SDL Tridion 2013 Deprecation Guide

Content Management Technologies Division of SDL

SDL Tridion 2013 Deprecation Guide

Table of contents

Chapter 1 About this document

The Deprecation Guide explains which third-party software and which SDL Tridion product features are deprecated, and what that deprecation means; which backwards compatibility issues you may encounter after you upgrade to the current product release; and which steps you can take to migrate from a (partly) deprecated implementation of SDL Tridion to a recommended one.

Deprecation

This release of SDL Tridion supports a number of deprecated third-party software product releases, such as operating systems or Web browsers. The **deprecation status** of a third-party software product release can be "Definitely" or "Probably":

- **Definitely**—support for the software product release will definitely be dropped in the next version release (that is, the next release that is not a Service Pack) of SDL Tridion.
- **Probably**—support for the software product release will probably be dropped in the next version release. Whether or not the software product release will actually be dropped typically depends on the exact time of a newer version of the third-party product.

This release of SDL Tridion also contains deprecated features such as APIs or the use of a certain programming language. In this case, deprecation means that the feature will remain supported in the foreseeable feature but that all new features are only offered in the new API or the new programming language.

Migration assistance

If your implementation contains deprecated product features you may want to migrate your implementation to a newer, recommended equivalent of that feature. Sometimes, such a migration is relatively trivial, such as when migrating from the Content Broker configuration to the new Storage Layer configuration. At other times, such a migration may involve more work, such as when migrating from VBScript-based templates to modular templates. This kind of migration involves reimplementing your business logic in a new framework.

Chapter 2 Content Manager deprecation status

This is a detailed description of the exact deprecation status of the third-party software marked as 'deprecated' for the Content Manager server software and its associated clients.

2.1 Content Manager server and client operating system deprecation status

Operating systems deprecation status for the Content Manager server and Content Manager Explorer client.

Content Manager server operating systems

Recommendation: SDL recommends the following operating system: Microsoft Windows Server 2012 (x64).

Supported: SDL supports Microsoft Windows Server 2008 R2 SP1 (x64).

Deprecated operating systems: No operating systems are now deprecated.

Content Manager Explorer operating systems

Recommendation: SDL recommends the following operating system: Microsoft Windows 8 (x86 or x64) or Apple Mac OS X 10.7 Lion.

Supported: SDL supports Microsoft Windows 7 SP1 (x86 or x64).

Deprecated operating systems: No operating systems are now deprecated.

2.2 Content Manager database deprecation status

The deprecation status of databases on the Content Manager side.

Recommendation

SDL recommends the following databases:

- Oracle Server 11.2.0.3 with one of the following patches applied:
  - For Linux/UNIX systems: PSU 11.2.0.3.2
  - For Windows systems: 11.2.0.3 Patch 2 or higher (for Windows)

- Microsoft SQL Server 2012 SP1

SQL Server databases

Supported SQL Server databases: SDL supports Microsoft SQL Server 2008 R2 SP2.

Deprecated SQL Server databases: No SQL Server databases are now deprecated.

Oracle databases

Supported Oracle databases: SDL supports Oracle 10.2.0.5.

Deprecated Oracle databases: No Oracle databases are now deprecated.

2.3 Content Manager .NET and Java deprecation status

The deprecation status of .NET Framework and Java releases on Content Manager.

.NET deprecation status

Recommendation: SDL recommends the following .NET version: Microsoft .NET Framework 4.5, CLR 4.5

Supported .NET release: SDL supports the following .NET version: Microsoft .NET Framework 4.0, CLR 4.0

Java deprecation status

Content Manager server requires Java for the following functionalities:

- Publishing content
- Searching content

Content Manager requires a 64-bit version of Java.

Recommendation: SDL recommends the following version of Java: Java SE 7.0

Deprecated Java release: Java SE 6.0 is currently deprecated, and will probably be dropped in the next version release.

2.4 Content Manager Web application server deprecation status

The deprecation status of Web application servers on Content Manager.

Deprecation status

Recommendation: SDL recommends the following Web application server: Microsoft IIS 8

Supported IIS release: SDL supports Microsoft IIS 7.5

Deprecated IIS releases: No IIS release is now deprecated.

2.5 Content Manager Explorer internet browser deprecation status

Supported internet browsers and their deprecation status.

Deprecation status

Recommendation: SDL recommends the following internet browser: Microsoft Internet Explorer 10.0 (if using Windows 8 or Windows Server 2012), Microsoft Internet Explorer 9.0 (if using another version of Windows), or Safari 5.1 for the Mac (if using Mac OS X)

For Google Chrome and Mozilla Firefox, SDL adopts the policy that the Content Manager Explorer GUI should run on the latest version of those browsers.

Supported browser: SDL supports Microsoft Internet Explorer 9.0 and 10.0, and Safari 5.1 for the Mac.

Deprecated internet browsers: No internet browsers are deprecated.

2.6 Microsoft Visio deprecation status

The deprecation status of Microsoft Visio, as used by the Visio Workflow Designer.

Microsoft Visio

Recommendation: SDL recommends the following version of Microsoft Visio: Microsoft Visio 2010 SP1 (32-bit), English version

Supported or deprecated Visio releases: No other versions of Microsoft Visio are supported or deprecated.

Chapter 3 Content Delivery deprecation status

This is a detailed description of the exact deprecation status of the third-party software marked as 'deprecated' for Content Delivery.

3.1 Content Delivery operating system deprecation status

Operating systems deprecation status for Content Delivery.

Recommendation

SDL Tridion recommends using the latest version of the supported operating systems.

Content Delivery Windows operating systems

Supported operating systems: SDL supports Windows 2008 R2 SP1 (x64)

Deprecated operating systems: No Windows operating systems are currently deprecated.

Content Delivery non-Windows operating systems

Supported operating systems: The following non-Windows operating systems are currently supported for Content Delivery:

- HP-UX 11i V3
- HP-UX 11i V2
- IBM AIX 6.1
- IBM AIX 7.1
- Redhat Enterprise Linux 5.9
- Redhat Enterprise Linux 6.4
- Sun Solaris 9
- Sun Solaris 10

3.2 Content Delivery and UGC database deprecation status

The deprecation status of databases for Content Delivery and UGC.

Content Delivery databases

Recommendation: SDL recommends one of the following databases:

- Oracle Server 11.2.0.3 with one of the following patches applied:
  - For Linux/UNIX systems: PSU 11.2.0.3.2
  - For Windows systems: 11.2.0.3 Patch 2 or higher (for Windows)
- Microsoft SQL Server 2012 SP1

Supported SQL Server databases: SDL supports Microsoft SQL Server 2008 R2 SP2.

Supported Oracle databases: SDL supports Oracle 10.2.0.5

Deprecated DB2 databases:

DB2 version	Dropped in next version release?
IBM DB2 9.7	Under review

UGC databases

Recommendation: SDL recommends one of the following databases for User-Generated Content, or UGC:

- Oracle Server 11.2.0.3 with one of the following patches applied:
  - For Linux/UNIX systems: PSU 11.2.0.3.2
  - For Windows systems: 11.2.0.3 Patch 2 or higher (for Windows)
- Microsoft SQL Server 2012 SP1

Supported SQL Server databases: SDL supports Microsoft SQL Server 2008 R2 SP2.

Supported Oracle databases: SDL supports Oracle 10.2.0.5

3.3 Content Delivery .NET and Java deprecation status

The deprecation status of .NET Framework and Java releases on Content Delivery.

.NET deprecation status

Recommendation: SDL recommends the following version of .NET: Microsoft .NET Framework 4.5

Supported .NET Frameworks: SDL supports Microsoft .NET Framework 4.0.

Deprecated .NET Frameworks: .NET Framework 3.5 SP1 is currently deprecated, and will probably be dropped in the next version release.

Java deprecation status

Recommendation: SDL recommends the following version of Java: Java SE 7.0

Deprecated Java releases: Java SE 6.0 is currently deprecated, and will probably be dropped in the next version release.

3.4 Content Delivery Web application server deprecation status

The deprecation status of Web application servers on Content Delivery.

Deprecation status

Recommendation: SDL recommends using the latest version of any of the supported Web application servers.

Supported Web application servers: SDL also supports the following Web application servers:

- Apache Tomcat 6.0
- IBM WebSphere 7 (with Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0 installed)
- JBoss Enterprise Server 4.3
- Microsoft IIS 7.5
- Oracle WebLogic Server 11*g* R1

Chapter 4 Custom code development recommendations

This topic provides general recommendations for developing custom code that interacts with SDL Tridion.

.NET developers: target .NET 4.5

Create your .NET assemblies to target Microsoft .NET Framework 4.5.

Java developers: target Java 7

Write your Java code to target Java SE 7.

Custom extension developers: cater for 64-bit operating systems

Enable your custom extensions to run on a 64-bit operating system. Compile .NET code to CPUAny.

Use alternatives for TOM API

You can use the deprecated COM/ASP-based TOM API in various contexts. Depending on the context, migration means one of the following things:

- Change code in templates and in Event Handler code into calls to the TOM.NET API.
- Change all other code that calls the TOM API into calls to the Core Service.

Chapter 5 Deprecated functionality

This section explains which functionality is deprecated in SDL Tridion 2013 for users who wish to migrate to non-deprecated features and technologies.

5.1 Deprecated functionality in Content Manager

This section explains which software components in Content Manager are deprecated in SDL Tridion 2011 SP1 in favor of a newer, improved version. By default, deprecated features are only installed when performing an upgrade, or when you explicitly choose to install these legacy features.

5.1.1 Deprecation of Business Connector and XML Responder

Business Connector and XML Responder are deprecated in favor of the Core Service.

The Core Service is a Web service interface designed to replace the Business Connector Web service (which combined with XML Responder). Migration to the Core Service entails becoming familiar with its functionality and API and turning your Business Connector calls into Core Service calls.

Core Service documentation

To learn what the Core Service is and how you interact with it, consult the documentation for the Core Service in the documentation portal, located in the section "Implementing Content Manager extension points".

Core Service API reference

For more in-depth information about the API, refer to the SDL Tridion World Web site (requires login), which you can find at the following URL: https://www.sdltridionworld.com/downloads/documentation/SDLTridion2011SP1/index.aspx

Once familiar with the Core Service, migration consists of finding and identifying those places in your custom code where you make calls to the Business Connector, and replacing them with equivalent calls to the Core Service.

5.1.2 Deprecation of VBScript/JScript templating

The VBScript/JScript templating framework, which interacts with the deprecated TOM API, is deprecated in favor of the .NET-based Modular Templating framework which interacts with the new TOM.NET API.

The Modular Templating framework was designed to replace the VBScript Templating Framework. Modular Templating helps you to debug, reuse templating features, and separate concerns by accepting C# fragments, .NET assemblies and Dreamweaver Templates as the building blocks of its templates. Migration to Modular Templating entails becoming familiar with its functionality, its client application called Template Builder, and its API, and turning your VBScript templates into Modular Templates.

To learn what the Modular Templating framework is and how you interact with it, consult the documentation for Templating in the documentation portal, located in the section "Implementing Content Manager extension points". This section of the portal not only explains the concepts and principles behind Modular Templating, it also offers a reference for the template expression language, explains the client application called Template Builder, and explains how to debug your templating code.

Performing the actual migration involves reimplementing the business logic of your VBScript templates into modular templates. The size of this task depends on how many templates your implementation contains and what they do. To help you with your migration, SDL Tridion ships with a number of default Template Building Blocks for typical tasks, as well as with a sample Page Template and Component Template.

Note that the Modular Templating framework is read-only; you can no longer write information to Content Manager from a template.

Also, new Publications created in the Content Manager no longer contain a Default Component Template, Default Page Template and Default Template Building Block, written in VBScript, in the `Building Blocks` folder. Instead, a new Publication contains a Folder `Building Blocks\Default Templates \`, containing a minimal set of the default modular templating items that SDL Tridion ships with.

For existing Publications, however, neither the Folder nor the items are added during upgrade. This is because such actions could create a conflict. As a result, if you intend to start using modular templating in an existing Publication, you must first create these default items from Template Builder.

5.1.3 Deprecation of TOM-based Event System

The TOM-based Event System is deprecated in favor of the TOM.NET-based Event System.

The TOM.NET-based Event System offers a more fine-grained and sophisticated way of catching and handling Content Manager events.

Also, it is not possible to use data in .NET that was obtained using MS XML (which is the case when using the old TOM-based Event System). Microsoft Knowledge Base article 815112 explains this issue: http://support.microsoft.com/default.aspx?kbid=815112

Note that calls to the TOM.NET API will no longer trigger old Event Handler. The SDL Tridion World community Web site offers a Legacy Event Adapter to handle this. Refer to https://www.sdltridionworld.com/community/2011_extensions/LegacyEventAdapter.aspx for more information.

Migrating from TOM Event System to TOM.NET Event System

Migrate from your Event System to .NET by mapping TOM-based event triggers to a Subject, Type and Phase in the new TOM.NET-based Event System. You also need to rewrite your original event handler code in .NET.

Steps to execute

1. Determine the item type that triggers your event. For example, if your event trigger is called `OnKeywordLocalizePost`, the item type is Keyword. In your new event code, set your Event Subject to this value.

2. Determine the operation that triggers your event. For example, if your event trigger is called `OnComponentTemplateUndoCheckOutPre`, the operation is Undo Check Out. In your new event code, set your Event Type to this value followed by `EventArgs`, for example, `SaveEventArgs`. However, note the following exceptions:

   - If the old operation is one of the following: `Assign`, `Restart`, `Start` or `Suspend`, add the string `Activity`. For example, if the original event trigger is called `OnActivityInstanceStartPost`, set the Event Type to `StartActivityEventArgs`.
   - If the old operation is `Finish`, check the item type. If the item type is `ActivityInstance`, set the Event Type to `FinishActivityEventArgs`; if it is `ProcessInstance`, set it to `FinishProcessEventArgs`.
   - If the old operation is `SetPublishedTo`, set the Event Type to `SetPublishStatusEventArgs`.
   - If the old operation is `PasteItem`, set the Event Type to `MoveEventArgs` or `CopyEventArgs`.

3. Determine the phase in which your event code is to be triggered, either Post or Pre. If your event trigger named ends in `Pre`, set the Event Phase to `Initiated`, and if it ends in `Post`, set the Event Phase to `Processed`.

   Using these steps, you would translate an event trigger ending in `SavePost` into an Event Type `SaveEventsArgs` and an Event Phase `Processed`. But depending on your wishes, in some circumstances you may want to set the type to `CheckInEventArgs` and the phase to `TransactionCommitted`. Refer to the documentation about Event Handlers to find out if this combination is more appropriate for your situation.

   The following TOM-based Event Triggers do not have a TOM.NET equivalent:

   `OnSchemaGetInstanceDataPost`

   This event trigger does not exist. You can approximate it by the combination of the Subject `Component`, the Type `SaveEventArgs` and the Phase `Initiated`.

   Any event trigger with a `Resolve` or `Render` operation

   There are no event triggers for resolve or render operations. You can implement custom resolving and/or rendering behavior by implementing a custom Resolver and/or Renderer.

4. Rewrite your original event handler code in .NET.

5.1.4 Deprecation of TOM API

The TOM API is deprecated in favor of the Core Service, introduced in SDL Tridion 2011.

Most calls to the TOM API typically occur in templating code and in event handling code, so migration normally consists of migrating your templates and your event handler code to TOM.NET. In all other contexts, the Core Service replaces the COM-based TOM API. If you currently call the TOM from any other code, replace such a call with a Core Service call.

API reference documentation for both TOM.NET and the Core Service is available from the SDL Tridion World Web site at this URL: https://www.sdltridionworld.com/downloads/documentation/SDLTridion2011SP1/index.aspx

Also note that it is not possible to use data in .NET that was obtained using MS XML (which is the case when using the TOM API). Microsoft Knowledge Base article 815112 explains this issue: http://support.microsoft.com/default.aspx?kbid=815112

5.1.5 Other deprecated functionality in Content Manager

This topic lists miscellaneous Content Manager functionality deprecated in SDL Tridion 2013.

LDAP

The following LDAP functionality has changed as of SDL Tridion 2011 SP1:

Combined LDAP and Active Directory authentication

SDL strongly recommends against authentication through LDAP and through Active Directory using a single machine. If you currently have such a setup running, submit a support ticket to SDL Customer Support.

LDAP version 2

Version 2 of LDAP is now no longer supported, in favor of LDAP v3.

Core Service API

Your Core Service client connects to the Core Service through a specific service contract endpoint. The .NET assembly used by your client determines the service contract endpoint:

Service contract version number	Endpoints	Introduced in
2010		SDL Tridion 2011
2011		SDL Tridion 2011 SP1
2012		SDL Tridion 2013

If you upgrade from SDL Tridion 2011, the new Core Service API also has the following impact on existing implementations:

- SDL Tridion 2011 had proxy type names that contained a version number: `CoreService2010Client`, `ICoreService2010` and so on. As of SDL Tridion 2011 SP1, the proxy type names contain no version number, and remain the same across releases: `CoreServiceClient`, `ICoreService` and so on. Modify the proxy type name wherever you use it in your client code.
- Existing endpoints have URLs that contain a version string at the end of the URL (for example, `http://localhost/webservices/CoreService.svc/wsHttp_2010`). New endpoints have the version string as part of the `.svc` filename, for example, `http://localhost/webservices/CoreService2011.svc/wsHttp`.
- Because the old endpoints still exist, your existing calls to the Core Service still work in 2011 SP1. Do note, however, that the Core Service now returns the value `UnknownByClient` if the client proxy receives an enum value from a newer version of the Core Service that the client cannot yet handle. This means that you may have to update and recompile your code to handle this special return value gracefully.
- To benefit from the functionality introduced in the this release, do the following:
    - Add an `endpoint` element in your application configuration that refers to the new Core Service URL:

```
<endpoint name="wsHttp_2011" address="http://localhost/webservices/CoreService2011.svc/wsHttp"
  binding="wsHttpBinding" bindingConfiguration="wsHttp"
  contract="Tridion.ContentManager.CoreService.Client.ISessionAwareCoreService">
```

    - In your code, create a Core Service client for this endpoint:

```
using (SessionAwareCoreServiceClient csClient = new SessionAwareCoreServiceClient("wsHttp_2011"))
{
  // Client code goes here
}
```

    - If necessary, update your code to work with the new endpoint. The new endpoint that is not guaranteed to be completely backward compatible with the old endpoint interface. As a result, it may happen that your client code compiles with errors.

Write operations in Template code

The TOM.NET API is still functionally a read-only interface during rendering and publishing. That is, you cannot create, update or delete Content Manager items while your Template is executing. This behavior also applies to the now deprecated TOM API, meaning that if you have Templates that use the TOM API and perform write actions, these Templates will now fail.

If you do want your old TOM templates to be able to write to the Content Manager, you can configure the Content Manager to allow writing during rendering and publishing. **SDL strongly recommends against this practice as it compromises your security.** You enable Templates to write to the Content Manager as by opening the Tridion Content Manager configuration file `Tridion.ContentManager.config` (located in the `config` subfolder of the Tridion Content Manager root location), and adding an attribute `allowWriteOperationsInTemplates`, set to `true`, to the element called `tridion.contentmanager.security`.

Avoid unmanaged binaries

Both the deprecated TOM API and the TOM.NET API have the possibility of rendering and publishing a binary resource without specifying a related Multimedia Component (either by omitting the parameter or explicitly setting it to `Nothing` or `null`). SDL strongly recommends against this practice.

Core Service 2010 endpoints dropped

Because they were not forward-compatible, Core Service endpoints introduced in SDL Tridion 2011 (labeled '2010') were dropped in SDL Tridion 2013.

5.1.6 Deprecated functionality in Content Manager Explorer

The SDL Tridion 2011 release completely re-engineered Content Manager Explorer. This topic lists the changed behavior resulting from these changes.

Custom extensions

Custom extensions you developed for earlier versions of Content Manager Explorer will no longer work. Refer to the Content Manager implementation topics to learn about the new GUI extension mechanism.

Custom URLs for Schema fields

If you are upgrading from a release earlier than SDL Tridion 2011, and if your implementation includes Custom URLs (that is, making Component field labels clickable by linking the Schema field to a Web resource), then any references to the field, its properties, and its container in the linked Web resource no longer work. As of SDL Tridion 2011, Web resources linked from custom URLs now have access to a JavaScript which you can use to retrieve fields and their properties.

Custom Pages

Any Custom Pages you created will need to be recreated in the new GUI. Refer to the Content Manager implementation topics for more information.

User preferences

Any user preferences, such as interface language, but also, for example, column widths in tables, will need to be reset by hand.

5.2 Deprecated functionality in Content Delivery

This section explains which software components in Content Delivery are deprecated in SDL Tridion 2011 SP1 in favor of a newer, improved version.

5.2.1 Deprecation of Search Filter API

As of the 2011 release, SDL Tridion uses the Content Broker Query API for search filtering functionality . Before 2011, Content Delivery offered a separate API for this in `Tridion.ContentDelivery.DynamicContent.Filters`.

The old Search Filter API still exists, but is deprecated. However, you may wish to migrate your Search Filter code to new Content Broker Query code for the following reasons:

- The Search Filter API may no longer be supported in the next release of SDL Tridion.
- The Broker Query API for filtering offers new functionality for querying custom metadata easily.

To perform such a migration, you will need to rewrite your filtering code. The Content Broker Query classes are in one of the following locations:

- .NET: `Tridion.ContentDelivery.DynamicContent.Query`
- Java: `com.tridion.broker.querying`

The following list provides an overview of the Content Broker Query classes you should use and their old Search Filter equivalents:

- The Broker Query class `ItemCreationDateCriteria` replaces the Search Filter call `Query.AddCriteria(creationdate, ...)`
- The Broker Query class `ItemModificationDateCriteria` replaces the Search Filter call `Query.AddCriteria(modifieddate, ...)`
- The Broker Query class `ItemInitialPublishDateCriteria` replaces the Search Filter call `Query.AddCriteria(initialpublishdate, ...)`
- The Broker Query class `CustomMetaKeyCriteria` replaces the Search Filter call `Query.AddCustomMetaQuery("KEY_NAME")`
- The Broker Query class `CustomMetaValueCriteria` replaces the Search Filter call `Query.AddCustomMetaQuery("KEY_NAME=VALUE")`

For more detailed information on the Content Broker Query API, refer to the implementation topic about Dynamic content querying and filtering.

5.2.2 Deprecation of old storage framework

As of the 2011 release, SDL Tridion release replaces the Content Broker configuration file, `cd_broker_config.xml`, with a Storage Layer configuration file, `cd_storage_conf.xml`, which configures how and where Content Delivery stores items. This also affects custom storage home classes.

Content Delivery automatically transforms your existing deprecated Content Broker configuration into a Storage Layer configuration. In most cases, this results in a working Storage Layer configuration that works the same way as your Content Broker configuration. But in some cases, you may need to update the configuration manually. The documentation topics about upgrading Content Delivery explain how.

The most important differences between the Content Broker configuration and the Storage Layer configuration are:

- In the Storage Layer configuration, you can configure multiple storage media and store various types of content in different media.
- The Storage Layer configuration no longer requires you to use Broker bindings.
- The Storage Layer configuration no longer requires you to configure a `QueryGenerator` element.

To support backward compatibility, SDL Tridion 2011 SP1 automatically detects your existing Content Broker configuration file and applies an XSLT to transform the file into an equivalent Storage Layer configuration. It uses this configuration and saves the result of the transformation to a file called `cd_storage_conf.xml.transformed`.

You can choose to continue using your old Content Broker file and having it transformed, or you can rename `cd_storage_conf.xml.transformed` to `cd_storage_conf.xml`, and use it instead. However, the Storage Layer configuration that the XSLT produces is by no means optimal, and it is recommended that you check and improve the transformed file yourself. Refer to the installation documentation for more information.

If you have created your own custom home storage classes for several Content Broker bindings, SDL Tridion 2011 will be able to upgrade them only if they implement one of the following home storage bindings (the strings you see are the values of the `Name` attribute in the `Binding` element in your Content Broker configuration):

- `Page`
- `Binary`
- `ComponentPresentation`
- `PageMeta`
- `BinaryMeta`
- `ComponentPresentationMeta`

In the event that your Content Broker configuration file contains customizations that the XSLT does not transform correctly, there are several ways to upgrade your configuration:

- Reimplement your customization using the Storage Layer framework. Refer to the installation documentation for more information.
- Change the XSLT. For your convenience, SDL Tridion also ships with a separate `.xsl` file that contains the XSLT being applied. This file is called `cd_storage_conf.xsl` and is located on the installation media in the folder `Content Delivery\resources\xslt\`. If you want, you can try and modify this file to transform your customizations properly. For more information about the Storage Layer configuration, refer to the installation documentation.
- Extend the Storage Layer. Refer to the Content Delivery implementation topics for details.
- Present your scenario to the SDL Tridion community in SDL Tridion World (http://www.sdltridionworld.com).
- Contact SDL Tridion Customer Support.

5.2.3 Deprecation of ASP API and ASP.NET API used in 5.3 SP1 and earlier

The ASP API and the ASP.NET API delivered with releases up to and including SDL Tridion R5 5.3 SP1 are deprecated in favor of the ASP.NET API introduced in SDL Tridion 2009. Inline code is also deprecated in favor of .NET Server Controls.

The ASP.NET API introduced in SDL Tridion 2009 makes the Linking Service, Content Broker Service and WAI Service (all Windows services) redundant, and so these are also deprecated. (If you move to the new ASP.NET API, you no longer need them.) You also no longer need to configure multiple instances of Content Delivery in an instances configuration file.

If you create custom code that talks to one of the deprecated Content Delivery APIs (the ASP API or the ASP.NET API offered until SDL Tridion R5 5.3 SP1), consider changing your API calls to calls to the ASP.NET API introduced in SDL Tridion 2009.

If you use either the old or the new ASP.NET API, SDL strongly recommends to stop using inline script generation in favor of ASP.NET Server Controls to make your code easier to manage. For ASP, this is not an option because Server Controls do not exist in ASP.

API reference documentation for the new ASP.NET API of Content Delivery is available from the SDL Tridion World Web site at this URL: https://www.sdltridionworld.com/downloads/documentation/SDLTridion2011SP1/index.aspx

5.2.4 Deprecated Senders and Receivers

In SDL Tridion 2011 Senders have a new Java class and Receivers are no longer used. SDL recommends to update your Transport Service configuration file on the Content Manager. Also, if you use custom Senders and Receivers, you must update your code.

Senders

Senders no longer extend the `com.tridion.transport.Sender` class, but instead implement the `com.tridion.transport.connection.TransportConnector` interface.

For default Senders, this means that an XSLT stylesheet automatically changes the `Class` attributes of the default Senders in your existing `cd_transport_conf.xml` to reflect the change. SDL recommends that you edit the `Class` attribute by hand to point to the new classes.

As far as your own custom Senders are concerned, you must reimplement such a Sender, so that it no longer extends `com.tridion.transport.Sender`, but instead implements `com.tridion.transport.connection.TransportConnector`.

Receivers

Receivers no longer exist. Your custom Receiver can no longer be configured in the Content Deployer configuration file, `cd_deployer_conf.xml`.

Content Delivery automatically transforms your existing Content Deployer configuration (which contained Receivers) into a new one. You can also update your configuration file yourself by hand by applying the XSLT. The documentation topics about upgrading Content Delivery explain how.

5.2.5 Deprecated metadata storage on the file system

Metadata stored on the local file system is deprecated as of SDL Tridion 2011 SP1 in favor of storage of metadata in the database. Storing metadata in a database requires database licences.

Specifically, the types of metadata impacted are:

- `BinaryMeta`
- `BinaryVariant`
- `ComponentMeta`
- `ComponentPresentationMeta`
- `ItemMeta`
- `LinkInfo`
- `PageMeta`
- `Reference`
- `Schema`

5.2.6 Deprecation of UGC SpamFilter

The `SpamFilter` interface, which was a part of User-Generated Content in SDL Tridion 2011 SP1, has been deprecated in favor of the `Validator` interface.

5.3 Deprecated Translation Manager functionality

This topic describes Translation Manager functionality that is deprecated in SDL Tridion 2011 SP1.

Deprecation of COM-based API

The COM-based API Translation Manager called from the TOM-based Event System and TOM-based Templates is deprecated; use the .NET API which can call the TOM.NET-based Event System and TOM.NET-based Templates.

Deprecation of Publication Configuration (Translation Manager .NET API.)

You can now configure translations on Organizational Items as well as on Publications. To reflect this change, `Tridion.TranslationManager.DomainModel.Api.PublicationConfiguration` is deprecated; use `Tridion.TranslationManager.DomainModel.Api.TranslationConfiguration` instead.

Deprecation of ANL files

The use of ANL files is deprecated. You should now set which fields you want translated in your Schemas. Translatable fields are sent in ITS (Internationalization Tag Set) so your translation system must support ITS (SDL WorldServer 10.1 or SDL TMS 2011 SP1 onwards). ANL files remain supported for older versions of SDL TMS.

After upgrading, you need to configure your Schemas, set other translation properties, configure the `TranslationManager.xml` file, and configure your translation system to use ITS. For more information, refer to the upgrade documentation for SDL Tridion 2011 SP1.

5.4 Deprecated Audience Manager and Outbound E-mail functionality

This topic describes Audience Manager and Outbound E-mail functionality that is deprecated in SDL Tridion 2011 SP1.

Outbound E-mail Content Delivery (Java) API

In the `com.tridion.marketingsolution.contact`:

- `int id = getGroupId()` and `void setGroupId(int id)` are deprecated; use get/set AddressBook instead.
- `Map details = getExtendedDetails()` and `void setExtendedDetails(Map details)` are deprecated; use `ExtendedDetails details = getDetails()` instead. Also, there is no `setDetails()` as modifications are done on the returned collection from `getDetails()`.

Audience Manager API (.NET)

The `Tridion.OutboundEmail.ContentManagement` namespace is deprecated; use the `Tridion.AudienceManagent.API` namespace which provides the same functionality.

Outbound E-mail Script Extension

The Outbound E-mail Script Extension for the VBScript/JScript templating framework is deprecated; use the .NET-based Modular Templating framework which interacts with the TOM.NET API to implement Mailings.