extensively logged and provide data for analysis and reporting of every execution.

Data Exchanges

Data Exchange is an abstract term which we refer to as a service endpoint. In the world of DES, a service is defined as a Data Exchange. Throughout the rest of this document, the term *Data Exchange* or simply *Exchange* will be used instead of *Service*.

A data exchange must be expressed as an uri template. For example, consider the uri below:

/GetCitationInformation?CitationNumber={CitationNumber}

In this uri, *"/GetCitationInformation"* forms part of a unique url in DES which contains the logic to provide citation data. This data exchange accepts a parameter called *CitationNumber*. To invoke this exchange, a HTTP GET call can be placed to DES via the browser or any client code to the url below:

<http://.../EPS/ExchangePointService.svc/GetCitationInformation?CitationNumber=A08ERPE>

"/EPS/ExchangePointService.svc" in the above url is the address of a service called Exchange Point Service, or EPS for short. This web service is the entry point of the DES system. All exchanges defined in DES can only be invoked by using the EPS service. EPS will be explained in more detail in a later section of this document.

A data exchange has several other properties. Most important of them is called *IsRealTime*. This is a simple true/false property that governs the behavior of the exchange in DES environment.

If an exchange is defined as real-time, invocation of the exchange is synchronous. This means that the caller of the exchange will get the result back after execution ends. This behaves like a traditional web service call. The EPS service executes the logic defined for this exchange.

Alternately, if the exchange is defined as Non-Real Time, any invocation of the exchange will return immediately to the caller. The data returned to the caller is only an acknowledgment and not the execution result. This acknowledgment contains a *Request ID* (a GUID), that represents the unique identifier for this specific request to DES. The *Request ID* can later be used by the caller to query DES for execution status or the final result.

A non-real-time exchange is ideal for long or intensive executions. The actions it performs can execute in an isolated process and there is no need to return an immediate result to the caller. As an example, consider the scenario where charge information on a case must to be transmitted to a justice partner such as Public Defender or State Attorney's Office. When a Charge record is created/modified in ShowCase CMS, a DES request is issued using a uri template like *'/PD/SendCharge?ChargeID=12345'*. ShowCase isn't required to wait for a response from the exchange. It just needs to request the exchange in a fire-and-forget fashion. If the request fails, DES can re-process the exchange at a later time without the need to involve ShowCase.

The table and figure below describes all properties of a data exchange.

Property	Description
Exchange Name	A friendly name for the exchange.
Exchange Description	A brief description of what the exchange does.
UriTemplate	A unique uri template path which can be used to access the exchange.
IsRealTime	Indicates if an exchange must be executed in real-time mode.
ProcessWorkflowID	A link to the workflow xml file that contains the implementation of the data exchange. You will learn more about implementing a workflow for a data exchange later in this document.
Priority	The priority of an exchange determines which requests get processed by the DES system first. This only applies to exchanges that are configured as non-real-time. Real time exchanges are processed synchronously by the EPS service, therefore Priority does not apply to them.
Worker Address	For non-real-time exchanges, DES allows the execution of an exchange request on a specific server or a service instance.
Anonymous	This is a true/false property. If false an exchange can be accessed only upon successful authentication token evaluation. Alternately, it can remain openly accessible without requiring any authentication. Please note that since an exchange is nothing but a URI request, all security parameters of the web server or the firewall still apply.

Create Data Exchange

Name

Description

UriTemplate

☐ Real Time

ProcessWorkflow

<NewWorkflow> ▼

Priority

Anonymous

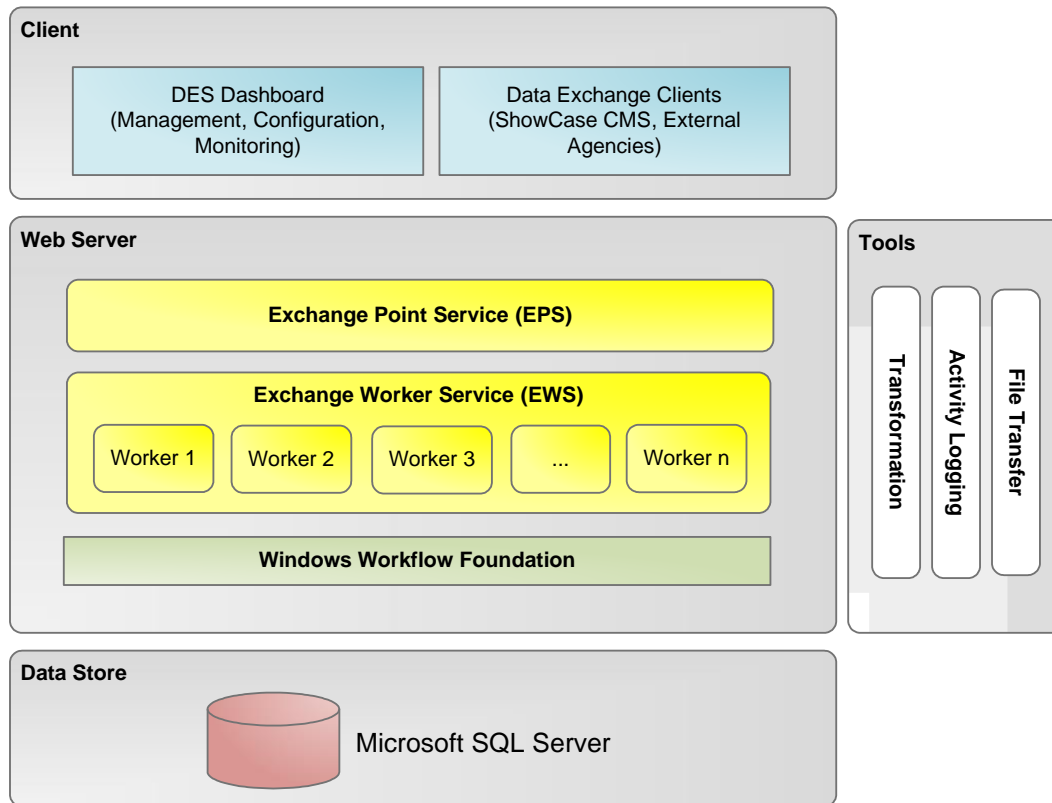
Not Set ▼

WorkerAddress

Create

DES Architecture

The figure below depicts the general architecture of the DES system.



DES Architecture Diagram

Exchange Point Service (EPS)

EPS is a standard ASP.NET web service that serves as an entry point into the DES system. This means that any exchange defined in the system can only be accessed via this service.

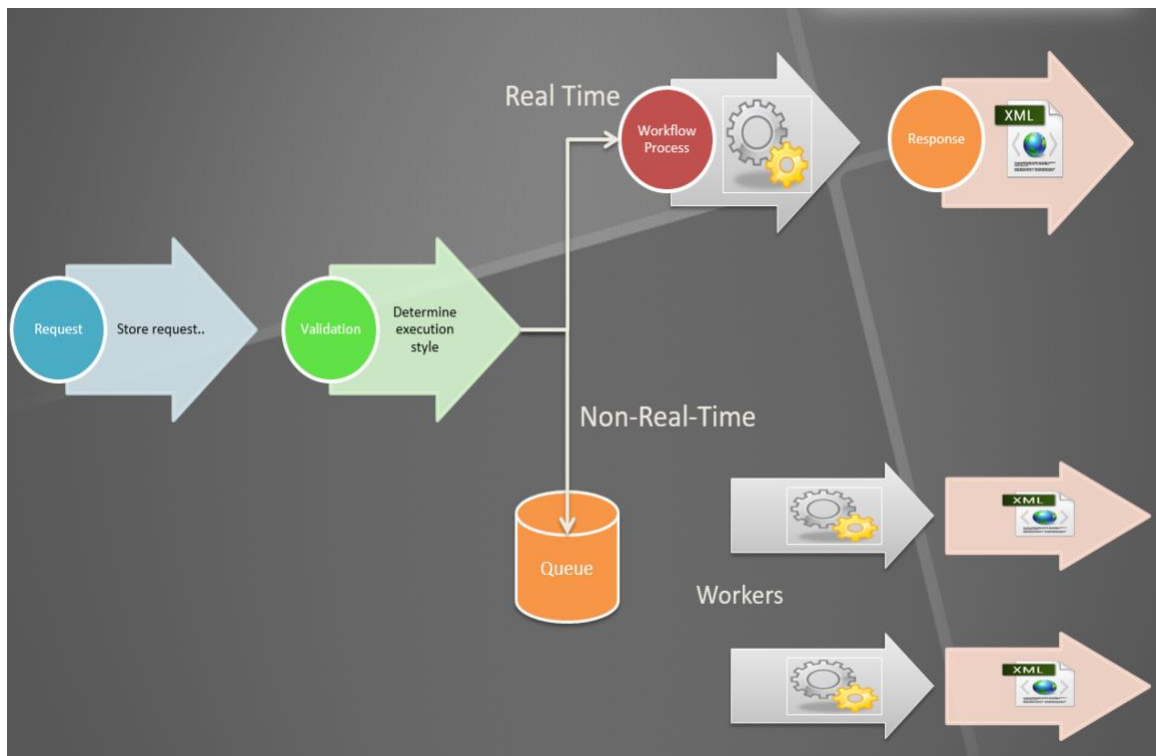
The life cycle of an exchange begins with EPS and it is responsible for following sequence of steps when a request is received:

1. Creates an entry into a request log table – *tblDataExchangeTransactionLog*.
2. Evaluates if the exchange is configured as real-time or non-real-time.
3. If it is a real-time exchange, EPS processes the request, by executing the workflow, and returns the result to the caller. If it is a non-real-time exchange, EPS sends an acknowledgment that contains *Request ID* for the request received. This request is now staged for processing by a background worker service called *Exchange Worker Service*, or *EWS*. The worker service's

responsibility is to pick up these pending requests periodically (a configurable interval) and process them.

Several instances of the EWS service can be installed on the web server(s). Exchanges can be configured to execute inside a specific instance of this service by specifying its address in the *Worker Address* property.

See the diagram below which provides a general overview of an exchange's life cycle.



Data Exchange life cycle

Exchange Worker Service (EWS)

EWS is also a standard ASP.NET web service that is used to process all non-real-time exchange requests. Several instances of this service can be installed on a web server or a collection of web servers to pair with a vertical or horizontal scaling strategy.

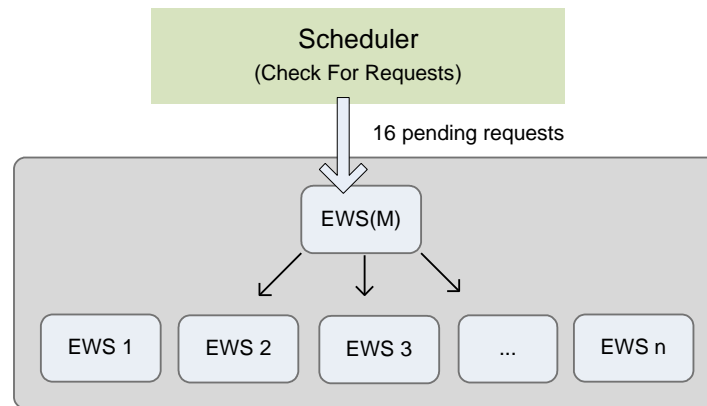
A set of EWS instances can form a logical work group that process exchanges related to a specific vendor or interface. As an example, consider the interface for Electronic Recording of documents. Such interface can typically have the following exchanges defined:

- Exchange 1 – Submit document for e-recording.
- Exchange 2 – Retrieve and process the recorded document by polling the vendor service for all documents that are pending recording. Some vendors may have the functionality to notify us via

service when recording is complete. In such cases, this exchange would simply be a service endpoint that the vendor can call.

In DES, both Exchange 1 and Exchange 2 can be setup to execute at specific EWS instance such as <http://.../EWSERecord/ExchangeWorkerService.svc>. Alternately each of these exchanges can be setup to execute on their own instances of EWS. This form of configuration provides a well-defined process isolation. Various exchanges can execute in a server environment in isolation without competing for resources from a single process.

When several EWS worker instances are created in DES, one of those instances must act as a manager. See figure below that illustrates this concept:



EWS Manager and Workers

The scheduler component identified in the above diagram is a simple SQL Server Agent Job that runs every 15 seconds. Its responsibility is to pick up 'x' number of waiting non-real-time requests and submit them to the EWS Manager (EWSM). The number of waiting requests (*16 by default*) that can be picked up and the frequency (*15 seconds by default*) at which they get picked up can be fine-tuned based on the capacity of web servers.

EWSM will perform one of the two following actions:

- If the pending request's Exchange is configured with a worker address, it will delegate/forward the request to the specified worker.
- If the request's Exchange is not configured with a worker address, it processes the request.

To setup a EWS instance as manager, set the flag *IsManager* to true in the configuration file of the service.

Centralized Log Monitoring

The first thing EPS Service does is store the request received in a log table called (*tblDataExchangeTransactionLog*). The request details are stored in the *RequestData* column as XML data. When the request is processed by a worker or EPS, the result of the execution is stored in *ResponseData* column of the same table. The address of the actual worker instance that processed the request is stored in the *WorkerAddress* column of this table. The *StateID* column of this table depicts the state of a request during its life cycle. Some of the state id values are:

State ID	Description
7	Waiting – request is waiting to be sent to a worker
14	In Progress – Request execution is in progress.
18	Complete - Request processing is successfully completed.
17	Error – There was an error during request processing.

One of the advantages of maintaining the state id of a request is that it provides the ability for a request to be re-processed. If a request execution fails, and is in state id of 17, to reprocess the request, switch the state id to 7.

The DES dashboard provides tools to re-process failed requests and cancel in-progress requests.

The *tblDataExchangeTransactionLog* table provides a ledger of all incoming requests, the request and response details, time and duration of execution and the worker responsible for processing the request.

Execution of a request could produce significant log activity. Aside from general purpose logs that the DES produces during a request lifecycle, developers can provide logging of their own in the workflow as Activity Tracing, Verbose, Errors, Warnings, Milestones etc. All of these log statements associated to a specific request are stored in an event table called *tblDataExchangeTransactionEvent*. Every event in this table is associated to a specific RequestID from the log table.

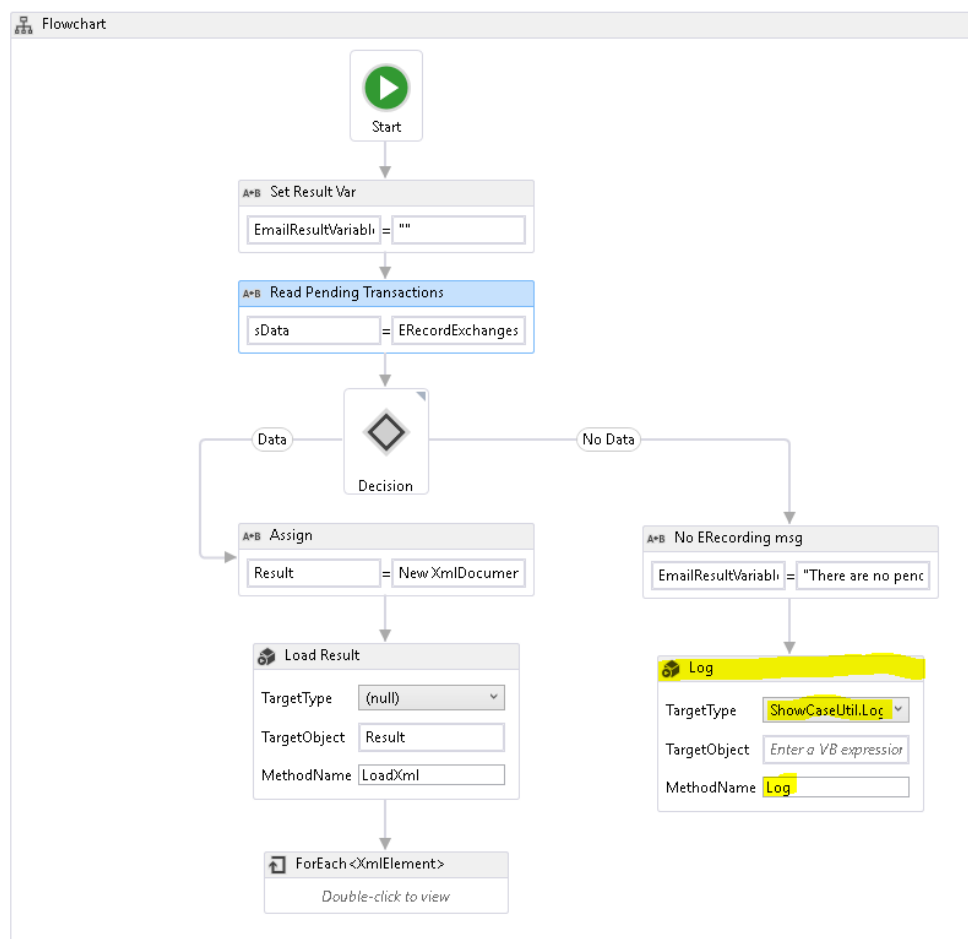
The Log and Event tables serve as a centralized data repository used for query, analysis and live monitoring of exchanges running in the DES environment.

DES Workflows

To implement the logic of an exchange, a workflow must be built using Windows Workflow Foundation (WF) provided by Microsoft .NET Framework. WF provides a workflow designer in Visual Studio that must be used create and edit a workflow. The designer allows creation of a workflow as a flow chart or sequential flow with conditions, loops and other programming constructs. It also provides compile time checking similar to any other code editor.

Almost all of the logic that can be expressed in code, can be expressed in the workflow designer. The workflow file itself is a simple XAML (based on xml) file that is not compiled into binary. WF provides an execution engine that can compile and execute these XAML files at run time.

Figure below shows a sample workflow that is designed as a flowchart.



Workflow for an exchange

Just like C# or Java code, these workflows can be used to create service clients and execute external web services. If an exchange demands more significant and complex logic, it can be implemented in standard

.NET code using c#, vb.net etc and compiled into an assembly. This assembly can then be referenced in the designer and used to build the workflow.

For example, in the screenshot above, see the highlighted block. That is an *Invoke Method* block of the designer that can execute a method in an external assembly by specifying the class name, method name and any parameters required.

The act of processing a request by EPS or EWS is executing the workflow defined for an exchange.

A Workflow is a XAML file which is un-compiled and contains the logic of an exchange. Workflows are stored in the database in a table called *tblProcessWorkflow*, and linked to a data exchange.

If the logic of an exchange needs to be modified, a developer would need to modify the workflow using the WF designer and upload the workflow into the database. There is no need for any compilation of source code and release deployment of an application.

Modifying the logic for a well-designed data exchange is a configuration change.

When a new interface/exchange is required, developers (*equivant, or the customer*) will perform following steps:

1. Define the data exchange in DES and set its properties such as *UriTemplate, IsRealTime, Worker Address, Anonymous etc.*
2. Implement a workflow that contains the logic for the exchange.
3. Import the workflow and link it to the new exchange.

DES dashboard provides tools to create new exchanges, import workflows and link an exchange to a workflow. Once an exchange has been defined, the dashboard can be used to export this entire setup and definition to be imported into a different environment.

Job Scheduling

Some exchanges need to run periodically without any user intervention. For example, *CCIS3 Push* interface. This interface is responsible to capture recently modified case data and push it to Florida State's FCCC system periodically. This exchange is implemented using a workflow that captures all case data that is modified within the last 'x' minutes and submits it to FCCC's web service. It can be executed using a simple url: <http://.../EPS/ExchangePointService.svc/CCIS3Push?<parameters>>

To execute this exchange periodically is a simple task of using your favorite task scheduler that can execute http requests. In this instance a SQL Server Agent job is defined with a schedule to run every 15 mins of every day. The job itself runs a PowerShell script that invokes the url above.

```
# Call CCIS3 Push Interface

$web = [Net.WebRequest]::Create("http://js-ml-qa/EPS/ExchangePointService.svc/CCIS3Push?_EmailResultsTo=admin@company.com")

$webResponse = [Net.HttpWebResponse]
$webResponse = $web.GetResponse()

$responseStream = [System.IO.StreamReader]
$responseStream = $webResponse.GetResponseStream() -as [System.IO.StreamReader]
$responseStream.ReadToEnd()
```

SQL Job with PowerShell

To schedule an exchange to run at periodic intervals, create a SQL Server Agent job. The job will need to execute a power shell (or other scripting language) script that will invoke the exchange URL.

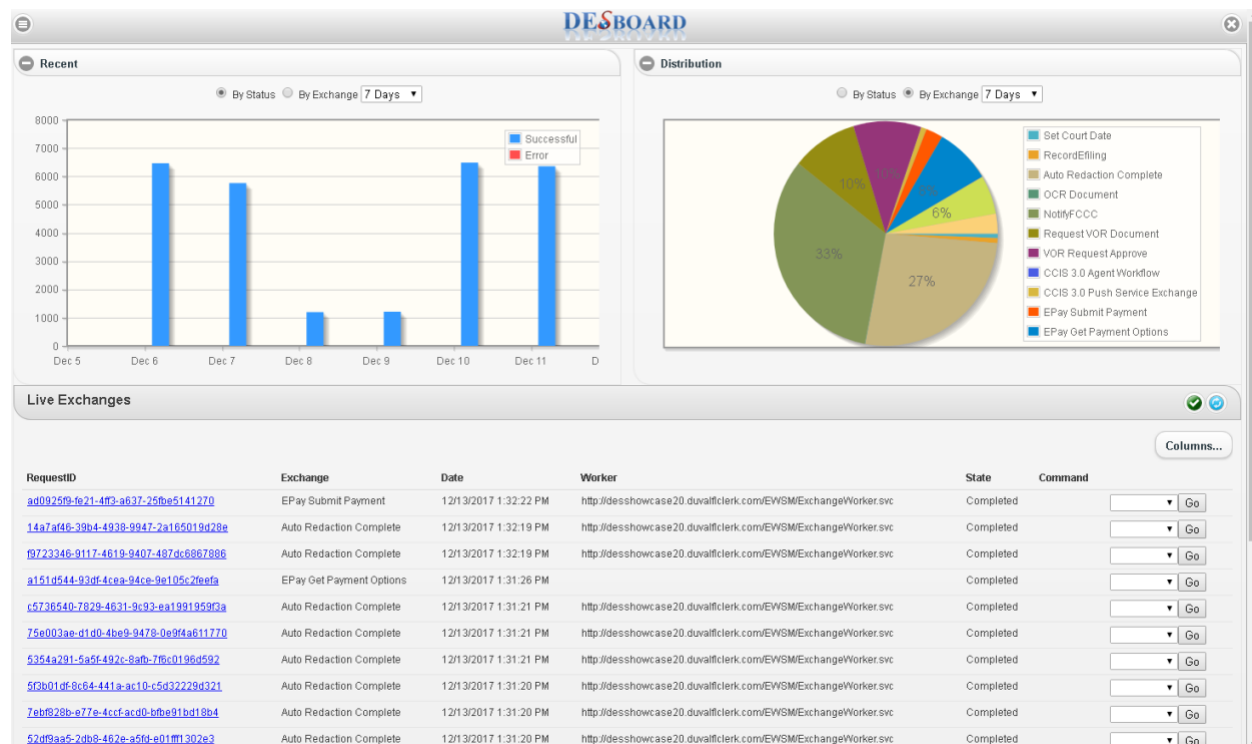
Since every request and its execution details are logged in the database, the dashboard can be used to analyze the activity and execution results of each run of this interface.

DES Dashboard

The dashboard is a web application that allows administrators and developers to manage data exchanges, monitor live activity, and to query and analyze logs.

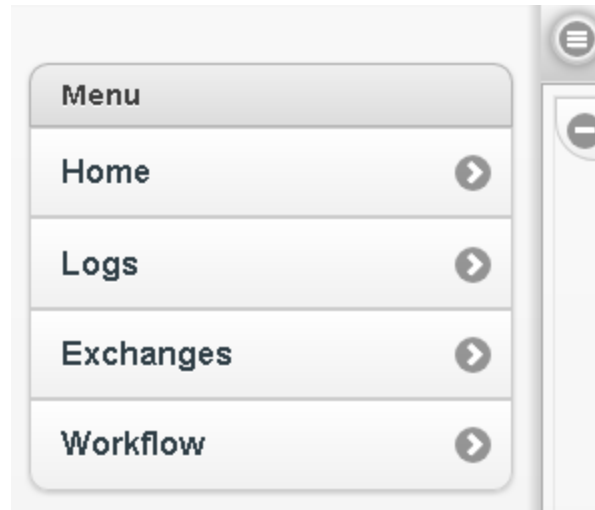
The home page of the dashboard (*see below*) provides interactive graphs that show the requests by exchange and/or by their state. These charts are plotted for a specific time period – Today, past 7, 15 or 30 days. User is able to drill down into a specific section of the pie chart or bar graph to see corresponding exchange requests and their activity.

A “Live Exchanges” view in the home page provides a list of latest exchanges running the environment. From this list user is able to Re-process, cancel or terminate an exchange request. User is also able to click on the Request ID link in this list to view all the event data produced for a request.



DESBoard Home Page

The dashboard provides few other areas to manage the DES environment – Logs, Exchanges and Workflow.



DESBoard Menu

The Logs page allows a user to search for requests based on Exchange Name, Request ID and/or a date range. The grid on this page can be used to view Event Data, Request Details and Response Details of a request. Multiple requests in this view can be selected and set to Re-process.

DESBOARD

Data Exchange Logs

Search: [Search Bar]



Buttons: [Lookup] [Refresh] [Reprocess]


Columns...

<input type="checkbox"/>	RequestID	Name	Request	Response	State	CreateByID	CreateDate	Worker Address
<input type="checkbox"/>	c6d0875e-391a-4fa4-acc1-03e918dcd8ac	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:16 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	8148bead-556c-45c7-80eb-53198b50b15e	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:16 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	f352409b-4c49-45ed-aa88-46b28cb06025	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:16 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	e7fd826e-efa3-4d0d-a520-1877e697ffeb	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:15 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	6aa77e08-e5bf-4764-9231-66f3a34d403	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:15 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	2fbcf074-e975-4a0b-bff0-4f214624bce	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:15 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	522d2da8-bec5-4b78-9ffe-03c8d6314c20	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:15 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	25ec8922-ed04-41b6-8a8f-6e09c5cbf950	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:15 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	1bc80170-5dab-408c-a290-f3a9156283f2	Auto Redaction Complete	Completed	7303	12/13/2017 1:36:14 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	85fc081e-5999-43f9-ba2c-8dae783775dc	NotifyFOCC	InProgress	5930	12/13/2017 1:36:02 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	db45e804-1d28-4e4a-8771-6b2aab32bb64	VOR Request Approve	Completed	506	12/13/2017 1:35:53 PM	http://desshowcase20.duvalitclerk.com/EWSMExchangeWorker.svc
<input type="checkbox"/>	d9a2fa60-9af2-4503-aae3-a285f4ae4cc4	EPay Get Payment Options	Completed	1994	12/13/2017 1:35:40 PM	

DESBoard Logs View

Data Exchanges page allows user to view/edit details of an exchange and also create new exchanges by clicking on “Create New” link on the page.















DataExchanges

[Create New](#)

Columns...

Search:

Name	Description	UnitTemplate	Real Time	ProcessWorkflow	CreateByID	CreateDate	ModifyByID	ModifyDate	Priority	Anonymous	WorkerAddress
  Auto Redaction Complete	Completes auto redaction process	/RedactionComplete	No	Auto Redaction Complete			1253	5/15/2015	3	Yes	
  CCIS 2.0 Agent Workflow	Create and Export CCIS 2.0 Files	/Agent/CCIS_2_Export?ServerUserName=(ServerUserName)&ServerPassword=(ServerPassword)&StartDateOffset=(StartDateOffset)&EndDateOffset=(EndDateOffset)&ApplicationTypes=(ApplicationTypes)&FTPRootDirectory=(FTPRootDirectory)	No	CCIS_2_Workflow	6335	4/22/2015	6335	4/22/2015	3	Yes	
  CCIS 3.0 Agent Workflow	Create and Export CCIS 3.0 Files	/Agent/CCIS_3_Export?ServerUserName=(ServerUserName)&ServerPassword=(ServerPassword)&StartDateOffset=(StartDateOffset)&EndDateOffset=(EndDateOffset)&ApplicationTypes=(ApplicationTypes)&FTPRootDirectory=(FTPRootDirectory)	No	CCIS_3_Workflow	6335	10/18/2016	6335	10/18/2016	3	Yes	
  CCIS 3.0 Push Service Exchange	CCIS 3.0 Push Service Exchange	/Agent/CCIS3PushService?SMTPServer=L_SMTTPServer&SMTPPort=L_SMTTPPort&EmailFrom=L_EmailFrom&EmailTo=L_EmailTo	No	CCIS 3.0 Push Service Workflow	6335	12/27/2016	6335	12/27/2016	3	Yes	
  CCIS Agent Workflow	Create and Export CCIS Files	/Agent/CCISExport?ServerUserName=(ServerUserName)&ServerPassword=(ServerPassword)&StartDateOffset=(StartDateOffset)&EndDateOffset=(EndDateOffset)&ApplicationTypes=(ApplicationTypes)&FTPRootDirectory=(FTPRootDirectory)	No	CCISWorkflow	6335	4/22/2015	6335	4/22/2015	3	Yes	
  E-Record Document	Submits a document for electronic recording	/ERecordDocument	Yes	ERecordDocumentActivity	6335	10/6/2017	6335	10/6/2017	3	Yes	

DESBoard DataExchanges View

Summary

DES is a framework for creation and management of services, data exchanges and interfaces for many distributed applications. Creation and change management of an exchange is a breeze with the use of workflow tools provided by the system. With the dashboard, centralized monitoring of exchange activity will help identify and resolve any issues in a quick and efficient manner.

DES provides ample data for analysis of exchange behavior, performance and load patterns. The architecture of the system enables a simple way to configure and load balance worker services to suit the demands of an organization.