Integration Framework

Reference Guide

Changepoint 2017

Contents

1. Overview

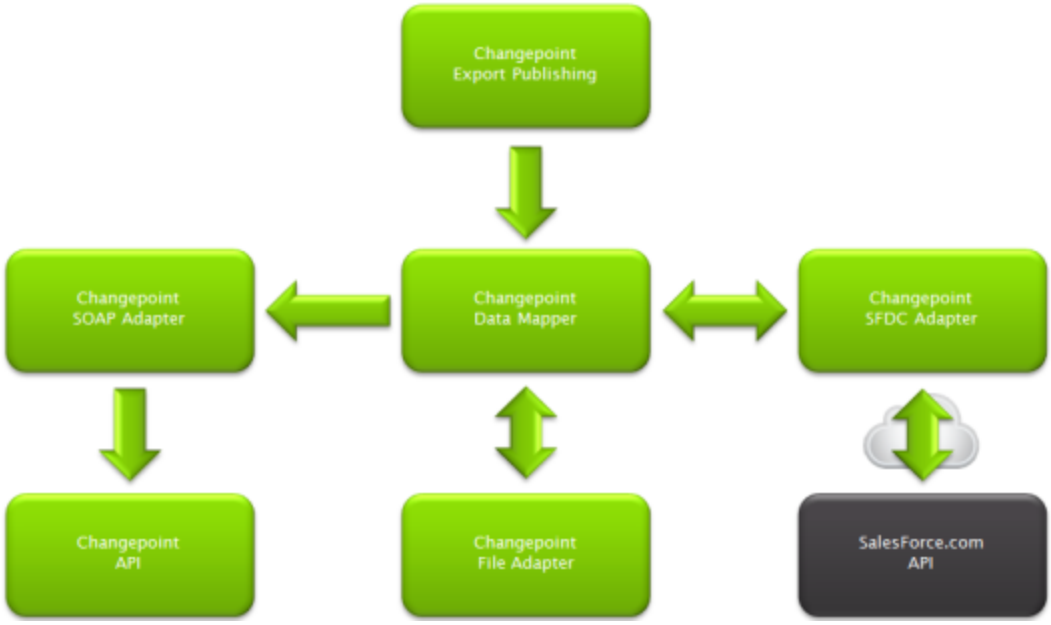
About the Changepoint Integration Framework

The Changepoint Integration Framework allows you to easily integrate data from external systems into Changepoint and to extract information from Changepoint to update external systems.

The Changepoint Integration Framework is a flexible framework that supports multiple ways for exchanging information and provides transformation capabilities through configuration. Depending on the functionality of the external system, minimal programming effort is required to implement an interface with Changepoint.

Changepoint Integration Framework architecture

The Changepoint Integration Framework is composed of multiple components that are loosely coupled using message passing as shown in the following diagram.

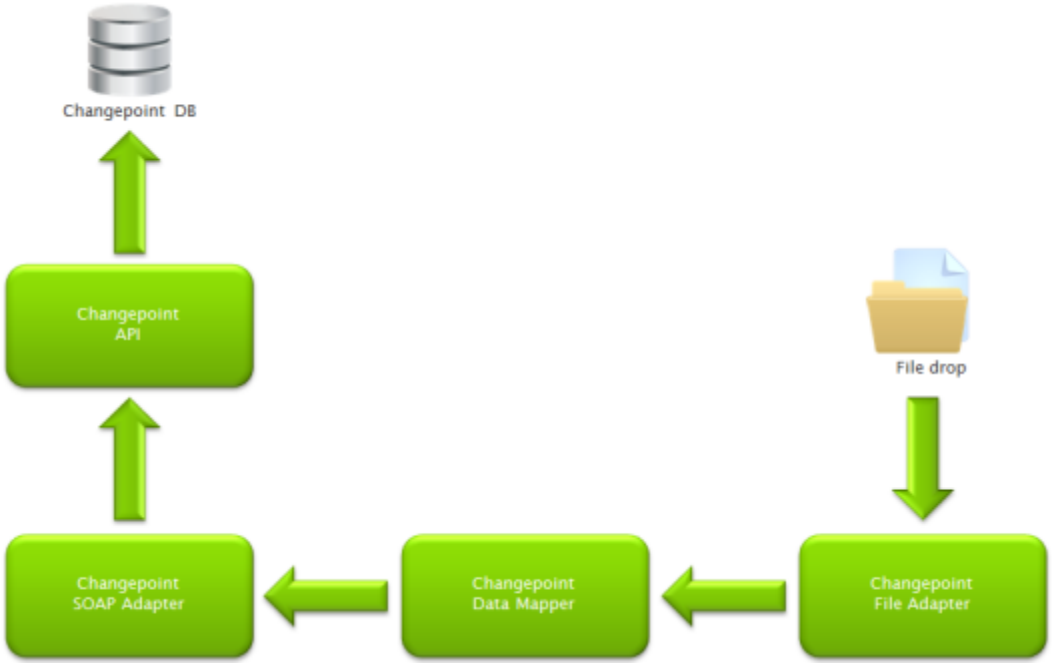


For inbound flow from external systems or outbound flow to external systems, you can use file drop or an adapter.

Inbound flow using file drop

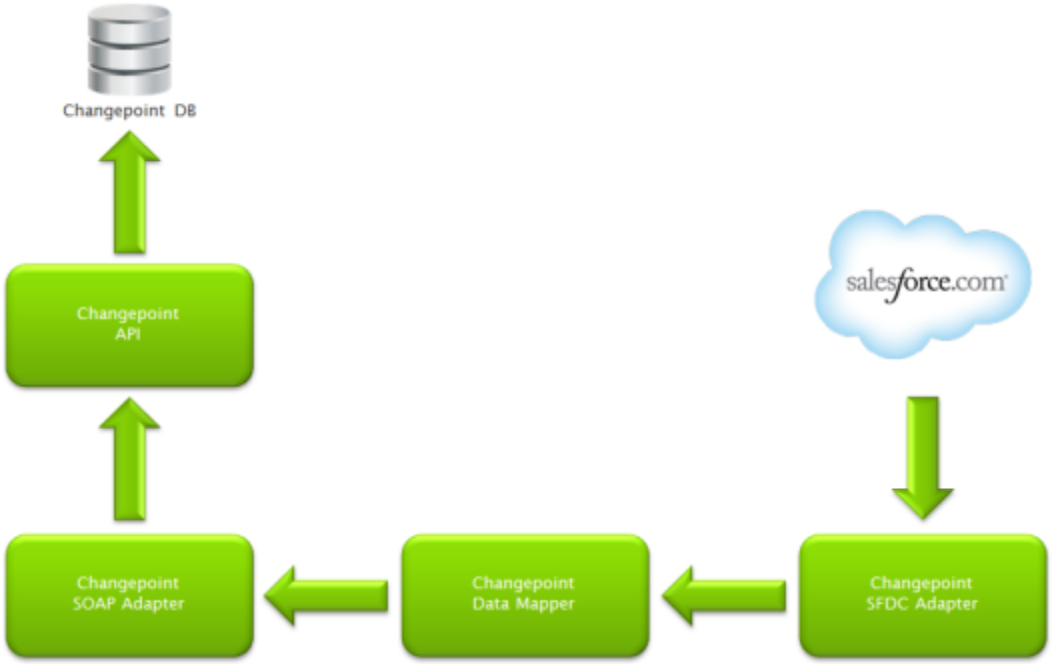
The inbound file drop allows you to integrate data from any external data source into Changepoint by writing a file to a specified folder on a server. Changepoint picks up the files in the file drop folder and processes them. The file can be formatted as a delimited file, a fixed format file, or an XML file.

> **Note:** Only one object type is supported per file. To integrate data for multiple object types, provide a separate file for each type. For example, to create an opportunity and an opportunityservice, provide two input files, one for each object.



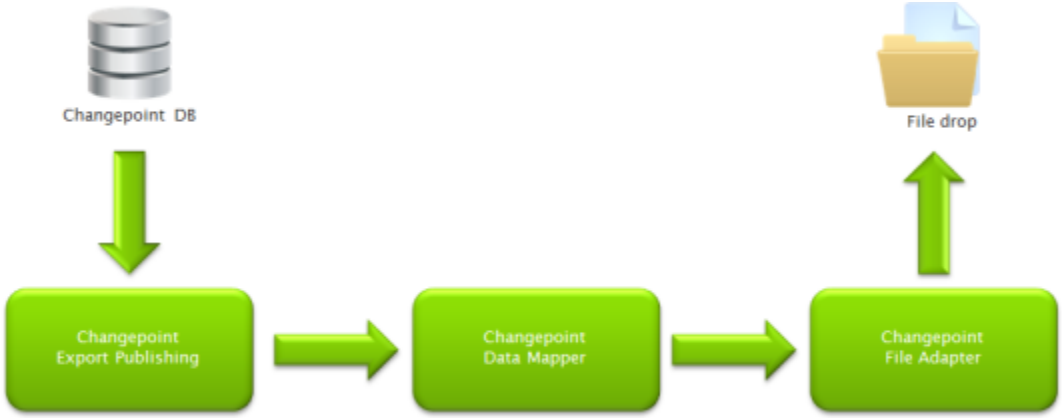
Inbound flow using the SFDC adapter

The inbound SDFC (Salesforce.com) adapter allows you to configure SDFC outbound messages that will be sent to Changepoint for processing. Outbound messages are SOAP transactions that SDFC automatically sends to external systems when triggered.



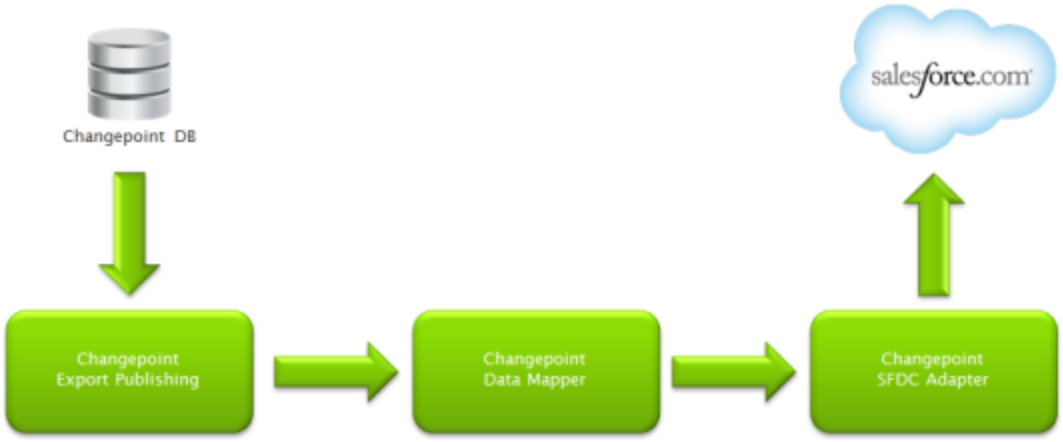
Outbound flow using file drop

The outbound file drop allows you to export data from Changepoint and write it to a file folder on a server. This file can be formatted as a delimited file, a fixed format file or an XML file.



Outbound flow using the SDFC adapter

The outbound SDFC (Salesforce.com) adapter allows you to export data from Changepoint and automatically create or update records in Salesforce.com using the SDFC API.



Microsoft Service Bus

Microsoft Service Bus provides both relayed and brokered messaging capabilities. In the relayed messaging pattern, the relay service supports direct one-way messaging, request/response messaging, and peer-to-peer messaging.

Brokered messaging provides durable, asynchronous messaging components such as Queues, Topics, and Subscriptions, with features that support publish-subscribe and temporal decoupling. Senders and receivers do not have to be online at the same time because the messaging infrastructure reliably stores messages until the receiving party is ready to receive them.

Changepoint DataMapper Service

The Changepoint Data Mapper Service is a Windows Service that is used by the Changepoint Integration Framework, both inbound and outbound, to map fields between Changepoint and an external system. Only one instance of the Changepoint Data Mapper Service is required. Multiple instances of this service are not supported.

The Changepoint Data Mapper Service is used in conjunction with the Changepoint Export Publishing Service, Service Bus, and the Communication Dispatcher Service. You can install Service Bus and the services on the same machine or separate machines as long as they can communicate with each other.

The Changepoint DataMapper Service is where the Changepoint Integration Framework transformation occurs. This service does the following:

- applies locale-aware numeric and date formats
- performs transformation lookups (Booleans, pick lists, etc.)

- processes defaults
- maps the input format to the output format
- routes messages
- provides centralized logging
- tracks the health of the Changepoint Communication Dispatchers

The Changepoint DataMapper service has the following configuration files, which you must set up correctly before activating the service:

- Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config
- ChangepointDataMapper.xml
- transformation files (1 or more)

Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config

This file contains the configuration settings for the logging infrastructure. The Changepoint DataMapper Service uses the Microsoft Enterprise Library to provide logging functionality. The default configuration logs information to the following files:

- Error.log – contains messages that the Changepoint Integration Framework categorizes as warning, error, or critical
- Rolling.log – contains informational messages

ChangepointDataMapper.xml

This file contains the configuration for the communication channels that the Changepoint DataMapper Service listens on for messages that it needs to process and the communication channels that it publishes messages to once it has finished processing them. The configured channels are grouped into the following classes:

- logging – queue that is used to pass logging information from the adapters to the Changepoint DataMapper Service so that the messages can be logged in a central logging file. Do not modify this configuration.
- external – queues that are used for receiving and publishing messages between the Changepoint DataMapper Service and a Changepoint Communication Dispatcher that is used for external systems (for example, file drop or SFDC).

Transformation files

The transformation files contain the configuration information that the Changepoint DataMapper Service uses to determine the mappings and transformations that are required for the messages that are processed. There is one file for each export or import that is processed through the Changepoint DataMapper Service.

Changepoint Communication Dispatcher Service

The Changepoint Communication Dispatcher Service is a Windows service that is used by the Changepoint Integration Framework, both inbound and outbound. It acts as a common communication funnel between the Changepoint DataMapper Service and various adapters. The File Reader Adapter is embedded into the Changepoint Communication Dispatcher Service.

The Changepoint Communication Dispatcher Service has one configuration file called `Adapter.xml` that you must configure before activating the Changepoint Communication Dispatcher Service. In the `Adapter.xml` file, you configure the following:

- external communication channel
- logging communication channel
- File Reader Adapter for processing of inbound files in Changepoint
- File Writer Adapter for outbound flow where a file will be written to a folder on the server
- external assemblies for other adapters

The Changepoint Communications Dispatcher Service handles communication for the Integration Framework by doing the following:

- handles communication between Service Bus and adapters (for example, Changepoint SOAP Adapter or File Reader Adapter for inbound transactions and the File Writer Adapter and SFDC Adapter for outbound transactions)
- implements the File Reader and Writer adapters for inbound and outbound transactions

Changepoint SOAP Adapter

The Changepoint SOAP Adapter is an inbound adapter that receives messages from the Changepoint DataMapper Service by way of the Changepoint Communication Dispatcher Service. The Changepoint SOAP Adapter is used to call the appropriate methods in the

Changepoint API to either create or update information in Changepoint. The adapter reduces the amount of programming knowledge required to work with the Web Services API directly.

Currently the adapter supports the following Changepoint entities:

- Contact
- Customer
- Engagement

  Note: Engagement split billing rules (initiative split charge rules) are not supported.
- Expense
- Opportunity
- Request
- Resource

The file `Changepoint.IntegrationServices.Adapter.Changepoint.dll` that implements this adapter is located in the service installation folder of the related Changepoint Communication Dispatcher Service. The specific configuration information that is required is located in the `external assemblies` section of the `Adapter.xml` file of the related Changepoint Communication Dispatcher Service.

Changepoint SFDC Adapter

The SFDC Adapter is made up of two separate components:

- SFDC Inbound Web Service
- SFDC Outbound Adapter

They are independent and can work alone or together. It is only necessary to configure the component that you require.

SFDC Inbound Web Service component

The SFDC Inbound Web Service component receives messages from the SDFC Outbound Messaging functionality. This web service must be externally addressable (that is, in a DMZ) such that Salesforce.com can call the URL that is configured in the SDFC Outbound Message setup.

The SFDC Inbound Web Service component has an `Adapter.xml` configuration file, which is located in the configuration subfolder of the installation folder.

SFDC Outbound Adapter

The SFDC Outbound Adapter receives messages from the Changepoint DataMapper Service by way of the Changepoint Communication Dispatcher Service. The SFDC Outbound Adapter is used to call the methods in the SDFC API that are required to create or update information in Salesforce.com.

The file `Changepoint.IntegrationServices.Adapter.SFDC.dll` that implements this adapter is located in the service installation folder of the related Changepoint Communication Dispatcher Service. The specific configuration information that is required is located in the `external assemblies` section of the `Adapter.xml` file of the related Changepoint Communication Dispatcher Service.

Currently the SFDC Outbound Adapter supports the following SFDC entities:

- contact
- account
- opportunity

Changepoint Export Publishing Service

The Changepoint Export Publishing Service packages information from Changepoint and publishes it for use by external systems. You must use Changepoint Administration to configure the specific data to be exported. The Changepoint Export Publishing Service has a `CPExportPublishingService.exe.config` file (located in the service installation folder), which you must configure before you activate the service.

When the Changepoint Export Publishing Service starts, it does the following:

- Reads the configuration file.
- Goes to the Changepoint database.
- Checks all of the export publishing setups that have been activated and scheduled.
- Looks at the next run date and compares it to the current date to determine if it is time to run.
- Puts the schedules that are supposed to run into a queue and then executes them.

- Gets the all the data (regular) or the data since the last run (batch).
- Sends the data in a Web Service message to Service Bus and saves the data in XML format if enabled.
- Writes errors to an error table and sends email notifications of errors (if notifications are configured.) Certain system errors that cannot be captured are written to the event log.
- Repeats the process at the interval defined by the RunInterval configuration setting.

You can configure subscriptions to Service Bus, which enable third-party applications to pick up the data exported from Changepoint. The subscription determines how often data is picked up, for example, data could be published daily and picked up weekly.

Changepoint Email Notification Service

The Changepoint Email Notification Service allows you to export information from Changepoint based on certain *events* (business conditions) in Changepoint. You must use Changepoint Administration to configure these conditions. The Changepoint Email Notification Service has a `CPEmailNotification.exe.config` file (located in the service installation folder), which you must configure before you activate the service.

Deployment options for the Changepoint Integration Framework

You can deploy the Changepoint Integration Framework on a single integration server or on multiple servers.

Single integration server deployment

The single integration server deployment makes the most sense when you are integrating Changepoint with internal systems or using file drop. In this deployment, all of the Changepoint Integration Framework components are installed on the same server as shown in the following diagram.

Dual integration server deployment

The dual integration server deployment makes the most sense when you are integrating Changepoint with external systems, such as Salesforce.com. In this deployment, the components of the Changepoint Integration Framework are distributed between two different servers.

2. Installation and Configuration

About installing the Changepoint Integration Framework

This chapter outlines the Changepoint Integration Framework components along with installation and configuration information. For detailed setup steps and examples, see the "Setup Samples" chapter on page 59.

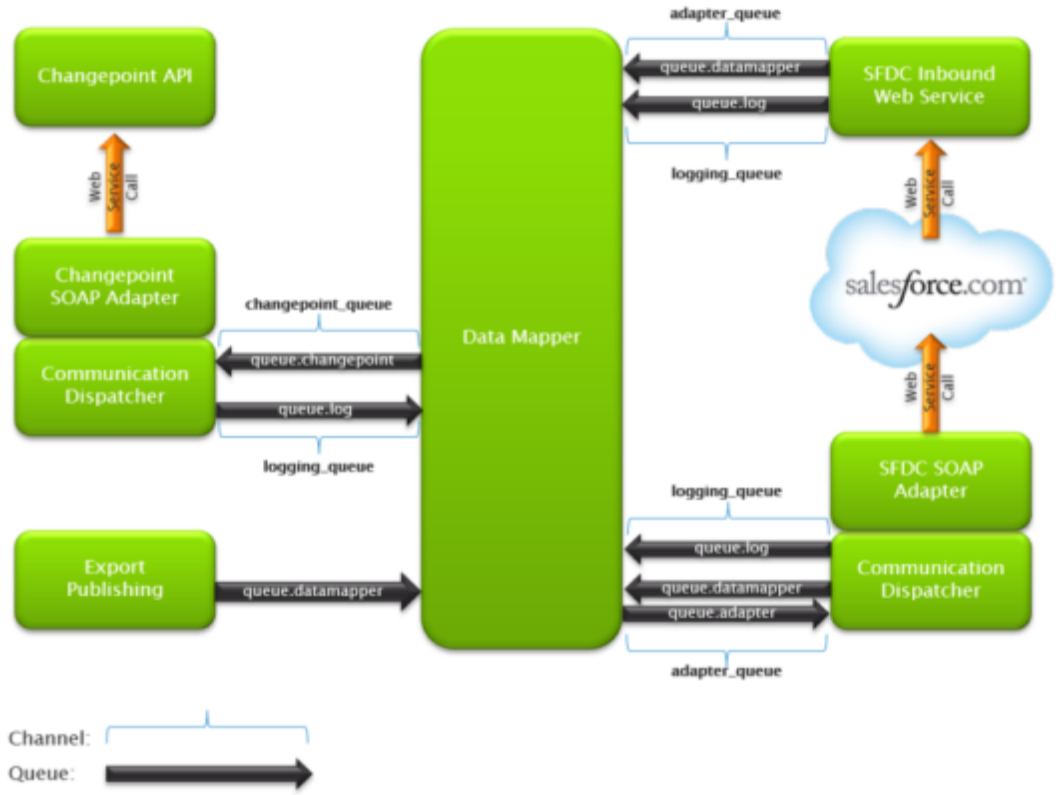
Changepoint documentation

This guide includes references to the following documentation, which is located in the 2017 Release Notes and Patches team folder, and is accessible through Client Portal:

- *Changepoint Installation Guide*
- *Changepoint Product Architecture and Technology Matrix*
- *Changepoint API Installation Guide*
- *Changepoint API Reference*
- *Changepoint Administration Guide*

Message flow

The following diagram shows the relationship of the channels and queues with the various components of the Changepoint Integration Framework. The correct configuration of the channel and queue names is critical for proper message flow. Keep this in mind as you configure the components and refer to the following diagram as necessary.



Templates

Included on the Integration Framework media is a `\Templates` folder that contains XML files that can be used for reference or as a starting point in configuring the various services.

Communication Dispatcher template:

- Adapter.xml – for Communication Dispatcher services

Changepoint Data Mapper templates:

- ChangepointDataMapper.xml – for the Changepoint Data Mapper service
- TransformationTemplate.xml – for configuring transformation files
- amex2cpccexpensefixed.xml – transformation file for American Express corporate credit card integration
- amex2cpccexpensecsv.xml – transformation file for American Express corporate credit card integration

- PassthroughTemplate.xml – for configuring a transformation file that performs no transformation

Changepoint entity templates (contain the entity fields):

- ChangepointContactTemplate.xml
- ChangepointCustomerTemplate.xml
- ChangepointEngagementTemplate.xml
- ChangepointExpenseTemplate.xml
- ChangepointOpportunityTemplate.xml
- ChangepointRequestTemplate.xml
- ChangepointResourceTemplate.xml

Salesforce (SFDC) templates (contain some of the SFDC fields):

- SFDCAccountTemplate.xml
- SFDCContactTemplate.xml
- SFDCOpportunityTemplate.xml

Upgrading the Changepoint Integration Framework

The following Windows services have been updated:

- Changepoint Data Mapper Service
- Changepoint Communication Dispatcher Service
- Changepoint Email Notification Service
- Changepoint Export Publishing Service

For each of the Changepoint Windows services that you use, you must do the following:

1. Create a backup copy of the configuration file for the service.
2. Uninstall the service using Windows Control Panel.
3. Reinstall the service from the Changepoint media.

4. For all configuration files except for the Export Publishing Service file, replace the configuration file in the newly-installed service with the backup copy.
5. For the Export Publishing Service configuration file only, copy the following keys from the backup copy of the configuration file and past them into the new configuration file:
   a. "SQLSettings" key
   b. "Microsoft.ServiceBus.ConnectionString" key
6. After the services are installed, you must enable the WSE API. In the web services Web.config file, uncomment the Microsoft.Web.Services2 element. The default path is:

```
<CP_Root>\API\CP Web Services\Web.config
```

Upgrading a Salesforce.com integration

To upgrade a Salesforce.com integration, do the following:

1. Create a backup copy of the configuration files:

```
ExternalEndPoint\Web.config
ExternalEndPoint\Configuration\Adapter.xml
```

2. On the SFDC Integration Toolkit media, copy the contents of the `Salesforce\ExternalEndPoint` folder, and then paste the contents under the `ExternalEndpoint` folder.
3. Move the backup copies of the configuration files back to the `ExternalEndpoint` folder.

About installing Microsoft Service Bus

This section describes how to install Microsoft Service Bus to be used either for export publishing or for use with the Changepoint Integration Framework.

If you are upgrading, see the "Upgrading the Changepoint Integration Framework" section on page 19.

Service Bus is available in the following modes, all of which are compatible with Changepoint.

- Service Bus 1.1 for Windows Server
- Service Bus for Microsoft Azure (public cloud)
- Service Bus for Windows Azure Pack (private cloud)

Installing Service Bus consists of the following steps:

1. Install Service Bus.
2. Get the Service Bus Connection String.
3. Create a namespace.

Planning your deployment

You can install Service Bus on a separate server, or on the same server as the Changepoint Export Publishing Service. Do not install Service Bus on the same server as the Changepoint database. Service Bus configuration requires the creation of SQL Server Databases. These databases can be deployed on the same server as the Changepoint database.

Messaging Quotas

The following are the messaging quotas specific to the Service Bus for Windows Server that are most relevant to Changepoint:

- Queue/Topic size
- Message size for a queue/topic/ subscription entity
- Message property size for a queue/topic/ subscription entity

Review the complete information for the messaging quotas at:

http://msdn.microsoft.com/en-us/library/ee732538.aspx

Installing Microsoft Service Bus

1. Download the Microsoft Web Platform Installer from:

   http://www.microsoft.com/web/downloads/

2. Start the Web Platform Installer.
3. Under **Products**, click **Windows Azure** and then search for **Windows Azure Pack: Service Bus [version]**.
4. Click **Add**, and then click **Install**.

About configuring Service Bus for Windows Server

Service Bus for Windows Server is a set of components that provide the messaging capabilities of Microsoft Azure Service Bus on Windows. Service Bus for Windows Server enables you to build, test, and run loosely-coupled, message-driven applications in self-managed environments and on developer computers.

This topic explains how to install the Service Bus for on-premise Changepoint deployments. For information about more advanced configurations and topologies, see the Microsoft Service Bus documentation at http://msdn.microsoft.com.

Creating a Service Bus farm

Use the Service Bus configuration wizard to create a new Service Bus farm (cluster of servers), join an existing farm, or leave a farm that you have already joined.

The wizard uses the Service Bus PowerShell cmdlets for all the operations. You can use the wizard to set your farm properties, and then export the generated cmdlet script for future use. However, you cannot use the wizard to modify settings or to perform operations after the farm is created.

1. Click **Start > All Programs > Service Bus 1.1.**
2. Click **Service Bus Configuration**.
3. Click **Using Default Settings (Recommended)**. The **New Farm Configuration** dialog box appears.
4. In the **SQL SERVER INSTANCE** field, enter the complete name of the SQL Server instance that will host the databases for the farm.
5. To verify that the instance name points to a valid instance, click **Test Connection**.
6. In the **Configure Service Account** section, enter the user ID and password for the user account under which services run (for example, Local Admin). The same user credentials are used for all Service Bus services.
7. In the **Certificate Generation Key** section:
    a. Enter a key and reconfirm the key in the second field.
    b. Record the key for future use because you must provide it every time you add a computer to this farm. The configuration cmdlets use this key for generating certificates. You can specify a custom certificate with the custom settings option.

8. If Service Bus clients (your application) will run on the same server as Service Bus, then clear (uncheck) the **Enable firewall rules on this computer** check box.
9. In the **Configure Service Bus Namespace** field, enter `Changepoint`.
10. Leave the **Manage this farm with the Service Bus Management Portal** check box cleared (unchecked).
11. Click next (right arrow).

    The **Summary** dialog box appears.

    Note: For some errors, it may not be possible to continue with the default configuration. For example, if the default port used for Service Bus for Windows Server 1.0 management is blocked by an application, it may not be possible to unblock it. You must then create a new farm with custom settings.
12. To approve the listed options, create the new farm, and add the server to the new farm, click the right arrow.
13. To close the wizard, click the right arrow.

Exporting certificates to remote client machines

To have remote clients connect to a Service Bus for Windows Server management endpoint or Service Bus for Windows Server gateway to configure an auto generated SSL certificate, you must export the auto generated SSL certificate authority (CA) and revocation list (CRL) to the remote clients.

1. On the Service Bus server, open a PowerShell prompt with Service Bus Modules by clicking **Start > Programs > Service Bus [version] > Service Bus PowerShell**.

   Note: If the server has User Account Control enabled, you must open the PowerShell prompt using elevated administrator permissions.
2. Execute the following cmdlet to export the Certificate Authority and Revocation list. By default, the cmdlet exports the Certificate Authority to `AutoGeneratedCA.cer`, and revocation list to `AutoGeneratedCA.crl` files.

```
Get-SBAutoGeneratedCA
```

3. Copy the files to the client machine.
4. On the client machine, open an MMC window:

   a. When prompted, add the Certificates snap-in.

b. Select the **Computer Account** and **Local Computer** options.

c. Right-click the Certificates\Trusted Root Certification Authorities,

d. Open All Tasks, and select Import.

e. Select the `AutoGeneratedCA.cer` file and import it.

f. Click **Next** and then **Finish**.

5. In the MMC window:

a. Right-click on **Intermediate Certification Authorities** and import the CRL files.

b. Select the `AutoGeneratedCA.crl` file and import it.

c. Click **Next** and then **Finish**.

d. If a message appears, asking if you want to replace the current CRL, click **Yes**.

At this point you should be able to trust connections from that particular client. The CER format exports only the public key, not the private key.

Getting the connection string for Service Bus for Windows Server

1. On the Service Bus server, open a PowerShell prompt with Service Bus Modules:

Start > Programs > Service Bus [version] > Service Bus PowerShell

Note: If your server has User Account Control enabled, you must open the PowerShell prompt using elevated administrator permissions.

2. Get the Connection String for a given Namespace:

```
Get-SBAuthorizationRule -Namespace "Changepoint" | format-table connectionstring
-autosize | out-string -width 4096 | out-file C:\Output.txt
```

The connection string is written to `C:\Output.txt`.

Creating a namespace for Service Bus for Windows Server

A namespace provides a scoping container for addressing Service Bus resources within your application.

1. On the Service Bus server, open a PowerShell prompt with Service Bus Modules:

Start > Programs > Service Bus [version] > Service Bus PowerShell

Note: If your server has User Account Control enabled, you must open the PowerShell prompt using elevated administrator permissions.

2. Create a new Service Bus namespace entry in the Service Bus for Windows Server farm:

```
New-SBNamespace -ManageUsers <String[]> -Name <String>
```

where:

- `ManageUsers` – specifies user or group names that will be managers of the service namespace
- `Name` – specifies the name for the new Service Bus for Windows Server service namespace.

Configuring Service Bus for Microsoft Azure (public cloud)

Microsoft Azure Service Bus provides a hosted, secure, and widely available infrastructure for widespread communication, large-scale event distribution, naming, and service publishing.

Tip: Service Bus Explorer is an optional component that you can use to connect to a Service Bus namespace and administer messaging entities. The tool provides advanced features such as import/export and the ability to test messaging entities and relay services. Download the tool and related documentation from the MSDN code gallery.

Creating a namespace for Service Bus for Microsoft Azure

To begin using Service Bus queues in Microsoft Azure, you must first create a namespace. A namespace provides a scoping container for addressing Service Bus resources within your application.

1. Log on to the Azure Management Portal at:

   http://manage.windowsazure.com

2. In the left pane of Management Portal, click **Service Bus**.
3. Click **Create**.
4. In the **Add a new namespace** dialog box, enter the namespace name.
5. Select the country or region in which your namespace is hosted.

   Note: Selecting the same region that you intend to deploy your application in will give you the best performance.

6. Click the check mark.

   The service namespace is created and enabled. It may take several minutes for resources to be provisioned for your account.

   The service namespace appears in the Management Portal.

7. Wait until the status is **Active** before continuing.

Getting the Service Bus Connection String using the Azure Management Portal

1. Log on to the Azure Management Portal.
2. In the left pane of the Management Portal, click **Service Bus**.
3. Select the `Changepoint` namespace.
4. Click **Connection Information**.
5. In the **Access connection information** dialog box, find the ACS Connection String.

Service Bus for Microsoft Azure Pack (private cloud)

Windows Azure Pack for Windows Server enables you to offer rich, self-service, multi-tenant cloud services that are consistent with the public Windows Azure experience. Windows Azure Pack runs on top of the System Center 2012 R2 and is available to Microsoft customers at no additional cost for installation in your data center.

For more information about deploying Microsoft Azure Pack, go to:

http://technet.microsoft.com/en-us/library/dn296432.aspx

Installing and configuring the Changepoint Data Mapper Service

1. You require the Service Bus connection string for the configuration setting `Microsoft.ServiceBus.ConnectionString`. For more information, see:
   - Service Bus for Windows – "Getting the connection string for Service Bus for Windows Server" on page 24
   - Service Bus for Azure – "Getting the Service Bus Connection String using the Azure Management Portal" on page 26
2. Install the Changepoint Data Mapper Service.
3. Configure the following files:

a. Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config

b. ChangepointDataMapper.xml

c. transformation files

Installing the Changepoint Data Mapper Service

Use the following information with the generic procedure for installing and configuring a Changepoint Windows service in the *Changepoint Installation Guide*.

Information Needed	Notes
Installation program name	Changepoint DataMapper Service Installer.exe
Installation path	Default: <cp_root>\Changepoint DataMapper Service\
Configuration file	Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config
Configuration Settings	

Information Needed	Notes

Microsoft.ServiceBus. ConnectionString	The connection string that is used to access Microsoft Service Bus.
	Service Bus for Windows Server:
	`Endpoint=sb://<machinename>/ <servicebusnamespace>;StsEndpoint=https:// <machinename>:9355/ <servicebusnamespace>;RuntimePort=9354; ManagementPort=9355; SharedAccessKeyName=RootManageSharedAccessKey; SharedAccessKey=<sharedaccesskey>`
	Microsoft Azure Service Bus:
	`Endpoint=sb:// <servicebusnamespace>.servicebus.windows.net/ ;SharedSecretIssuer=<issuer>; SharedSecretValue=<secret>`

KeepAliveInterval	Ping-back interval, in minutes, for DataMapper. The value represents the number of minutes between two consecutive ping-backs and must be a positive integer. If set to 0 (zero) then the ping-back mechanism is disabled. Default: 5

Configuring Changepoint.IntegrationServices.WindowsServices. DataMapper.exe.config

The `Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config` file is located in the Changepoint DataMapper Service installation folder.

Search in the file for the `Email Trace Listener` section in the `<listeners>` collection. The service uses this configuration to email errors when they are encountered.

At a minimum, you must update the smtpServer and toAddress attributes.

```
<add name="Email Trace Listener"
   toAddress="to@example.com"
   fromAddress="from@example.com"
   subjectLineStarter="Changepoint DataMapper Service"
   subjectLineEnder=" "
   smtpServer="127.0.0.1"
   smtpPort="25"
```

```
   formatter="Text Formatter"
   traceOutputOptions="None"
   filter="All"
   type="Microsoft.Practices.EnterpriseLibrary.Logging.TraceListeners.
      EmailTraceListener,Microsoft.Practices.EnterpriseLibrary.
      Logging"
   listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.
      Configuration.EmailTraceListenerData,
      Microsoft.Practices.EnterpriseLibrary.Logging" />
```

No additional setup is required for the `Error Rolling Flat File Trace Listener` and the `Rolling Flat File Trace Listener`.

KeepAliveInterval parameter

The KeepAliveInterval parameter controls the interval that the Changepoint DataMapper Service polls the Changepoint Communication Dispatchers to ensure that they are still up and listening. The parameter is in the `<appsettings>` section. The default value is 5 minutes. A value of 0 turns off the polling.

Configuring ChangepointDataMapper.xml

The `ChangepointDataMapper.xml` file is located in the configuration subfolder of the service installation folder. This file describes all of the communication channels that are used by the service. You must configure logging channels and external channels.

Logging channel

The logging channel is a queue that is used to pass logging information from the adapters to the Changepoint DataMapper Service so that the messages can be logged in a central logging file.

Note: Do not modify this configuration.

```
<logging>
      <channel>
      <!-- This channel is used for the Datamapper to receive error/warning
      messages from Adapters -->
      <name>logging_queue</name>
      <subscription>
         <!--Queue to listen on and receive messages -->
         <inboundtype>PointToPoint</inboundtype>
         <inbound>queue.log</inbound>
         <!--Queue to send messages to -->
         <outboundtype>PointToPoint</outboundtype>
         <outbound></outbound>
      </subscription>
```

```
   </channel>
</logging>
```

Note: Do not change any of the other settings or centralized logging will not function correctly.

External channels

The external channels describe the different queues that are used by the Changepoint DataMapper Service to exchange messages with the Changepoint Communication Dispatcher Service that is used for external systems. It is recommended that you use the inbound and outbound queue names as defined in the template file.

The excerpt below shows the channel called `adapter_queue`, which is used to send and receive messages with an instance of the Changepoint Communication Dispatcher Service.

```
<external>
   <channel>
      <!--name needs to match channel name in Adapter.xml -->
      <name>adapter_queue</name>
      <subscription>
         <!--Queue to listen on and receive messages -->
         <inboundtype>PointToPoint</inboundtype>
         <inbound>queue.datamapper</inbound>
         <!--Queue to send messages to -->
         <outboundtype>PointToPoint</outboundtype>
         <outbound>queue.adapter</outbound>
      </subscription>
   </channel>
</external>
```

The following excerpt shows the channel called `export_publishing_customers`, which is used to send messages to the output channel. In this example, messages are sent to the topic `changepoint.customers`.

```
<external>
   <channel>
      <!--name needs to match channel name in Adapter.xml -->
      <name>export_publishing_customers</name>
      <subscription>
         <!--Queue to listen on and receive messages -->
         <inboundtype>PointToPoint</inboundtype>
         <inbound></inbound>
         <!--Queue to send messages to -->
         <outboundtype>PublisherSubscriber</outboundtype>
         <outbound>changepoint.customers</outbound>
```

```
      </subscription>
   </channel>
</external>
```

Note: Do not modify any of the other configuration settings or the messages will not be passed properly.

Configuring the transformation files

The transformation files are located in the `configuration` subfolder of the Changepoint Data Mapper service installation folder. These transformation files describe all of the transformations that are acted upon for every message that is processed by the Changepoint DataMapper Service.

There are four main areas that must be configured in a transformation file:

- direction
- communicationchannels
- messageformat
- fields

Notes regarding transformation files

- transformations are case sensitive:
    - Changepoint field names must be lower case.
    - When configuring Boolean values, ensure that the case is the same in the configuration file and incoming data file.
- When configuring transformation files, refer to the API documentation (Changepoint or SFDC) for the field names to use for each entity. Ensure that you include all the mandatory fields in the inbound file. Mandatory fields include both system mandatory fields and fields that have been configured at your installation as mandatory in metadata.
- When using a new or edited transformation file for file writer or file reader, it is recommended that you first test a small number of records in a test environment.

Defining direction in a transformation file

The direction section in the transformation file must indicate whether the message is an Inbound (import into Changepoint) or an Outbound (export from Changepoint) integration.

Example

```
<direction>Outbound</direction>
```

Defining communication channels in a transformation file

The communicationchannels section in the transformation file must indicate the channel name from the `ChangepointDataMapper.xml` file that the message will be sent to after transformation.

Example 1

```
<communicationchannels>
   <name>adapter_queue</name>
</communicationchannels>
```

Example 2

```
<communicationchannels>
   <name>export_publishing_customers</name>
</communicationchannels>
```

Defining message formats in a transformation file

The messageformat section has two subsections that you must configure:

- changepoint
- external

These two subsections describe generic message formatting information to the Changepoint DataMapper Service. The messageformat configuration is summarized in the following table.

Element	Description
entity	The destination's entity name, for example: <entity>customer</entity> <entity>opportunity</entity> <entity>contact</entity> <entity>engagement</entity> <entity>request</entity> <entity>resource</entity>
type	Type of file being processed. The valid values are: • Delimited • Fixed • XML When the type is Fixed, an additional node called hasheaderrow is required to indicate whether or not a header row is present. For example: <type>Fixed</type> <hasheaderrow>false</hasheaderrow> When the type is Delimited, additional nodes are required to specify the value delimiter, the value separator, and the header row indicator. For example: <type>Delimited</type> <valuedelimiter>"</valuedelimiter> <valueseparator>,</valueseparator> <hasheaderrow>true</hasheaderrow>
culture	Culture of the system. For valid values, see the Microsoft website and search on cultureinfo. Example: <culture>en-US</culture>

Element	Description

booleantransformation	Describes how the system represents Boolean data in the messages.
	Note: This transformation is case sensitive and must match the setting in your inbound or outbound file. For example:
	`<booleantransformation>`
	`<true>true</true>`
	`<false>false</false>`
	`</booleantransformation>`

nullstring	Describes how the system represents a null value in the message. If you define a default value, the default value replaces the null value in the message. For example: `<nullstring>NULL</nullstring>`

Element	Description

datetransformation	Describes how to manage daylight savings time (dst) with respect a defined time zone offset.
	• The offset must be an integer in the range of -14 to 14.
	• If you set the offset to zero, the dst element is ignored.
	• If you do not define a start date and end date for the dst element, the dst element is ignored.
	• The start date and end date format is MM/W/D/HH:mm where:
	MM = two digit month (01..12)
	W = week (1..5)
	D = day of the week. (1..7) where 1 = Sunday, 2 = Monday, etc.
	HH = two digit 24 format hour (00..23)
	mm = two digit minute (00..59)
	Example:
	`<datetransformation>`
	`<offset>-5</offset>`
	`<!-- Eastern Standard Time -->`
	`<dst>`
	`<start>03/2/1/02:00</start>`
	`<end>11/1/1/02:00</end>`
	`</dst>`
	`</datetransformation>`

additionalinfo	Adapter-specific information that is passed to an adapter to describe how to look up entities based on primary key fields (pkfields). There are two formats: one for changepoint and one for external.

Defining entity-based lookups in a transformation file

The following structure in the transformation file is used to define entity-based lookups.

```
<messageformat>
   <changepoint>
      <additionalinfo>
         <pkfields>
            <entity></entity>
            <activity></activity>
            <budget></budget>
            <contact></contact>
            <customer></customer>
            <engagement></engagement>
            <fundingsource></fundingsource>
```

```
            <kmattachment></kmattachment>
            <kmversion></kmversion>
            <opportunity></opportunity>
            <portfolio></portfolio>
            <product></product>
            <project></project>
            <request></request>
            <resource></resource>
            <resourcerequest></resourcerequest>
            <task></task>
         </pkfields>
      </additionalinfo>
   </changepoint>
<messageformat>
```

This structure allows you to pass in a description and look up an entityid based on the passed-in description and a specified configurable field.

The values of the child nodes of the <pkfields> are the names of the configurable fields (also known as *UDFs*). The specified configurable field name must correspond to a configurable text field, and not a configurable code field.

Example 1

In this example, the text2 field in the Changepoint Customer entity will be checked to see if the customer exists in Changepoint based on the passed-in description for the text2 field. If the customer exists, the customer record will be updated.

```
<entity>customer</entity>
...
<messageformat>
   <changepoint>
      <additionalinfo>
         <pkfields>
            <entity>text2</entity>
            ...
         </pkfields>
      </additionalinfo>
   </changepoint>
<messageformat>
```

Example 2

The configurable text field configured in the <resource> child node does have to be in the inbound message, but its value must be mapped to the name node. For example, <accountmanager.name> in a customer message is a resource entity in Changepoint.

To look up the accountmanagerid, set the <resource> child node of <pkfields> to the configurable text field that contains the unique value from the external system. Ensure that the external field that contains this unique value is mapped to <accountmanager.name>.

```
<entity>customer</entity>
...
<messageformat>
   <changepoint>
      <additionalinfo>
         <pkfields>
            ...
            <resource>text1</resource>
            ...
         </pkfields>
      </additionalinfo>
   </changepoint>
<messageformat>
```

Example 3

If the <entity> child node is the same as any of the nodes that follow it, then ensure that you specify the same configurable text field for each. For example, if you want to look up the parent customer on a customer record, then specify the same configurable text field for the <entity> and <customer> child nodes in the customer message.

```
<entity>customer</entity>
...
<messageformat>
   <changepoint>
      <additionalinfo>
         <pkfields>
            <entity>text2</entity>
            <customer>text2</customer>
            ...
          </pkfields>
      </additionalinfo>
   </changepoint>
<messageformat>
```

Defining external lookups in a transformation file

The following structure in the transformation file is used to define external lookups.

```
<messageformat>
   <external>
      <additionalinfo>
         ...
      </additionalinfo>
   </external>
<messageformat>
```

Example

Note: This example is specific to SFDC.

The <pkfield> node describes how to lookup the SFDC entity for the <entity> of the message. In this particular example, the message is a customer message and the SFDC customer will be looked up based on the Customer_Id__c primary key in SFDC.

The <externalids> node describes how to look up additional entities in SFDC.

```
<entity>account</entity>
...
<messageformat>
   <external>
      <additionalinfo>
         <pkfield>Customer_Id__c</pkfield>
         <externalids>
            <account>Customer_Id__c</account>
            <contact>Contact_Id__c</contact>
            <opportunity>Opportunity_Id__c</opportunity>
         </externalids>
      </additionalinfo>
   </external>
<messageformat>
```

Defining field mappings in a transformation file

The fields section in the transformation file is where you configure all of the individual field to field mapping. It consists of a collection of field elements. The fields configuration is summarized in the following table.

For Fixed type data files for inbound file drops, the first column of the fixed width data file is considered to be column 1. You must set the `<start>` location in the transformation file accordingly, as per the following example.

```
<fields>
   <field>
      <datatype>String</datatype>
      <order>0</order>
```

```
      <location>
      <index>0</index>
      <start>1</start>
```

Datatype element

Data type. The valid values are:

- String
- Numeric
- Boolean
- Date

Example:

```
<datatype>String</datatype>
```

Note: Although the field names suggest a Boolean datatype, EngagementWorkcode and EngagementWorklocation "selected" fields are string type, with values 1=selected and 0=not selected.

Order element

The order element defines the order of the fields. The order must start at 0 and must be unique.

Example:

```
<fields>
   <field>
      <datatype>String</datatype>
      <order>0</order>
      ...
   </field>
   <field>
      <datatype>Boolean</datatype>
      <order>1</order>
      ...
   </field>
   ...
</fields>
```

Location element

The location element defines where the Changepoint DataMapper Service can find the field in the input message. The location element includes the following sub-elements:

- index – Index of the field in the input message. Assign index values sequentially starting at zero (0, 1, 2, …). Assign an index value for each field that exists in the input message. Do not assign an index value to a derived field. For more information, see the "Defining derived fields in a transformation files" section on page 41.
- start – Start position of the field, starting at 0. This is only required for the Fixed format type.
- end – End position of the field. This is only required for the Fixed format type.

Example:

```
<location>
   <index>15</index>
   <start></start>
   <end></end>
</location>
```

Mapping element

The mapping element defines the field mapping. Field mappings are case-sensitive. The mapping element has two sub-elements:

- changepoint – Name of the field in Changepoint
- external – Name of the field in the external system

Example:

```
<mapping>
   <changepoint>mainphone</changepoint>
   <external>Phone</external>
</mapping>
```

Tranformation element

The transformation element defines the collection of transformation nodes. Each transformation node has two sub-elements:

- from – value from the input message that you want to replace in the output message
- to – value to be substituted in the output message

Example:

```
<transformation>
   <from>Inactive</from>
```

```
   <to>Old Customer</to>
</transformation>
```

Default value element

The defaultvalue element defines the value that will be substituted in the output message if a NULL is detected in the input message, or if an XML input file is missing a configured node.

This value is also used to configure derived fields. For more information, see the "Defining derived fields in a transformation files" section on page 41.

Example:

```
<defaultvalue>--not available--</defaultvalue>
```

Configuring a passthrough in a transformation file

You can configure a transformation to be a passthrough. When you configure a passthrough, no transformation occurs and the Changepoint DataMapper Service acts as a message router only.

To configure a passthrough, set the value of the passthrough node to `true`. The MessageFormat and Fields collections are not required in a passthrough.

> **Note:** A passthrough sends the data in XML format only. To send the file in a delimited or fixed length format, you must use a transformation file that contains all the fields.

The position of the passthrough node is not important provided that it is a child node of the configuration node.

Example

This sample is the entire contents of the transformation file if you want it passed through.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration type="transformation">
   <passthrough>true</passthrough>
   <direction>Outbound</direction>
   <communicationchannels>
      <name>adapter_queue</name>
   </communicationchannels>
</configuration>
```

Defining derived fields in a transformation files

In the fields section of the transformation file, you can specify derived fields. A derived field is any of the following:

- static string that does not exist in the input message

- field that is derived from the string values of one or more fields in the input message
- combination of static text and the string values of one or more fields in the input message

Use the <defaultvalue> node of the derived field to specify the static text and/or fields to include. Each field that you include in the derived field must be specified in the format |*n*| where *n* is the value of the <index> node of the <field> element that you want to include. The <index> node of the derived field itself must be empty.

Example (Inbound)

In the following example, a derived field has been set up to define a URL that links to a customer page. The URL contains a static portion and a dynamic portion based on the mapped ID.

```
...
<field>
   <datatype>String</datatype>
   <order>3</order>
   <location>
      <index>3</index>
      <start></start>
      <end></end>
   </location>
   <mapping>
      <changepoint>userdefinedcustomerid</changepoint>
      <external>Id</external>
   </mapping>
</field>
...
<field>
   <datatype>String</datatype>
   <order>6</order>
   <location>
      <index></index>
      <start></start>
      <end></end>
   </location>
   <mapping>
      <changepoint>udf.text16</changepoint>
      <external></external>
   </mapping>
   <defaultvalue>http://yourcompany.com/customerpage/|3|</defaultvalue>
</field>
...
```

In the above example, if the mapped value of Id from Salesforce.com is 04kQ0000000DJux, then the URL in udf.text16 in Changepoint appears as follows:

http://yourcompany.com/customerpage/04kQ0000000DJux

Installing and configuring the Changepoint Communication Dispatcher Service

1. You require the connection string for the Service Bus.
2. Install the Changepoint Communication Dispatcher Service.
3. Configure the Changepoint Communication Dispatcher Service.

Installing the Changepoint Communication Dispatcher Service

You can install the service on any server that can communicate with Service Bus. Multiple installations of this service are supported, depending on your configuration.

Use the following information with the generic procedure for installing and configuring a Changepoint Windows service in the *Changepoint Installation Guide*.

Information Needed	Notes
Installation program name	Communication Dispatcher Service Installer.exe
Installation path	Default: <cp_root>\ Changepoint Communication Dispatcher Service\
Configuration file	Changepoint.IntegrationServices.WindowsServices. CommunicationDispatcher.exe.config
Configuration Settings	

Information Needed	Notes

Microsoft.ServiceBus. ConnectionString	The connection string that is used to access Microsoft Service Bus for Windows Server:
	`Endpoint=sb://<machinename>/ <servicebusnamespace>;StsEndpoint=https:// <machinename>:9355/<servicebusnamespace>; RuntimePort=9354;ManagementPort=9355; SharedAccessKeyName=RootManageSharedAccessKey ;SharedAccessKey=<sharedaccesskey>`
	Microsoft Azure Service Bus:
	`Endpoint=sb://<servicebusnamespace>.servicebus.windows.net/ ;SharedSecretIssuer=<issuer>;SharedSecretValue=<secret>`

KeepAliveInterval	Ping-back interval, in minutes, for DataMapper. The value represents the number of minutes between two consecutive ping-backs and must be a positive integer. If set to 0 (zero) then the ping-back mechanism is disabled. Default: 5

Configuring the Changepoint Communication Dispatcher Service

You configure the Changepoint Communication Dispatcher Service in the `Adapter.xml` file, which is located in the configuration subfolder of the service installation folder. Avoid having additional transformation files in the configuration subfolder and avoid using it as a working folder.

The `Adapter.xml` file describes the setup of the Changepoint Communication Dispatcher Service and all of the adapters that it supports.

You must configure the following areas in the `Adapter.xml` file:

- communicationchannels
- filereader
- filewriter
- externalassemblies

Defining communications channels in an Adapter.xml file

The `communicationchannels` section is where you configure the channels that communicate with the Changepoint DataMapper Service for messages and logging. The queue names and

channel names must be exactly the same as the names that are specified in the `ChangepointDataMapper.xml` file.

Example

```
<communicationchannels>
   <external>
      <channel>
         <!--name needs to match channel name in Adapter.xml -->
         <name>adapter_queue</name>
         <subscription>
            <!--Queue to listen on and receive messages -->
            <inboundtype>PointToPoint</inboundtype>
            <inbound>queue.adapter</inbound>
            <!--Queue to send messages to -->
            <outboundtype>PointToPoint</outboundtype>
            <outbound>queue.datamapper</outbound>
         </subscription>
      </channel>
   </external>
<logging>
   <channel>
      <name>logging_queue</name>
      <subscription>
         <!--Queue to listen on and receive messages -->
         <inboundtype>PointToPoint</inboundtype>
         <inbound></inbound>
         <!--Queue to send messages to -->
         <outboundtype>PointToPoint</outboundtype>
         <outbound>queue.log</outbound>
      </subscription>
   </channel>
</logging>
</communicationchannels>
```

Configuring the File Reader adapter in the Adapter.xml file

The `filereader` section of the Adapter.xml file is used to configure the File Reader adapter within the Changepoint Communication Dispatcher Service. Multiple folders can be monitored by configuring multiple <target> nodes.

> **Note:** If File Reader Adapter is not being used, the filereader section must be an empty node.

The configuration elements within each <target> node are:

- datamappingconfiguration – Name of the transformation file that is used by the Changepoint DataMapper Service. The file name must include the .xml suffix.
- path – Path to the folder that the file reader will monitor.
- filenotready – Number and interval of retries. Do not modify these settings.
- enablearchive – when set to 1, the file dropped in the filedrop path is archived. When set to 0, the file is picked up by the Communication Dispatcher Service service, and removed.

  Note: Once <enablearchive> is enabled, the directory that the archived files are moved to will continue to grow. Therefore, you must create a process either for the long term storage or purging of these files.
- archivepath – Path to the archive folder. The archivepath must be blank if enablearchive is 0.

Example

```
<filereader>
   <target>
      <datamappingconfiguration>transformcustomer.xml</datamappingconfiguration>
      <path>c:\inbound\customer</path>
      <filenotready>
         <retry>10</retry>
         <delay>100</delay>
      </filenotready>
      <enablearchive>1</enablearchive>
      <archivepath>c:\filereader\archive</archivepath>
   </target>

   <target>
      <datamappingconfiguration>transformengagement.xml</datamappingconfiguration>
      <path>c:\inbound\engagement</path>
      <filenotready>
         <retry>10</retry>
         <delay>100</delay>
      </filenotready>
      <enablearchive>1</enablearchive>
      <archivepath>c:\filereader\archive</archivepath>
   </target>
</filereader>
```

Configuring the File Writer adapter in the Adapter.xml file

The `filewriter` section of the Adapter.xml file is used to configure the File Writer adapter within the Changepoint Communication Dispatcher Service. Multiple folders can be written to

by configuring multiple <target> nodes.

The configuration elements within each <target> node are:

- datamappingconfiguration – Name of the transformation file that is used by the Changepoint DataMapper Service. The file name must include the .xml suffix.
- path – Path to the folder where the file writer will place the file. If the path does not exist it will be created upon first use.
- filename – Base portion of the file name of the file that will be written. A timestamp will be appended to the file name. The timestamp is of the form YYYYMMDDHHMMSSFFF where

  YYYY = 4-digit year

  MM = 2-digit month

  DD = 2-digit day of the month

  HH = 24-hour clock hour

  MM = minutes

  SS = seconds

  FFF= milliseconds

  Note: If you are using `filewriter`, you must include an entry in the `externalassemblies` node. See the "Configuring the external assemblies in the Adapter.xml file" section on page 47.

Example

```
<filewriter>
   <target>
      <datamappingconfiguration>transform.xml</datamappingconfiguration>
      <path>c:\filewriter\data</path>
      <filename>export</filename>
   </target>
</filewriter>
```

Configuring the external assemblies in the Adapter.xml file

This `externalassemblies` section of the Adapter.xml file is used to configure the external assemblies that implement specific adapter functionality for the Changepoint Communication

Dispatcher Service. There is support for three external assemblies: Changepoint, FileDrop, and SFDC. This is specified in the `type` attribute of the `externalassemblies` element.

Example

```
<externalassemblies>
   <externalassembly type="SFDC">
   ...
   </externalassembly>
   <externalassembly type="Changepoint">
   ...
   </externalassembly>
   <externalassembly type="FileDrop">
   ...
   </externalassembly>
</externalassemblies>
```

Configuring the SFDC external assembly

The SFDC external assembly is used to call the SFDC API on outbound messages from Changepoint.

Each `<configuration>` node is the name of a configured transformation file.

Example

```
<externalassembly type="SFDC">
   <configurations>
      <configuration>cp2sfdc_customer.xml</configuration>
      <configuration>cp2sfdc_opportunity.xml</configuration>
      <configuration>cp2sfdc_contact.xml</configuration>
   </configurations>
   <extendedconfiguration>
   ...
   </extendedconfiguration>
</externalassembly>
```

Extended configuration node

The `extendedconfiguration` node of the SFDC external assembly contains the URL for the SDFC web service, the username and password, and configuration settings for proxies.

Note: Nodes within the `extendedconfiguration` section can only appear once.

Example 1 below uses password encryption (recommended). For more information on how to encrypt the password, see the "Encrypting passwords in the Adapter.xml file" section on page 51.

If you do not want the password to be encrypted, you must remove the <encrypt></encrypt> wrapper elements as shown in Example 2.

Example 1 (encrypted password)

```
<externalassembly type="SFDC">
   <configurations>
   ...
   </configurations>
   <extendedconfiguration>
      <username>sfdc_username</username>
      <encrypt>
         <password>iFH3DWxZidSFwCaAYOCaJcYWa86xBf0wgYqeK8CgxYg=</password>
      </encrypt>
      <bindingtimeout>60000</bindingtimeout>
      <sforceurl>
         https://test.salesforce.com/services/Soap/u/19.0
      </sforceurl>
   </extendedconfiguration>
</externalassembly>
```

Example 2 (plain text password)

```
<externalassembly type="SFDC">
   <configurations>
   ...
   </configurations>
   <extendedconfiguration>
      <username>sfdc_username</username>
      <password>sfdc_password</password>
      <bindingtimeout>60000</bindingtimeout>
      <sforceurl>
         https://test.salesforce.com/services/Soap/u/19.0
      </sforceurl>
   </extendedconfiguration>
</externalassembly>
```

Configuring the Changepoint external assembly

The Changepoint external assembly is used to call the Changepoint API on an inbound message from an external source.

- Each `<configuration>` node is the name of a configured transformation file.
- There is also an `extendedconfiguration` section for additional configuration that is specific to Changepoint. For more information, see the "Extended configuration node" section on page 50.

Example

```
<externalassembly type="Changepoint">
   <configurations>
      <configuration>sfdc2cp_contact.xml</configuration>
      <configuration>sfdc2cp_customer.xml</configuration>
      <configuration>sfdc2cp_opportunity.xml</configuration>
   </configurations>
   <extendedconfiguration>
   ...
   </extendedconfiguration>
</externalassembly>
```

Extended configuration node

The `extendedconfiguration` node of the Changepoint external assembly contains the URL for the CPWebService and the username and password that will be used to login to the Web Service.

Example 1 below uses password encryption (recommended). For more information on how to encrypt the password, see the "Encrypting passwords in the Adapter.xml file" section on page 51.

If you want the password to be plain text, you must remove the <encrypt></encrypt> wrapper element as shown in Example 2.

Example 1 (encrypted password)

```
<externalassembly type="Changepoint">
   <configurations>
   ...
   </configurations>
   <extendedconfiguration>
      <username>cpadmin</username>
      <encrypt>
         <password>LFFTr1tF1PETvmFm2u5V4PjkyNbNdg3yth0xRLEDuTM=</password>
      </encrypt>
      <cpurl>http://localhost/CPWebService</cpurl>
   </extendedconfiguration>
</externalassembly>
```

Example 2 (plain text password)

```
<externalassembly type="Changepoint">
   <configurations>
   ...
   </configurations>
```

```
   <extendedconfiguration>
      <username>cpadmin</username>
      <password>cpadmin</password>
      <cpurl>http://localhost/CPWebService</cpurl>
   </extendedconfiguration>
</externalassembly>
```

Configuring a proxy server in the Adapter.xml file

In some instances, you may need to configure a proxy server in the `Adapter.xml` file. Add the following configuration to the <extendedconfiguration> section after the <password> node:

```
<proxy></proxy>
<domain></domain>
<domainusername></domainusername>
<domainpassword></domainpassword>
```

Encrypting passwords in the Adapter.xml file

1. Install the encryption utility. From the Cognos BI media, copy the contents of the `Utilities/EncryptionUtility` folder to a location on your server.
2. Run:

   ```
   EncryptionUtility.exe
   ```

   The **Encryption Utility** dialog box appears.
3. In the **Source** field, enter the password to be encrypted.
4. Click **Encrypt**.

   The encrypted password is displayed in the **Result** field.
5. Copy the encrypted password into the value of the appropriate <password> node in the Adapter.xml file, as shown in the following example:

   ```
   <encrypt>
      <password>encrypted password</password>
   </encrypt>
   ```

   Note: Depending on your deployment configuration, you may have two passwords to encrypt: one for the Changepoint API login, and one for the SFDC API login.

Installing and configuring the Changepoint Export Publishing Service

The Changepoint Export Publishing service queries the Changepoint database and locates scheduled export publishing that requires processing. The service can be installed on any server that has access to both Service Bus and the SQL server that hosts the Changepoint database.

1. You require the connection string for the Service Bus.
2. Install the Changepoint Export Publishing Service.
3. Configure the database connection settings for the service. For more information, see the *Changepoint Installation Guide*.
4. Set up export publishing in Changepoint Administration. For more information, see the "Setting up export publishing in Changepoint Administration" section on page 54.

Installing the Changepoint Export Publishing Service

Use the information from the following table with the generic procedure for installing and configuring a Changepoint Windows service.

Information Needed	Notes
Installation program name	Export Publishing Service Installer.exe
Installation path	Default: <cp_root>\Changepoint Export Publishing Service\
Configuration file	CPExportPublishingService.exe.config
Configuration Settings	

Information Needed	Notes

Microsoft.ServiceBus. ConnectionString	The connection string that is used to access Microsoft Service Bus.
	Service Bus for Windows Server:
	`Endpoint=sb://<machinename>/ <servicebusnamespace>;StsEndpoint=https:// <machinename>:9355/<servicebusnamespace>;RuntimePort=9354 ;ManagementPort=9355;SharedAccessKeyName= RootManageSharedAccessKey;SharedAccessKey=<sharedaccesskey>`
	Microsoft Azure Service Bus:
	`Endpoint=sb:// <servicebusnamespace>.servicebus.windows.net/ ;SharedSecretIssuer=<issuer>;SharedSecretValue=<secret>`

SQLSettings	Configure the database connection settings for the service.
OutboundQueue	The name of the Service Bus queue that messages are written to. Do not change this value from the default (queue.datamapper),
RunInterval	The interval in minutes between checks to see if there is any export publishing to be run. Default: 5
FilePath	The path that the XML files are written to when EnableWriteXmlFiles=1 Also, the path that the log files are written to when EnableWriteLogFiles=1 Default: `<cp_root>\Changepoint Export Publishing Service\`
EnableWriteXmlFiles	When enabled, XML files that contain export publishing data are written to the path set in FilePath. Default: 0 XML files are normally used only to troubleshoot the data that is being sent to Service Bus. (Export publishing data is sent to Service Bus through Web Service messages regardless of the EnableWriteXmlFiles setting.) For more information, see the "Changepoint Export Publishing Service XML files" section on page 89.
MaxNumberOfRecords InXml	The maximum number of records allowed in an export publishing XML file. Both the Web Service message delivered to Service Bus and the XML file (if EnableWriteXmlFiles=1) are written to disk. When the maximum is reached, a new XML file is created. Default: 500

Information Needed	Notes
EnableWriteLogFiles	When enabled, log files are created for export publishing. Default: 0 The log files record the following: • time that the service started • export publishing schedules that were queued • time that export publishing schedules were processed • time that export publishing schedules finished See also the FilePath parameter.
MaxLogFileSize	The maximum size in KB of the export publishing log file. When the maximum is reached, a new log file is started. Default: 0 (no limit on the log file size)
EnableEmail	When enabled, email notifications are sent for export publishing errors. Default: 0 **Note:** For notifications to be sent, the Changepoint Mail Service must be installed.
EmailAddress	A semicolon-separated list of email addresses to which email notifications of export publishing errors are sent. **Note:** For notifications to be sent, the Changepoint Mail Service must be installed, and EnableEmail must be enabled.
CommandTimeOut	Maximum amount of time in seconds that the service can take to query the database for a call before timing out. Default: 1200 seconds.
MaxExpPubThread	The maximum number of export publishing threads that can run concurrently. Default: 5

Setting up export publishing in Changepoint Administration

For more information about any of the steps in this procedure, see the *Changepoint Administration Guide*.

1. Sign into Changepoint Administration.
2. Import the license file.
3. Assign the **Export Publishing** feature to the appropriate resources and roles.
4. Set up the required export views, data types, definitions, and batch numbering.

Notes:

- Once you create an export publishing schedule, the export definition associated with the schedule can no longer be used in a manual export process. This is true regardless of whether the schedule is active or not.

  To run both manual export and export publishing, set up separate export definitions.

- In export definitions for export publishing, column names can only contain alphanumeric characters.

  Note: This restriction does not apply to manual export.

- In a manual export, you can specify the constant %USER% in the export definition as a placeholder for the name of the user who runs the export.

  However, in export publishing there is no "user". If %USER% is included in an export publishing definition, the value in the export is blank.

- In date format types, use MM for month, for example, dd-MM-yy.

5. Assign access to the **Export Publishing Audit** report and the **Export Publishing Error** report.

Installing and configuring the Changepoint Email Notification Service

For installation and configuration information, see the *Changepoint Installation Guide*.

Starting the Windows services for Changepoint Integration Framework

After you have installed or upgraded the Windows services for Changepoint Integration Framework, they must be started in the following order:

- Service Bus
- Changepoint Data Mapper service
- Changepoint Communication Dispatcher service
- Changepoint Export Publishing service

  Note: For event-based exports, the Changepoint Email Notification service must also be started, but it has no dependencies on the other services.

You can start the services manually, or you can create dependencies between them to ensure that they are started in the correct order. Do not create a dependency for the Changepoint Email Notification service, as it could also be required for general Changepoint notifications.

1. Ensure that the Startup type for each of the services is set to **Automatic (Delayed Start)**.
2. Open a command prompt.
3. Copy and paste the following commands:

```
sc config "Changepoint DataMapper Service" depend= "Service Bus Message Broker"
sc config "Changepoint Communication Dispatcher service" depend= "Changepoint DataMapper service"
sc config "Changepoint Export Publishing" depend= "Changepoint Communication Dispatcher Service"
```

American Express corporate credit card integration

You use the file drop functionality to import American Express (GL 1025) transactions from corporate credit cards into Changepoint.

The credit card transactions are automatically generated for the appropriate resources. The resources can access the credit card transactions in Changepoint and then add the required expense information. They can then add the resulting expenses to expense reports for reimbursement.

To set up an inbound file drop for a corporate credit card feed, see the "S1. Setup for inbound file drop" section on page 61.

Transformation files for corporate credit card feed

You must use the `amex2cpccexpensefixed.xml` and `amex2cpccexpensecsv.xml` transformation files from the `Templates` folder on the Integration Framework media. The transformation files have been preconfigured with settings that are specific to American Express GL 1025 data feeds.

However, you must configure the transformation nodes to map the expense types and currencies to Changepoint values. The transformation files include commenting that explains how to configure the transformation file, and provides examples.

Note: The example configurations map Employee ID in the credit card feed to Resource.UserDefinedId in Changepoint.

Transformation files > message format > external > excluderecordpattern
Transformation files > message format > external > includerecordpattern

The `excluderecordpattern` and `includerecordpattern` nodes are preconfigured to filter the data to be excluded or included in the output.

Transformation files > fields

The `decimalplaceindicator` node in the `fields` section of the transformation file is preconfigured to specify the field that indicates the number of decimal places for the `Numeric` datatype.

Modifying integration framework files

If you modify Adapter.xml, ChangepointDataMapper.xml, transformation files, or data files, you must reprocess the files to put the changes into effect.

1. Open the files in Internet Explorer to verify that the files contain well-formed XML.
2. Stop the Changepoint Communication Dispatcher service and Changepoint Data Mapper service.
3. Restart the Changepoint Data Mapper service first.
4. Restart the Changepoint Communication Dispatcher service.
5. After the services have run, check the `data\received` and `data\inprocess` folders. Files in these folders indicate that an invalid data file, transformation file, or configuration file was used, even if no errors were reported in the `error.log`.
6. Delete the invalid file from the folder, correct the errors, and drop the corrected file into the folder after restarting the services again.

Setting up FileDrop on a language server

To deploy FileDrop on a language server, modify or add a <globalization> tag to the Web Services API web.config as required. The default location of the file is:

```
C:\Program Files\Changepoint\Changepoint\API\CP Web Services\Web.config
```

1. If the `<globalization>` tag already exists in web.config, update both `culture` and `uiCulture` to `"en-US"`
2. If the `<globalization>` tag already does note exist in web.config, add the `<globalization>` tag to the `<system.web>` node, as follows:

```
<system.web>
```

```
<globalization culture="en-US" uiCulture="en-US" />
<!--
Set compilation debug="true" to insert debugging symbols into the
compiled page. Because this affects performance, set this value
to true only during development.
Visual Basic options:
Set strict="true" to disallow all data type conversions where
data loss can occur.
Set explicit="true" to force declaration of all variables.
-->
<compilation debug="true" strict="false" explicit="true">
...
```

3. Restart IIS.

3. Setup Samples

The files referred to in all of the samples are included in `Samples.zip` file in the 2017 Release Notes and Patches team folder.

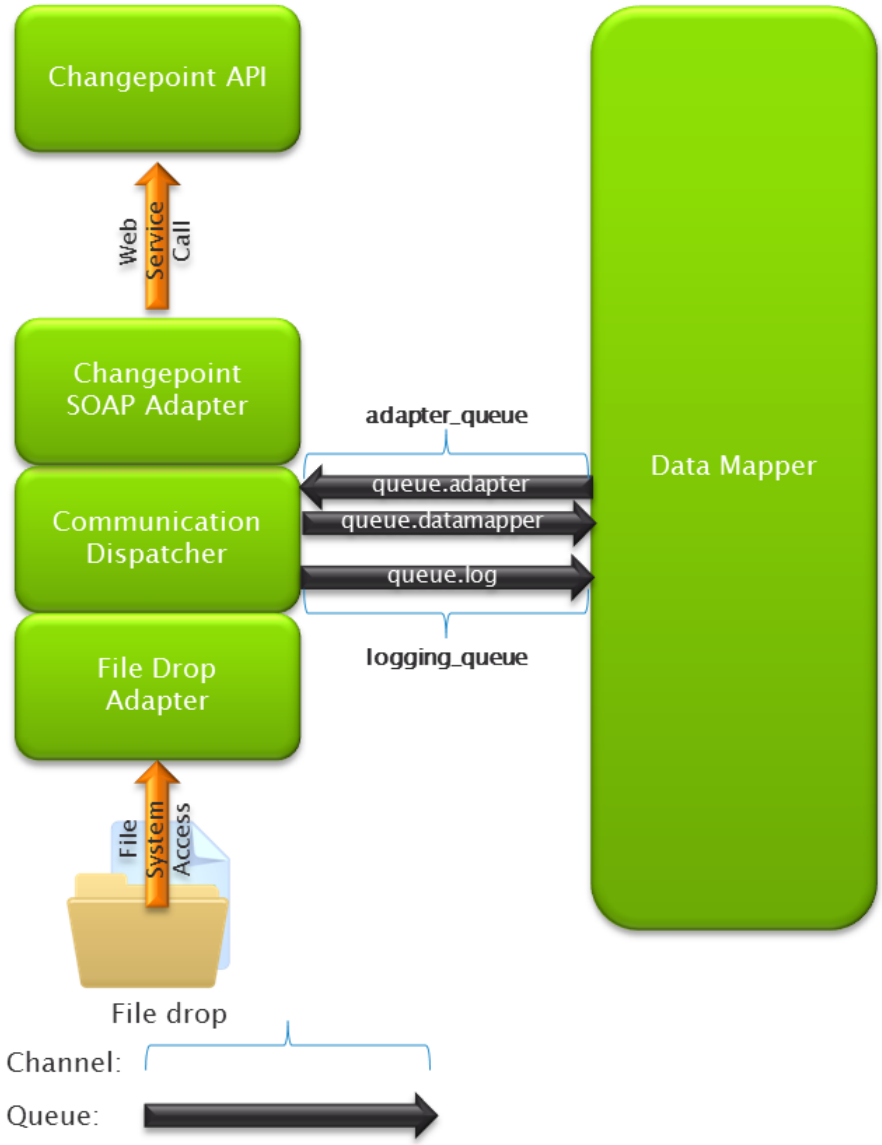
Sample 1. Inbound file drop to synchronize Customer

In this sample we will define the configuration files that could be used to synchronize (update) the Customer entity with a Primary Key field (PKField) from an external system, and persist the field in a Customer configurable text field, which in this case is Text10. Before you use the Text10 configurable field, you must configure it in Changepoint Administration.

It is assumed that the following components are all installed on the same server:

- Service Bus
- Changepoint Data Mapper Service
- Changepoint Communication Dispatcher Service
- Changepoint API

We will configure only one Communication Dispatcher as shown in the following diagram.



The following configuration files are in the directory \Samples\Sample #1:

- Adapter.xml
  - The Changepoint API login and password have been left empty in this file and would need to be populated for this file to be complete.
  - You must either use the encryption utility to encrypt your API password, or remove the “encrypt” tags
  - Archiving of the inbound file is enabled, and the archive directory must be created before using this sample file.
- ChangepointDataMapper.xml

- CustomerSyncTransformationFile.xml

The following sample test data file, which shows the expected input format, are in the directory \Samples\Sample #1 :

- TestData.csv

Notes:

- Some of the configuration and data values might not exist in your environment.
- If Service Bus or the Changepoint API are installed on different servers, then you must update the URLs appropriately.
- The configuration of the filereader requires that the input file be “dropped” into the `c:\inbound\customer` folder for processing.

Included with the sample files are transformation files and test data files for a basic and an advanced Customer import and which include all of the system mandatory fields that will allow Customer records to be created. The advanced transformation file contains examples of how to map various fields, and how to do data transformations.

The following files for the basic Customer import are located in the directory:

`\Samples\Sample #1\Basic Import`

- Adapter.xml
- CustomerBasicImportTransformationFile.xml
- TestData_Customer_Basic_Import.csv

The following files for the advanced Customer import are located in the directory:

`\Samples\Sample #1\Advanced Import`

- Adapter.xml
- CustomerAdvancedImportTransformationFile.xml
- TestData_Customer_Advanced_Import.csv

S1. Setup for inbound file drop

The inbound file drop calls the Changepoint Web Services API for each record created or updated, and therefore very large volumes of data could take considerable time to process. Keep this in mind if you are considering using inbound file drop as an initial data load. If large

volumes of records and fields (for example, 10,000 or more) are included in the inbound files, it could take several hours or even several days for the processing through the API to complete.

Install Service Bus

For more information, see the "About installing Microsoft Service Bus" section on page 20.

Install the Changepoint Web Services API

You must install the Changepoint Web Services API. Create the virtual directory on the IIS server under any website (for example, under the Changepoint website).

Ensure that you set up the virtual directory for anonymous access.

When you use your test connection, use the port number of the website that you put the virtual directory under, for example, `http://localhost:8080/CPWebService/WSLogin.asmx`.

Install the Changepoint Data Mapper Windows service

You must complete the following steps before starting the service.

1. Configure the `ChangepointDataMapper.xml` file.

   Note: Do not change the channel name or queue names. They must be consistent in all of the configuration files.

2. Configure the `Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config` file.

Install the Changepoint Communication Dispatcher service

You must complete the following steps before starting the service.

1. Configure the `Adapter.xml` file.
2. Change the Web Services URL to where the Changepoint API virtual directory is located (for example, the Changepoint URL and port number).
3. Change the Web Services user ID and password to a Changepoint user account that exists in your system (not a SQL account). Ensure that the user account has the security access that is required to perform the same actions in Changepoint, such as creating customers, creating opportunities, etc. You can encrypt the passwords, if required.

4. Under the “filereader” node, configure one “target” node for each type of file drop that you want to perform, and add the config file to the “external assemblies” (Changepoint) node. The following example has three different entities as well as different formats of files, each in its own folder.
5. Specify the same configuration file as in the filereader nodes in the “external assemblies” node further down. The file names are case sensitive.

Configure a transformation file for each inbound transaction

For each inbound node, edit the transformation file to specify the expected data and transformations that are required. For example, one node can point to `File2CP_Customer.xml`. The transformation file must in the `Changepoint DataMapper Service\configuration` folder, and the exact file name (case sensitive) must be referenced in two places in the `Adapter.xml` file.

Some notes about the transformation file:

- Direction is Inbound (case sensitive)
- The format at the top of the file is always XML. The format at the bottom represents the inbound file format (xml, delimited or fixed width).
- The entity at the top represents the type of entity contained in the file (for example, customer, contact, expense). The entity node at the beginning of the “external” node can be “entity” or a specific entity such as “customer”, “contact”, or “expense”.
- The adapter_queue channel must be consistent with the `Adapter.xml` and `ChangepointDataMapper.xml` files and should not be modified.
- For fixed and delimited files, the field order is critical. However, the field name is not critical because it may not even be present in the input file since the header row is optional.
- For xml type files, the field name is critical. Fields can be in any order in the xml file, as long as the field name nodes match what is in the transformation file.

Start the services

Once the configuration is complete, start the services in the following order:

1. Service Bus
2. Changepoint Data Mapper Service

3. Changepoint Communication Dispatcher Service

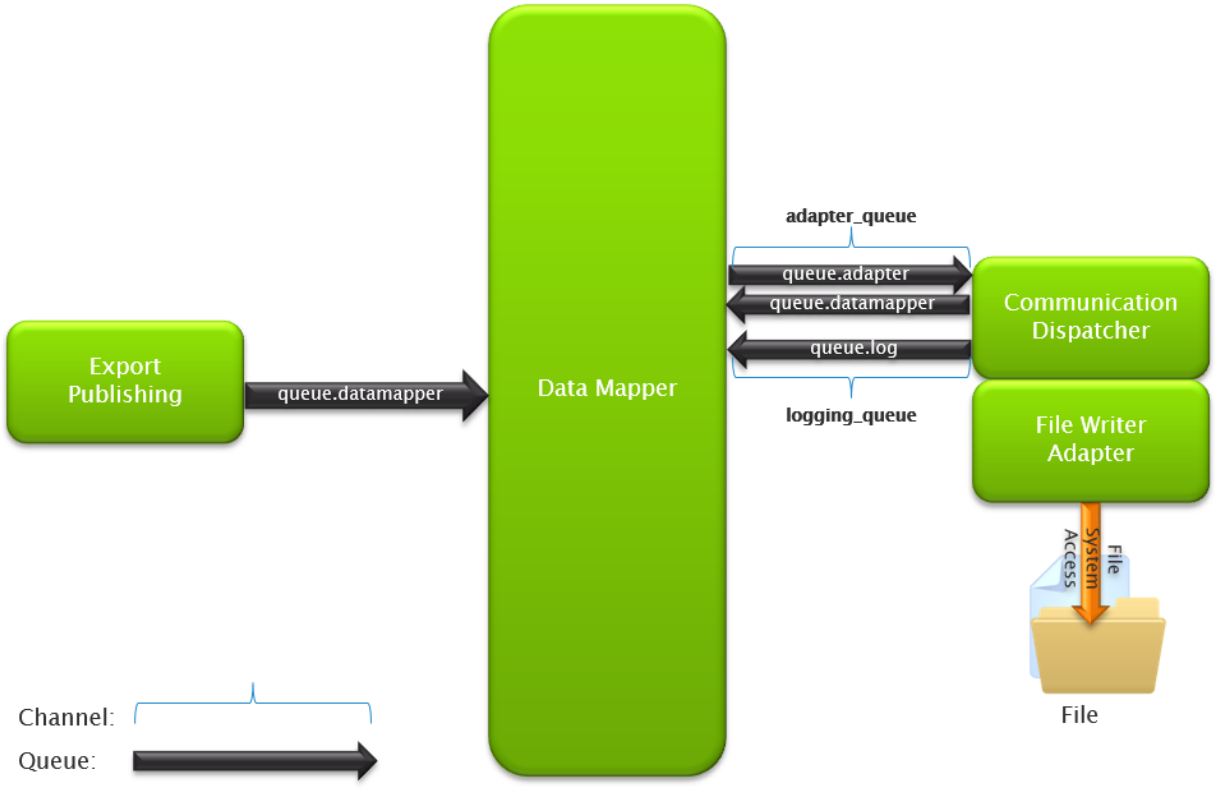
Sample 2. Outbound file drop to export Customer

In this sample, we will define the configuration files that could be used to export the Customer entity to a file. To do this, we will configure only one Communication Dispatcher, as in Sample 1.

For the outbound messaging from Changepoint to function correctly, an Export Publishing view and message must be configured to export the customer information. The Export Publishing message can either be a Batch or Regular message, or an Event-based message, which will generate a message only when a specific criterion is met with the data in the system. In this sample we will show an Event-based message setup.

It is assumed that the following components are all installed on the same server:

- Service Bus
- Changepoint Data Mapper Service
- Changepoint Communication Dispatcher Service



The following configuration files can be found in the directory \Samples\Sample #2:

- Adapter.xml
- ChangepointDataMapper.xml
- CustomerExportTransformationFile.xml

Note: Some of the configuration and data values might not exist in your environment.

S2. Setup for outbound file drop

1. Install the following components:
   - Service Bus
   - Changepoint Data Mapper service
   - Changepoint Communication Dispatcher service
   - Changepoint Export Publishing service
   - Changepoint Email Notification service (only required for event-based exports)

   Note: To use both inbound and outbound file drop, you can combine the information in the two sample files into one `Adapter.xml` file.
2. Start Service Bus.
3. Configure and then start each of the services in the following order:
   a. Changepoint Data Mapper service
   b. Changepoint Communication Dispatcher service
   c. Changepoint Export Publishing service
   d. Changepoint Email Notification service
4. Configure a node in the `Adapter.xml` file in the “file writer” section and in the “File Drop” assembly section.
5. Set up the export view that you want to use and ensure that it contains all the data that you want to export. The existing “ExportExpPub …” views can be used as a starting point, and contain most of the data that would be required.
6. Set up an export definition in Changepoint Administration, selecting the appropriate fields from the view
7. To use event-based exports, set up the event.

Note: Event-based exports are not available for the contact entity and other entities that are not currently supported by the email notification functionality.

8. Set up an export publishing schedule for the export, either a “regular,” “batch” or new “event-driven” export.

   To export contacts or other non-supported entities from Changepoint:

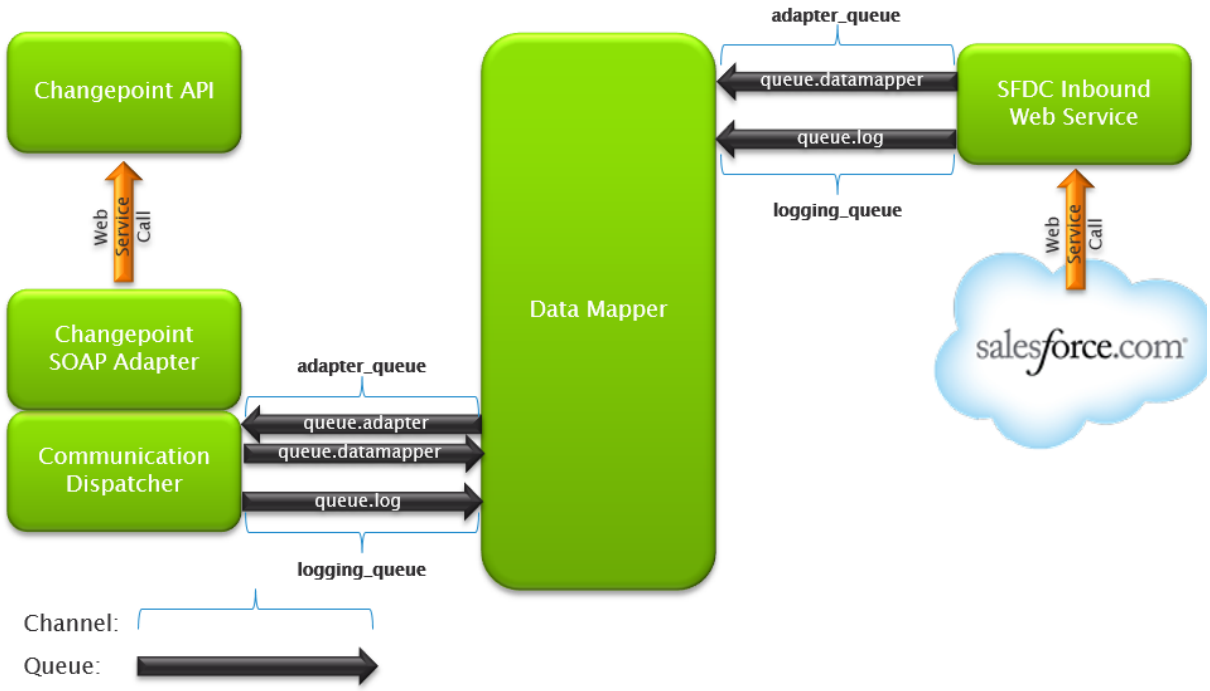
   - Use a “regular” export that filters contacts that were created or updated within a certain time period, and coordinate the export schedule with it.
   - Ensure that the export schedule starts at a future date or time; otherwise, it will not be active when the service starts.

9. Set up the transformation file by specifying each field in the export definition and all necessary transformations. The Changepoint field names must all be lower case.

   Note: If you do not want to transform the data that comes from Changepoint, you can configure a Passthrough Transformation file to preserve the data and field names exactly as they were configured in the export.

Sample 3. Inbound SFDC to Changepoint Customer

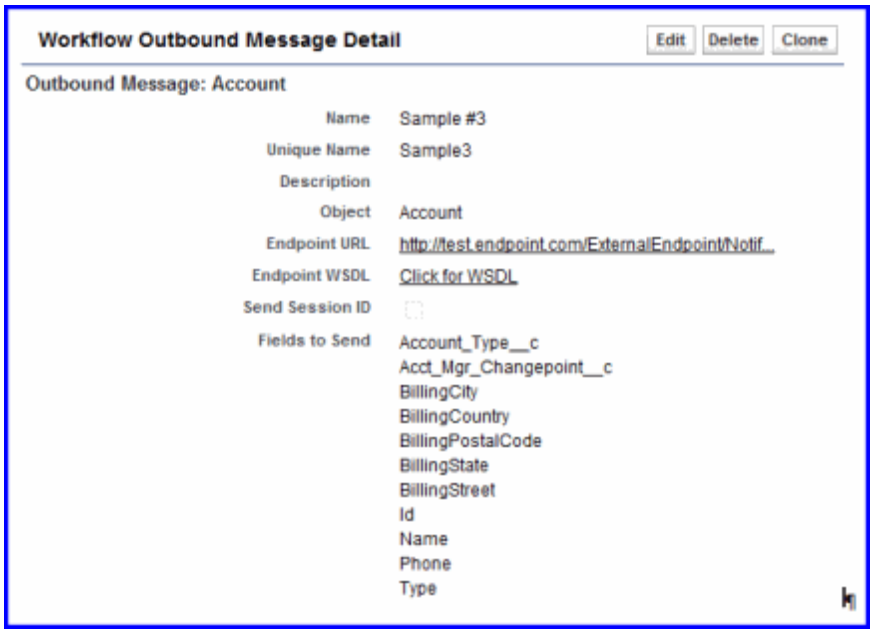
In this sample we will define the configuration files so that a message can be used to transfer SFDC account entities to Changepoint Customer entities. It is assumed that Service Bus, the Changepoint Data Mapper Service and the Changepoint Communication Dispatcher Service and the Changepoint API are all installed on the same server. To do this we will configure one Communication Dispatcher and the SFDC Web Service component as shown in the diagram below. The SFDC Web Service component must be installed on a server that has an Internet address that is accessible from the SFDC servers sending the message.



The following configuration files are in the directory `\Samples\Sample #3`:

- Adapter (CP SOAP Adapter).xml:
  - The Changepoint API login and password are blank (empty) in this file and must be populated for this file to be complete.
  - You must either use the encryption utility to encrypt your API password, or remove the "encrypt" tags
- Adapter (SFDC Web Service).xml
- ChangepointDataMapper.xml
- SFDC2CPCustomerTransformationFile.xml

You must set up an SFDC outbound message with the following format.



Where the Endpoint URL is:

http://test.endpoint.com/ExternalEndpoint/NotificationService.asmx?ConfigId=SFDC2CPCustomerTransformationFile.xml

Note: The parameter ConfigId on the URL is the name of the transformation file, and the FQDN of text.endpoint.com is the specific endpoint deployed on the customer premises.

S3. Setup for inbound SFDC

For both outbound and inbound integrations with SDFC (Salesforce.com), an initial mapping exercise and setup should be performed.

Note: All of the sample files in this procedure are from Sample #3.

1. Set up a configurable text field for each entity that will be used as the PK field (and will be populated with the SFDC record ID), to ensure the appropriate records are updated. This includes customer, contact, opportunity and resource. Resource is used as a lookup when creating or updating Changepoint records, for example, an account manager or sales representative, and should match the SFDC ID. The user-defined ID for customer, contact, and resource can also be used, if they are not already in use.
2. Perform an initial data mapping of existing records between Changepoint and Salesforce.com, to populate the configurable text fields with the corresponding SFDC ID. This can be done manually or through a one-time push of initial data from Salesforce.com.

Note: If names are not the same in the two systems (for example, a sales representative is "John Smith" in SFDC, and "Smith, John" in Changepoint), the PK field in Changepoint might have to be updated manually.

3. Decide which records and fields are going to be created or updated in both directions. Ensure that you include all mandatory fields in Changepoint (for inbound) and SFDC (for outbound).
4. Start Service Bus. Service Bus can be installed inside the firewall, but ports must be opened to allow communication from the Internet to the Service Bus instance.
5. Configure the ChangepointDataMapper.xml file. Use the sample file, ChangepointDataMapper.xml.
6. Configure the transformation file. Use the sample file: SFDC2CPCustomerTransformationFile.xml.

   Refer to examples of different options in the mapping file, such as using PK fields, transformations, using configurable fields, etc.
7. Configure the Adapter.xml file for the Changepoint Communication Dispatcher service. Use the sample file: Adapter (CP SOAP Adapter).xml, but rename the file to Adapter.xml.
8. Start the services in the following order:
   a. Changepoint DataMapper service.
   b. Changepoint Communication Dispatcher service.
9. Copy the contents of the following folder from the Changepoint Salesforce.com Integration Toolkit product media to a server outside the firewall:

```
Salesforce\Adapter\SOAP\ExternalEndPoint
```

10. In Internet Information Services (IIS) on the outside server:
    a. Create a virtual directory under the default website called: SFDC Endpoint.
    b. Point the virtual directory to the folder:

```
Salesforce\Adapter\SOAP\ExternalEndPoint
```

    c. Set the DefaultAppPool application pool to use .NET 4.0.
    d. Modify the Web.config configuration setting "Microsoft.ServiceBus.ConnectionString". For the value, see:

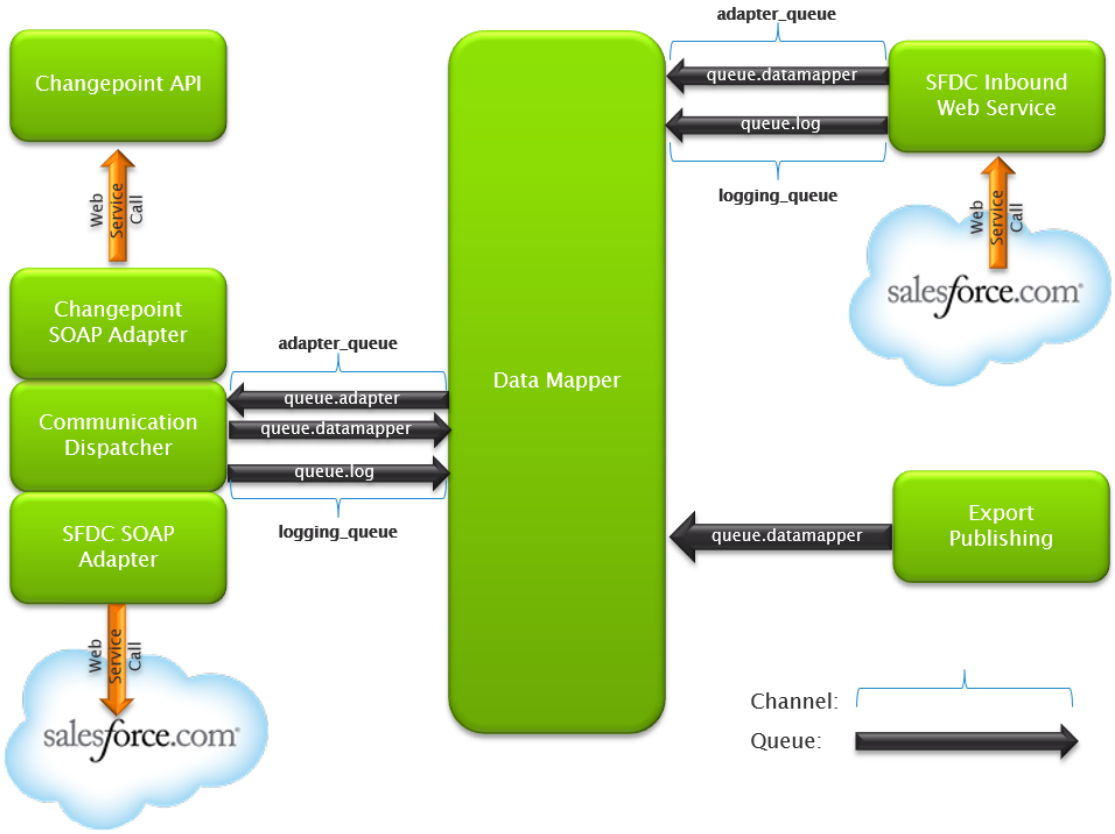
- Service Bus for Windows – "Getting the connection string for Service Bus for Windows Server" on page 24
- Service Bus for Azure – "Getting the Service Bus Connection String using the Azure Management Portal" on page 26

11. Create a Workflow Rule, where the condition to send the outbound message is set up. For example: when AccountType='Customer' and Name starts with 'Changepoint'.
12. Create an outbound message in SFDC for each entity (customer, contact, opportunity) that you want to push into Changepoint, including the fields to be sent to Changepoint and conditions under which the message will be generated. See the screen clip in the "Sample 3. Inbound SFDC to Changepoint Customer" section on page 66. The message must call the Web Services URL and reference the transformation file (one transformation file per entity, case sensitive). Ensure you include the ID of each entity in the message, and map it to the Changepoint text field or the user-defined ID that you configured as the primary key field.

Sample 4. Bidirectional SFDC-Changepoint Customer

In this sample, we will define the configuration files so that Changepoint customers can be created and updated from SFDC account entities, and Changepoint customers can update SFDC account entities. It is assumed that Service Bus, Changepoint Data Mapper Service, Changepoint Communication Dispatcher Service, and the Changepoint API are all installed on the same server.

We will configure one Communication Dispatcher and the SFDC Web Service component as shown in the diagram, below. The SFDC Web Service component must be installed on a server that has an Internet address that is accessible from the SFDC servers sending the message. It is assumed that the Communication Dispatcher is installed on a server that can access the Internet to make web service calls to Salesforce.com



The following configuration files can be found in the directory \Samples\Sample #4:

- Adapter (CP & SFDC SOAP Adapter).xml
  - The correct Changepoint API login and password must be populated for this file to be complete.
  - The correct SFDC API login and password must be populated for this file to be complete.
  - The SFDC sforceurl parameter must be updated to the correct production URL for access to the API.
- Adapter (SFDC Web Service).xml (same as in Sample #3)
- ChangepointDataMapper.xml
- CP2SFDCCustomerTransformationFile.xml
  - This file will allow the update of the customer name back to SFDC if it is changed in Changepoint.
- SFDC2CPCustomerTransformationFile.xml (same as in Sample #3)

For the inbound messaging from SFDC to function correctly, an SFDC workflow rule and outbound message must be set up, similar to Sample #3.

For the outbound messaging from Changepoint to function correctly, an Export Publishing message must be configured to export the customer information. The Export Publishing message can either be a Batch or Regular message, or an Event-based message, which will generate a message only when a specific criterion is met with the data in the system.

For more information, see export publishing in the Changepoint Administration Guide.

Note: The **Configuration ID** field must contain the name (case sensitive) of the outbound datamapper file, which in this example is CP2SFDCCustomerTransformationFile.xml. The topic selected must correspond to the changepoint.cp.customers topic that is configured in the ChangepointDataMapper.xml file.

S4. Setup for bidirectional SFDC-Changepoint

1. Install the inbound SFDC Integration Framework components on a server inside the firewall. For more information, see the "S1. Setup for inbound file drop" section on page 61.

   Service Bus can be installed inside the firewall, but ports must be opened to allow communication from the Internet to the Service Bus instance.

2. Start Service Bus.

   Note: Do not start the services until all the configuration files are complete and in the proper folders.

3. Configure the Adapter.xml file under the Changepoint Communication Dispatcher service. Use the example from the samples provided (sample 4: file name Adapter (CP SOAP Adapter).xml. Rename the file to adapter.xml, and set up both the inbound and outbound sections in the file.

   Note: A token might need to be generated from the Salesforce.com API. It is also possible to go through a proxy, if required.

4. Configure the ChangepointDataMapper.xml file. Use the example from the samples provided (file name ChangepointDataMapper.xml).

5. Configure a channel and topic for each outbound message. For more information, see “Adding a new topic for export publishing” in the Changepoint Administration Guide.

   Note: The “topic” in the export definition must match the topic in the channel

6. Configure the outbound and inbound transformation files

   The outbound file is very similar to the inbound transformation file, except for the direction. Also, the “to” and “from” would be reversed in the data transformations (“from” being the Changepoint value and “to” being the SFDC value). The entity in the “external” node would be “account,” which is the SFDC name for Customer.

   Ensure the Changepoint field names are all lowercase for outbound transactions

7. Start the services in the following order:

   - Changepoint Data Mapper
   - Changepoint Communication Dispatcher

8. Install the Export Publishing service.

9. If you are using event-based reports, install the Changepoint Email Notification service.

   Note: Event-based exports are not available for the contact entity.

10. Set up the export view that you want to use, ensuring it contains all the data you wish to update to Salesforce.com. The existing “ExportExpPub …” views can be used as a starting point, and contain most of the data that would be required.

11. Set up an export definition in Changepoint Administration, selecting the appropriate fields from the view.

12. If you are using event-based reports, set up the event.

13. Set up an export publishing schedule for the export, either a “regular,” “batch” or new “event-driven” export.

    To create or update contacts in SFDC from Changepoint, use a “regular” export that filters on contacts that were created or updated within a certain time period, and coordinate the export schedule with it.

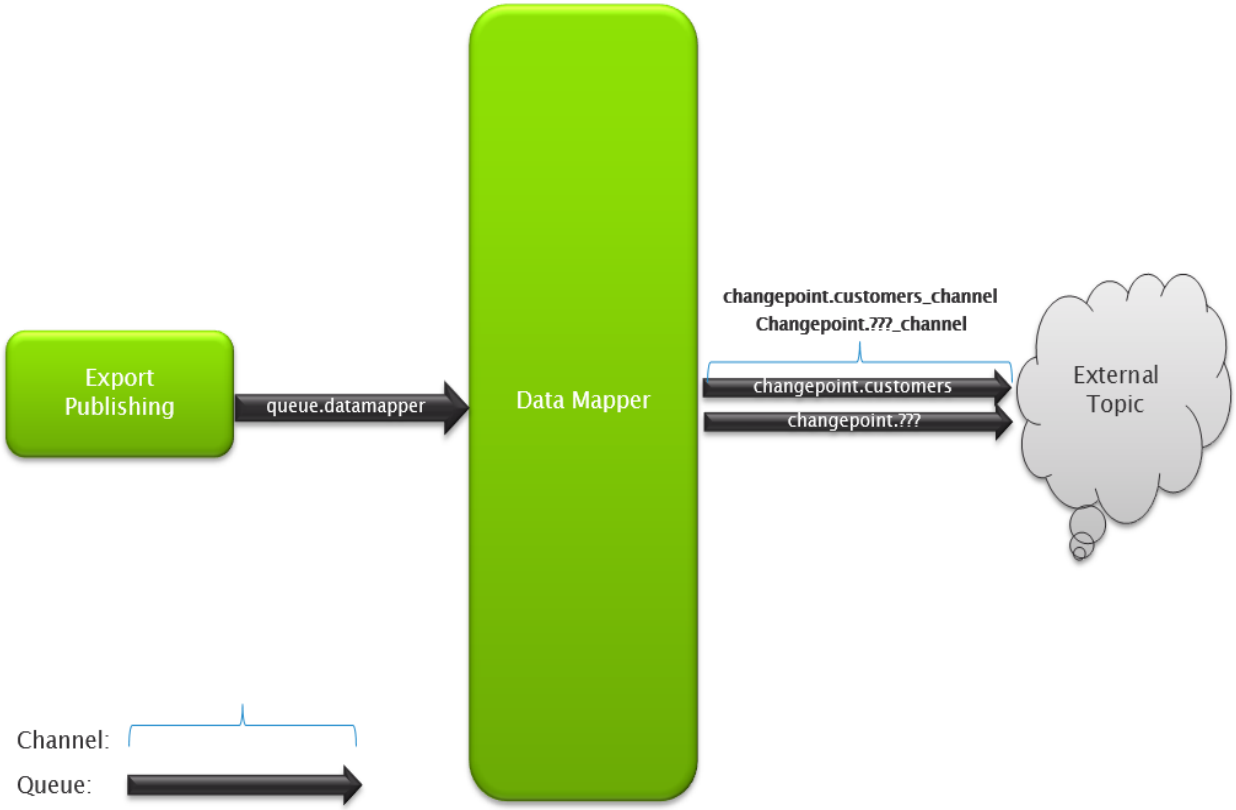
    Ensure the schedule starts at a future date and time, otherwise it will not be active when the service starts.

Sample 5. Outbound export Customer to external topic

In this sample, we will define the configuration files that could be used to export the Customer entity to an external topic.

It is assumed that the following are all installed on the same server:

- Service Bus
- Changepoint Data Mapper Service



The following configuration files can be found in the directory \Samples\Sample #5:

- ChangepointDataMapper.xml
- ExportAllCustomersToTopic.xml

Note: Some of the configuration and data values might not exist in your environment.

For the outbound messaging from Changepointto function correctly, an Export Publishing message must be configured to export the customer information. The Export Publishing message can either be a Batch or Regular message, or an Event-based message, which will generate a message only when a specific criterion is met with the data in the system. In this sample we will show a Regular message setup.

First, you have to define the export view, which contains the SFDC ID (a text field or the user-defined ID that is mapped to the SFDC record ID) and the other information you want to update.

Next you have to define the export, which is done in the Export Publishing Setup view in Changepoint Administration.

Note: The Configuration ID field must contain the name (case sensitive) of the outbound datamapper file, which in this example is ExportAllCustomersToTopic.xml. The topic selected must correspond to the changepoint.customers topic that is configured in the ChangepointDataMapper.xml file.

S5. Setup for outbound export to external topic

1. Install the following components:
   - Service Bus
   - Changepoint Data Mapper service
   - Export Publishing service
   - Email Notification service (only required for event-based exports)

   Note: To use both inbound and outbound file drop, you can combine the information in the two sample files into one Adapter.xml file.
2. Start Service Bus.
3. Configure and then start each of the services in the following order:
   a. Changepoint Data Mapper service
   b. Export Publishing service
   c. Email Notification service
4. Set up the export view that you want to use and ensure that it contains all the data that you want to export. The existing "ExportExpPub …" views can be used as a starting point, and contain most of the data that would be required.
5. Set up an export definition in Changepoint Administration, selecting the appropriate fields from the view
6. To use event-based exports, set up the event.

   Note: Event-based exports are not available for the contact and other entities that are not currently supported by the email notification functionality.

7. Set up an export publishing schedule for the export, either a "regular," "batch" or new "event-driven" export. To export contacts or other non-supported entities from Changepoint, use a “regular” export that filters contacts that were created or updated within a certain time period, and coordinate the export schedule with it. Ensure the schedule starts at a future date or time; otherwise, it will not be active when the service starts.
8. Set up the transformation file, specifying each field in the export definition and any transformations necessary. The Changepoint field names must all be lower case.

   Note: The export publishing functionality can still be used to generate messages, but have to go through the data mapper, instead of directly to Service Bus. A pass-through configuration file can be set up to send the data without any transformations (sample is provided).

Primary key lookup fields

When designing the integration of data from one system to another, you must identify the primary key lookup fields that are used for the different entities in the source and target systems. If you plan to update the data that is moved from one system to another at a later time, you must ensure that the primary key from the source system is transferred to the target system and is available to be used as a lookup. This is what the PKFields node in the transformation files is used for.

For example, if you are bringing customer records into Changepoint, the primary key field from the external source system must be mapped to either the userdefinedid field or a configurable text field. If a configurable text field is used, then you must configure the configurable text field in the PKFields node.

When you are initially implementing integration between Changepoint and an external system, both systems will have data entities that represent the same entity (Customer, Contact, etc.). You should first synchronize the entities by updating the target system with the PKField from the source system.

The most efficient way to do this is to:

1. Generate an extract file with the entity name and its PK from the source system.
2. Import the extract file into the target system, matching by entity name and updating a field in the target system to hold the PK field from the source system.

3. At this point, steady state integration can be set up to use the PK field from the source system to do lookups in the target system. For more information, see "Sample 1. Inbound file drop to synchronize Customer" on page 59.

   Note: The resource entity is used as a lookup for some entity fields. Therefore, you may have to synchronize the resource entity from the external system.

How Changepoint looks up information

When the Changepoint API is identifying whether a record being passed from the Changepoint SOAP Adapter is a new or existing record it follows a consistent order as follows:

Contact API

1. SQL GUID of the record in the Changepoint database
2. PKField
3. Email1
4. Name

Customer API

1. SQL GUID of the record in the Changepoint database
2. PKField
3. UserdefinedID
4. Name

Opportunity API

1. SQL GUID of the record in the Changepoint database
2. PKField
3. Name

Request API

1. SQL GUID of the record in the Changepoint database
2. PKField
3. Request Number

Resource API

1. SQL GUID of the record in the Changepoint database
2. PKField
3. UserdefinedID
4. First name and last name

Engagement API

1. SQL GUID of the record in the Changepoint database
2. PKField
3. UserdefinedID
4. Name

Engagement API – lookup logic for UpdateByXML

The following sequence is used to find the engagement ID:

1. If the sEngagementId parameter is passed in, the method uses this value for the engagement ID.
2. If this fails, the method attempts to extract the engagement ID from <engagement id> in the XML.
3. If this fails, the engagement ID is taken from the object properties.
4. If this fails, an attempt is made to look up the engagement ID using <userdefinedengagementid> in the XML. If <userdefinedengagementid> has a value, but the value is invalid or duplicated, then you will get an error and no further attempts are made to look up the engagement ID.
5. If <userdefinedengagementid> is empty, an attempt is made to look up the engagement ID using <name> in the XML.

EngagementBillingRate

If the rateid is not provided in the <engagementbillingrate> node in the XML, the UpdateByXML method uses the following fields/nodes to find the billing rate ID:

- Engagement – mandatory, and in all cases is included in the lookup.
- Resource – required if the rate is associated with a resource

- BillingRole – required if the rate is associated with a billing role
- BillingOffice – required if the rate is associated with a billing role

EngagementBillingRateHistory

If the billingrateseffectivedatesid is not provided in the <engagementbillingratehistory> node in the XML, the UpdateByXML method uses the following fields/nodes to find the billingrateseffectivedates ID:

- EffectiveDate – mandatory, and in all cases is included in the lookup.
- BillingRateId – optional, if not provided in the XML, the method will look up the billingrateid from the engagementbillingrate node that has the same billingrateindex number as the engagementbillingratehistory node.

EngagementFixedFee

If the fixedfeeid is not provided in the <engagementfixedfee> node in the XML, the UpdateByXML method uses the following fields to find the FixedFee ID:

1. Engagement - mandatory, and in all cases is included in the lookup.
2. If UserDefinedFixedFeeId has a value, the method attempts to lookup the Fixed Fee ID using <userdefinedfixedfeeid> in the XML.
3. If UserDefinedFixedFeeId does not have a value, then the lookup is based on Deliverable, BillingAmount and BillingDate, all of which should have values.

EngagementFixedFeeItem

If the fixedfeeid is not provided in the <engagementfixedfeeitem> node in the XML, the UpdateByXML method uses the following fields to find the FixedFeeItem ID:

1. ParentFixedFee - mandatory, and in all cases is included in the lookup.

   The method extracts the <id> value from the XML; if <id> is empty, an attempt is made to look up the parent Fixed Fee ID using <userdefinedid> and if <userdefinedid> is empty, it uses <name>.
2. If UserDefinedFixedFeeId has a value, the method attempts to lookup the Fixed Fee Item ID using <userdefinedfixedfeeid> in the XML.
3. If UserDefinedFixedFeeId does not have a value, then the lookup is based on Deliverable, BillingAmount and BillingDate, all of which should have values.

EngagementProduct

If the engagementproductid is not provided in the <engagementproduct> node in the XML, the UpdateByXML method uses the following fields to find the EngagementProduct ID:

1. Lookup is based on Engagement, Product, and ProductDate, all of which should have values.

   For Product:

   - the method extracts the <id> value from the XML
   - if <id> is empty, an attempt is made to look up the Product ID using <name>

EngagementProjectedResource

If the projectedresourceid is not provided in the <engagementprojectedresource> node in the XML, the UpdateByXML method uses the following fields to find the ProjectedResource ID:

1. The Engagement, BillingRole, StartDate, and FinishDate – mandatory and in all cases are included in the lookup.

   For BillingRole:

   - the method extracts the <id> value from the XML
   - if <id> is empty, an attempt is made to look up the Billing Role ID using <name>

2. If either or both of Resource and Function have values, then they are used along with the four mandatory fields.

   For Resource:

   - the method extracts the <id> value from the XML
   - if <id> is empty, an attempt is made to look up the Resource ID using <userdefinedid>
   - if <userdefinedid> is empty an attempt is made to look up the Resource ID using <firstname> and <lastname> in the XML.

   For Function:

   - the method extracts the <id> value from the XML
   - if <id> is empty, an attempt is made to look up the Function ID using <name>

EngagementRequestProcessingRule

If the engrequestbillingruleid is not provided in the <engagementrequestprocessingrule> node in the XML, the UpdateByXML method uses the following nodes/fields to find the EngRequestBillingrule ID:

Engagement and RequestType. If RequestType is empty, an attempt is made to look up the RequestType using <requesttypedescription>.

EngagementRequestSLA

If the engagementslaid is not provided in the <engagementrequestsla> node in the XML, the UpdateByXML method uses the following fields to find the EngagementSLA ID:

1. Engagement and Product – mandatory and in all cases are included in the lookup.

   For Product:

   - the method extracts the <id> value from the XML
   - if <id> is empty, an attempt is made to look up the Product ID using <name>.

2. If either or both of Priority and RequestType have values, then they are used along with the two mandatory fields.

   For Priority:

   - the method extracts the <id> value from the XML
   - if <id> is empty, an attempt is made to look up the Priority ID using <name> in the XML.

   For RequestType:

   - the method extracts the <requesttype> value from the XML
   - if <requesttype> is empty, an attempt is made to look up the RequestType using <requesttypedescription>.

EngagementSplitBillingRule

If the splitengagementid is not provided in the <engagementsplitbillingrule> node in the XML, the UpdateByXML method uses the following fields to find the SplitEngagement ID:

MainEngagement and Customer – mandatory.

For both nodes, the method extracts the <id> value from the XML:

- if <id> is empty, an attempt is made to look up the MainEngagement/Customer ID using <userdefinedid> in the XML
- if <userdefinedid> is empty, an attempt is made to look up the MainEngagement/Customer ID using <name> in the XML.

EngagementWorkCode

Lookup is based on Engagement.

The UpdateByXML method can update the DefaultWorkCodeCategory and DefaultWorkCode fields, and change the engworkcode selection list using the <sxmlworkcodes> node.

For DefaultWorkCodeCategory and DefaultWorkCode:

- the method extracts the <id> value from the XML
- if <id> is empty, an attempt is made to look up the WorkCodeCategory/WorkCode ID using <userdefinedid> in the XML
- if <userdefinedid> is empty, an attempt is made to look up the WorkCodeCategory/WorkCode ID using <name> in the XML.

For each <engworkcode> within the <sxmlworkcodes> node:

- the method first extracts <workcodecategoryid> and <workcodeid> values from the XML
- if <workcodecategoryid> and <workcodeid> are empty, an attempt is made to look up the WorkCodeCategory/WorkCode ID using <userdefinedworkcodecategoryid> and <userdefinedworkcodeid> in the XML
- if <userdefinedworkcodecategoryid> and <userdefinedworkcodeid> are empty, an attempt is made to look up the WorkCodeCategory/WorkCode ID using <workcodecategory> and <workcode>in the XML.

EngagementWorkLocation

Lookup is based on Engagement.

The UpdateByXML method can update the DefaultWorkLocationGroup and DefaultWorkLocation fields, and change the engworklocation selection list using the <sxmlworklocations> node.

For DefaultWorkLocationGroup and DefaultWorkLocation:

- the method extracts the <id> value from the XML

- if <id> is empty, an attempt is made to look up the WorkLocationGroup/WorkLocation ID using <userdefinedid> in the XML
- if <userdefinedid> is empty, an attempt is made to look up the WorkLocationGroup/WorkLocation ID using <name> in the XML.

For each <engworklocation> within the <sxmlworklocations> node:

- the method first extracts <worklocationgroupid> and <worklocationid> values from the XML
- if <worklocationgroupid> and <worklocationid> are empty, an attempt is made to look up the WorkLocationGroup/WorkLocation ID using <userdefinedwlgid> and <userdefinedwlid> in the XML
- if <userdefinedwlgid> and <userdefinedwlid> are empty, an attempt is made to look up the WorkLocationGroup/WorkLocation ID using <worklocationgroup> and <worklocation> in the XML.

Forcing a new record to be created

To ensure that a new record is always created rather than updating an existing record, use the nullid in the ID field in the following format, including the brackets:

{00000000-0000-0000-0000-000000000000}

You can set this as a default in your transformation file without it being included in the input file, if desired.

If it is unlikely that the name of the entity would be unique, or if there will not be an exact match between the data coming into Changepoint and the Changepoint field, you should use the user-defined ID, a PK field in your inbound file, or the ID (if it can be retrieved). For example, if there is more than one customer with the same name in Changepoint, or if the external system has “John Smith” and Changepoint has “Smith, John,” then you could not use “name” to look up the customer or resource.)

Creating opportunity sub-items

You can create sub-items for an opportunity, such as opportunityservice, opportunityproduct, opportunityexpense, opportunityfixedfee. This topic describes points to keep in mind when creating opportunity sub-items.

Include the action field in the XML file

You must include the "action" field in your XML file. The Changepoint field name would be opportunityservices.opportunityservice.action. The name would vary for each sub-item, Data type = String, value is one of: 1 (for create), 2 (for update), 3 (for delete), 9 (no change) for each sub-item. The transactions must be unique for each action and must be mapped to separate configuration files. That is, to create and update services, you will need two different entity configuration files, one with the “create” action in it, and the other with the “update” action in it, plus the other accompanying data).

Billing role name must be unique for opportunity services

For opportunity services, the billing role name must be unique or it cannot be used in your data file with the “name” parameter. For example, if you have a “developer” billing role in both the “North American services” and “European services” billing offices, you must do one of the following:

- rename one of the billing roles so that they are unique
- use the billing role ID instead, since it is unique

Each sub-item requires its own transformation file

The same transformation file can be used for the create or update of the sub-object provided that the "action" is passed in the input file. The "action" can be defined as a static field with a default set to 1 in the transformation file, in which case two transformation files will be required per sub-object.

Parent opportunity must already exist in Changepoint

Creation of an opportunity and its sub-item at the same time in the same input file is not supported.

The input file can contain multiple rows of data for the same sub-object type.

You must use an opportunity lookup field (for example opportunity PK field, ID, Name, Userdefined ID) that is the same for each row of data in the input file. Provided that the lookup field is the same, you can create multiple opportunity services either for the same opportunity or for different opportunities.

Examples

To create three opportunity service records for two different (existing) opportunities as provided below:

- The "OpportunityID" field represents the PK field which is mapped to an opportunity UDF text field.
- The action field =1 to create.

	A	B	C	D	E	F	G	H	I	J	K
1	Opportunit	serviceid	ServiceSta	ServiceEn	BillingOffic	Currency	Billingrolell	Estimate	Billingrate	Costrate	action
2	PK-IF-171		3/24/2011	1/31/2012	Corporate	CAD	D807DF89	50	77.8	66.5	1
3	PK-IF-171		3/24/2011	1/31/2012	Corporate	DEM	D807DF89	60	77.85	66.5	1
4	PK-IF-190		3/31/2011	1/31/2012	Corporate	USD	D807DF89	70	80	66.5	1
5											

This is an example of a main item (opportunity) and sub-item (opportunity services) in separate .csv file format transactions. The ID of the second file must be known, for example, the PK field of the opportunity. The parent opportunity must exist before the sub-items can be created.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	id	Customer	name	descriptio	salesrep	workgrou	currency	includeinf	initialclos	revenuefc	type		
2	wr006	wr001	Warren's	Mega Wra	148570	Enterprise	Canadian	TRUE	########	75000	Software Development		

	A	B	C	D	E	F	G	H	I	J
1	id	serviceid	ServiceSta	ServiceEn	BillingOff	Currency	Billingrole	Esttime	Billingrate	Costrate
2	wr006		########	########	Corporate	USD	Develope	50	77	66
3	wr006		########	########	Corporate	USD	Develope	60	77	66

4. Troubleshooting Changepoint Integration Framework

Troubleshooting Microsoft Service Bus

Service bus fails to start

Verify that you have met all the requirements for the service bus as outlined in the *Changepoint Product Architecture and Technology Matrix*.

"Exception: System.Web.Services.Protocols.SoapHeaderException: SOAP header Security was not understood." error

The Changepoint Web Services API is required for the Integration Framework.

To enable the Web Services API, you must uncomment the Microsoft.Web.Services2 element in the `<cp_root>\API\CP Web Services\Web.config` file.

Checking the ports on the firewall

Check that the Service Bus for Windows Server ports are open:

- 9000-9004 - Internal communication
- 9354 - Service Bus brokered messaging
- 9355 - Service Bus Management
- 5672 (optional) - AMQP support for brokered messaging

Viewing Service Bus for Windows Server events in the Event Viewer

Service Bus for Windows Server writes all error, warning, and information events to the event viewer. These events appear in the application logs in the Windows Event Viewer application.

The Event Viewer uses the following monitors to identify events generated by Service Bus for Windows Server:

- Service Bus Message Broker
- Service Bus Gateway
- Fabric

By default, verbose Service Bus for Windows Server events are not traced. This procedure describes how to use the Event Viewer to enable event tracing and to locate Service Bus for Windows Server events.

To view events in the Event Viewer

1. Open the **Event Viewer**.
2. From the **View** menu, select **Show Analytic and Debug Logs**.
3. Under **Applications and Services Logs** then **Microsoft-ServiceBus**, locate the Trace channel log for Service Bus for Windows Server .
4. Right-click Analytic and Debug, and then click the Enable Logging check box to start the event tracing. For more information about channels, see Event Logs and Channels in the Windows Event Log.
5. Service Bus for Windows Server events appear in the event window for the Debug channel. Double-click an event in the list to view detailed information. You can view an event in XML or “Friendly View” format.

Troubleshooting the Changepoint Export Publishing Service

If messages are not getting to the destination, the problem could be in the applications themselves (Changepoint, Service Bus, or third-party application) or the problem could be something that occurs en route between the applications.

If the messages are not getting from Changepoint to the third-party application, use the following guidelines to determine the source of the problem.

1. Enable logging for the Changepoint Export Publishing Service. For information on enabling logging, see the "Installing the Changepoint Export Publishing Service" section on page 52.
2. If the logging is enabled and log files are not being generated, then the issue is in the Changepoint configuration. Check that the export publishing schedules are set up properly in Changepoint Administration:
    a. Verify that the schedule has been activated, and has been run (that is, the start date is not in the future).
    b. Check the Export Publishing Audit Report and the Export Publishing Error Report in Changepoint Administration.

       For more information, see the Changepoint Administration Guide.

3. If the files are successfully generated in Changepoint, then check Service Bus for problems.

Changepoint Export Publishing Service XML files

The XML files are available if you configured the Changepoint Export Publishing Service to save the XML files. The XML files are saved to the specified location using the following naming pattern:

`CPExpPub`*`yyyyMMdd_hhmm_filenumber`*`.xml`

where *yyyyMMdd_hhmm* is the server local date and time when the XML file is generated.

The structure of the XML files is as follows:

```
<root>
   <exppubname name="MyExport" exporttype="R" batchnumber=""
         culturename="en-US" filenumber="3" totalfiles="11"
         numberofrecords="500">
      <groupname name="MyExportGroup1">
         <format>timedate=dd-MM-yyyy</format>
         <data>...</data>
         <data>...</data>
         <data>...</data>
         ...
         <data>...</data>
      </groupname>
   </exppubname>
</root>
```

The various elements and attributes in the example are explained in the following table.

Element	Attribute	Description
exppubname	name	Name of the export publishing setup.
	exporttype	Export type, which is either "R" for regular or "B" for batch.
	batchnumber	The batch number for batch exports.
	culturename	Culture of the server that the data was sent from.

Element	Attribute	Description
	filenumber	Identification number that represents the position of the file in the sequence of files included in the export.
	totalfiles	Total number of files included the export.
	numberofrecords	Maximum number of records in a file
groupname		Name of the export group. There is only one groupname element per XML file, so if there are multiple export groups, then each group will have its own XML file.
format		Format for dates and numbers. If specified, this format overrides the format normally associated with the culturename attribute.
data		Export record.

Changepoint Export Publishing Service log files

If you enable logging, the log files are saved to the specified location using the following naming pattern:

`CPExpPub`*`yyyyMMdd_hhmm`*`.log`

where *yyyyMMdd_hhmm* is the server local date and time when the log file is generated.

The following events are logged in the log files:

- Errors during processing – you can check the Export Publishing Error Report for more information. For more information, see the *Changepoint Administration Guide*
- Errors during the queuing stage – refer to the Windows Event Viewer for more details.
- Details of processing that has completed successfully.

Application pool issues

"Server Error in '/ExternalEndpoint' Application" error when browsing the SFDC web service

Ensure that the application pool DefaultAppPool is configured for .NET 4.0.

"Message: Exception: Found 2 DNS claims in authorization context" error in error log after upgrading to Changepoint 2017

A change to the identity authorization model was introduced in .NET Framework 4.6.1 that caused an incompatibility with the Changepoint Integration Framework. Specifically, the change affects the behavior of the X509CertificateClaimSet.FindClaims method in an environment that has multiple DNS entries.

Microsoft recommends adding the following configuration setting to the `<runtime>` section of the configuration files that have a Microsoft.ServiceBus.ConnectionString set, which for Changepoint are the Communication Dispatcher service, Data Mapper service, and Export Publishing service:

```
<runtime>
   <AppContextSwitchOverrides
   value="Switch.System.IdentityModel.DisableMultipleDNSEntriesInSANCertificate=true" />
</runtime>
```

Reference: https://msdn.microsoft.com/en-us/library/mt620030(v=vs.110).aspx

Data mapping issues

Invalid characters

Ensure that you use valid characters. The hexadecimal values of the allowed characters are summarized in the following table.

Hex value (or range of values)	Comments
0x9	Tab character
0xA	New line character
0xD	Carriage return character
0x20 – 0xD7FF	
UxE000 – UxFFFD	
0x10000 – 0x10FFF	

Mapping multiline fields

There is currently no way to split a mapping of a multiline SFDC field into more than one Changepoint field.

The multiline SFDC field in the following example would be populated into the Changepoint addressline field. The Changepoint addressline2 and addressline3 fields would not be populated.

```
<mapping>
   <changepoint>customeraddress1.addressline</changepoint>
   <external>BillingStreet</external>
</mapping>
```

You must set up text fields to map separate address lines into Changepoint.

Troubleshooting transformation files

Incorrect datatype (-1 System operation error)

Although the field names suggest a Boolean datatype, EngagementWorkcode and EngagementWorklocation “selected” fields are string type, with values 1=selected and 0=not selected.

If you pass values of “true” or “false,” the result is “-1 System operation error”.

Following is a valid fragment of an EngagementWorkcode field definition:

```
<field>
  <datatype>String</datatype>
  <order>10</order>
  <location>
    <index>10</index>
    <start></start>
    <end></end>
  </location>
  <mapping>
    <changepoint>engagementworkcode.sxmlworkcodes.root.engworkcode.selected
</changepoint>
    <external>engagementworkcode.sxmlworkcodes.root.engworkcode.selected
</external>
  </mapping>
</field>
```

Malformed XML

After creating or editing an XML transformation file, open the file in Internet Explorer. If there are missing or extra tags, Internet Explorer displays an error.

Truncated fields

Symptom: When using the Fixed file type, it cuts off one character for each field based on your start and end location.

Solution: Each field must have a space added to the end, so when counting the start and end location, this space has to be counted. For example:

Field1 (starts from 0, ends at 14)

When configuring the mapping, 15 will be the end point:

```
<location>
   <start>0</start>
   <end>15</end>
</location>
```

pkfield is ignored

Symptom: An entityid is passed in with nullid. A pkfield is set up, but it always creates a new entity instead of updating the existing entity based on the pkfield lookup. The pkfield is ignored.

Solution: If you want the entity updated, do not pass in the nullid.

Troubleshooting Changepoint Windows services

The following sections give troubleshooting tips for issues with various Changepoint Windows services.

Changepoint Communication Dispatcher Service does not start

If the Changepoint Communication Dispatcher Service does not start, then logging will not be available. Check the event viewer for errors that prevent the service from starting.

Changepoint DataMapper Service does not start successfully

Check the Event Viewer for details. Also check the error.log file in the Changepoint DataMapper Service folder.

Inbound transaction errors

If you see any errors in the error.log file when you try an inbound transaction, also check in the \API\APILogs folder for further details.

You can look in the “received” folder (either in \data or \archive under the data mapper services folder) for the inbound data in its “raw” format (as it was received). The files in the “in process” folder are the transformed XML format of the data.

“Failure sending mail” messages

If you receive 'failure sending mail' messages in the Changepoint DataMapper Service\error.log verify the IP address of the SMTP mail server in the following file:

```
Changepoint DataMapper Service\
Changepoint.IntegrationServices.WindowsServices.DataMapper.exe.config
```

DataMapper cannot process message because the routing ConfigId does not point to an active/valid configuration file

Check the file name specified in the **Configuration ID** field in the Export Publishing Setup in Changepoint Administration. The file name is case sensitive.

Configuration folder

The path where the Changepoint DataMapper Service is installed contains a configuration folder. Transformation files in this folder must match the transformation files identified in the Adapter.xml for the Changepoint Communication Dispatcher service. Avoid having additional transformation files in the configuration folder and avoid using this folder as a working folder.

Changepoint Export Publishing Service stops working

If the Changepoint Export Publishing Service stops working after a time, verify that the Adjust Daylight Saving job has been run on the SQL server for the Changepoint database.

Troubleshooting SFDC issues

Contact not updated when changing the customer associated with a contact

In SFDC, the associated customer for an existing contact can be changed, which is not allowed in Changepoint. When going inbound, the contact in Changepoint will not be updated if the associated customer has been changed in SFDC. No error message is logged.

Index

T

X