MODULE 11: WEB SERVICES

Module Overview

Web services are a lightweight, industry-standard way to make application functionality available to many different external systems and users. Web services architecture enables programs that may be written in different languages or even on different platforms, to communicate with one another in a language- and platform-independent manner. Applications access Web services through widespread Web protocols and data formats such as HTTP, XML, SOAP, and OData. This eliminates concern about how each Web service is implemented.

Microsoft Dynamics NAV® 2013 supports the creation and publishing of Microsoft Dynamics NAV 2013 data and functionality as web services.

Objectives

- Describe Microsoft Dynamics NAV 2013 Web services architecture.
- Explain the protocols Microsoft Dynamics NAV 2013 uses for Web services.
- Evaluate the benefits of Web services over other integration options in Microsoft Dynamics NAV.
- Explain how to expose codeunit, page, and query objects as Web services.
- Consume Web services from external applications.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Prerequisite Knowledge

Microsoft Dynamics NAV 2013 supports creating and publishing Microsoft Dynamics NAV functionality as web services. You can expose pages, codeunits, or queries as web services, and even improve a page web service that uses an extension codeunit. When you publish Microsoft Dynamics NAV objects as web services, they are immediately available on the network.

Microsoft Dynamics NAV 2013 web services support the following protocols:

- SOAP (Simple Object Access Protocol) – Publishes pages for create, read, update, and delete access, and codeunits for function call access.
- OData (Open Data Protocol) – Publishes pages and queries for read access.

Web Services Architecture

Web services are a built-in feature of Microsoft Dynamics NAV 2013 Server. They are contained in the same executable file, and run as part of the same service process in Microsoft Windows. By default, when you install Microsoft Dynamics NAV 2013, it is configured to run both SOAP and OData web services. By using the Microsoft Dynamics NAV Server Administration Tool, you may turn either of the web services on or off.

Additional Reading: *For more information about Microsoft Dynamics NAV Server Administration Tool, refer to the Microsoft Dynamics NAV Server Administration Tool topic in the Microsoft Dynamics NAV 2013 Developer and IT Pro Help.*

Transactions and State

When you make a web service call, no server state is preserved between the calls. This means that no variables or single-instance codeunits are remembered and no session is maintained. Therefore, each separate web service call is always a single independent transaction.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Security

Every web service consumer must authenticate with Microsoft Dynamics NAV 2013. Web services uses all authentication types that are available in Microsoft Dynamics NAV 2013. You can also configure the web services to use Secure Sockets Layer (SSL) encryption. SSL is a cryptographic protocol that provides security and data integrity for data communications over a network. By encrypting the Microsoft Dynamics NAV web services that use SSL, you make your data and the network secure and reliable.

Publishing Web Services

Web services automatically publish objects from Microsoft Dynamics NAV 2013 as SOAP and OData web services. You publish the objects as web services on the **Web Services** page in the Microsoft Dynamics NAV 2013 client for Windows.

To publish an object as a web service, follow these steps:

1. Start Microsoft Dynamics NAV 2013 client for Windows.
2. In the **Search** field, enter "Web Services", and then click the **Web Services** page in the results.
3. Click **New**.

In the **New – Web Services** page, enter appropriate values in the **Object Type**, **Object ID**, **Service Name**, and **Published** fields.

Following definitions of fields that are displayed in the Web Services page.

Field	Remarks
Object Type	Specifies the type of the object to expose as a web service. You can select from Codeunit, Page, and Query options.
Object ID	Specifies the ID of the object to expose. You can enter the ID manually, or select it from the drop-down list.
Service Name	Specifies the name of the web service that is created for the specified object. The service name is visible to consumers of the web service and is used for identifying and distinguishing web services. For naming guidance, see the Web Services Best Practices topic in the Microsoft Dynamics NAV 2013 Developer and IT Pro Help.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Field	Remarks
Published	Specifies whether the web service is available to serve the requests. The service is available immediately when you select this check box.

When you publish a web service for an object, depending on the object type, it is available as either of the following:

- SOAP for codeunits and pages
- OData for pages and queries

SOAP Web Services

SOAP web services enable full flexibility for building operation-centric services. They provide industry standard interoperability. You can use SOAP to interact with page or codeunit web services in Microsoft Dynamics NAV 2013. SOAP Web services use the Web Service Description Language (WSDL) to describe their data structure and methods.

Web service users can discover published web services by pointing a browser or a tool, such as the Web Services Discovery Tool, at the computer that is running Microsoft Dynamics NAV Server and retrieve a list of available services. For SOAP web services, you typically enter a URI in a browser to view the discovery document, which lists published web services, or to view the WSDL document for a particular web service.

To display all published SOAP web services that are available at a Microsoft Dynamics NAV Server instance, use a URI of the following type.

```
http://<Server>:<Port>/<ServerInstance>/WS/<CompanyName>/services
```

Note: *Specifying the CompanyName parameter here is optional.*

For example, to display all published SOAP web services that are exposed by the CRONUS International Ltd. demo database, point the browser to the following URI.

```
http://localhost:7047/DynamicsNAV70/WS/services
```

To access the WSDL definition of a particular web service, you use the following syntax.

```
http://<Server>:<Port>/<ServerInstance>/WS/<CompanyName>/<ObjectType>/
<ServiceName>
```

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

The following table explains the components of the SOAP web service URI.

Component	Remarks
Server	The network address of the Microsoft Dynamics NAV 2013 Server. For example, localhost, nav.cronus.com, and 192.168.210.6 are all examples of valid network addresses.
Port	The port on which the SOAP web services are running. The default port is 7047. However, you can configure a different value.
ServerInstance	The name of the Microsoft Dynamics NAV 2013 Server instance. You may run multiple instances of Microsoft Dynamics NAV 2013 on the same server. The default value is DynamicsNAV70. However, you can configure a different value.
CompanyName	The name of the company that you want to access. This part of the URI is case-sensitive. Therefore, you must make sure that you enter it exactly as the company is named in Microsoft Dynamics NAV 2013. Otherwise, the server is unable to locate the web service. When you access the WSDL document, the CompanyName part is optional. When it is consuming the web service, it is mandatory.
ObjectType	The type of the object. Possible values are Page and Codeunit.
ServiceName	The name of the service, as configured in the **Web Services** page in Microsoft Dynamics NAV 2013. The name is arbitrary, but every web service of the same object type must have a unique name.

The following example displays the WSDL description for the Customer service.

```
http://localhost:7047/DynamicsNAV70/WS/Page/Customer
```

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

System Service

You can use the SystemService service in a SOAP web service application to retrieve a list of companies that are available in a specific database. A company name is typically part of the URI when you consume a Microsoft Dynamics NAV 2013 web service. The system service lets you retrieve names of available companies.

The SystemService service is available at the following URI.

```
http://<Server>:<Port>/<ServerInstance>/WS/SystemService
```

For example, for a default Microsoft Dynamics NAV 2013 demo installation, the SystemService service is available at the following URI.

```
http://localhost:7047/DynamicsNAV70/WS/SystemService
```

Page Web Services

When you expose a page as a SOAP web service, you expose a default set of operations that you can use to manage common operations such as Create, Read, Update, and Delete. Page-based web services offer built-in optimistic concurrency management. Each operation call in a page-based web service is managed as a single transaction.

Codeunit Web Services

When you expose a codeunit as a web service, all functions that are defined in the codeunit are exposed as operations. This lets you run business logic that is contained in C/AL code from external applications. For example, external applications can create new sales orders through a page web service, and also can call the release or post functions.

Following are several limitations of codeunit web services:

- Only global functions are visible as methods. You cannot call local functions directly from web services.
- Each exposed method may only have parameters of a simple type that map directly to XSD data types. You cannot use complex data types, such as record, page, or codeunit. The only exception is the XMLport data type, which is translated to a complex XSD definition. The XMLport data type enables client applications to map complex Microsoft Dynamics NAV 2013 data structures to object hierarchies.
- Single-instance and ordinary codeunits are the same. A single-instance codeunit is instantiated one time for each web service call.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Page Operations with SOAP Web Services

When you publish a page as a SOAP web service, it has a set of default operations that are exposed to consumers of the web service.

These operations match the corresponding actions that a user can perform by interacting with a page in the RoleTailored client. All page and table triggers that run when a user performs the same action from the RoleTailored client run when you call a web service operation.

The following table lists the operations and provides links to reference pages.

Operation	Description
Create	Creates a single record in the underlying table of a page.
CreateMultiple	Creates multiple records of the same type.
Delete	Deletes a single record.
Delete_<part>	Deletes a single record from a subpage on a page. For example, deletes a line in the **Lines** subpage of the **Sales Order** page.
GetRecIdFromKey	Converts a key, which uniquely represents a record in the database when it interacts with SOAP web services, into a string that matches the Record ID format that is used internally by Microsoft Dynamics NAV 2013.
IsUpdated	Verifies whether another user has updated a record after it was last read from the database.
Read	Reads a single record.
ReadByRecId	Reads a record by Record ID.
ReadMultiple	Reads a filtered set of records. This method lets you page through records, specify the page size and the key of the last record read.
Update	Updates a single record.
UpdateMultiple	Updates multiple records of the same type.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

All basic page operations are atomic. This means that either all relevant records are affected or no records are affected, even if there was a faulty condition in only one record.

***Note:** Only those methods that match the operations a user can do in the RoleTailored client are actually present on each page web service. For example, if page has the ModifyAllowed property set to **No**, then the Update and UpdateMultiple operations are not available in the web service.*

Extension Codeunits

For SOAP services, you can also use extension codeunits to extend the default set of operations that are available on a page. Adding an extension codeunit to a page is useful if you want to perform operations other than the standard Create, Read, Update, and Delete operations. The benefit of adding an extension codeunit to a page is that you can make the web service complete by adding operations that are logical to that service. Those operations can use the same object identification principle as the basic page operation.

When extending a page web service by using an extension codeunit, you must create a new web service record that has the same name as the name of the page web service that you are extending. Do not select the Published check box.

When you configure an extension codeunit for a page web service, add only those functions as operations to the web service that have the first parameter set to the Record data type, and the subtype equal to the source table of the page.

OData Web Services

The OData standard is good for web service applications that require a uniform, flexible, general purpose interface for exposing retrieve operations on a tabular data model to clients. OData supports Representational State Transfer (REST) based data services, which enable resources that are identified by using Uniform Resource Identifiers (URIs). They are defined in an abstract data model (EDM), and published within corporate networks and across the Internet by using simple Hypertext Transfer Protocol (HTTP) messages. OData services are lightweight with functionality that is frequently referenced directly in the URI.

Microsoft .NET Framework 4.0 was improved with WCF Data Services, which implement all the non-NAV specific parts of the OData stack. Visual Studio service references understand OData services and can generate EDM-based proxies. This enables the developer to use LINQ to write the data access logic.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

OData is supported in PowerPivot, a data-analysis add-in to Microsoft Excel 2010 that provides improved Business Intelligence capabilities. PowerPivot supports sharing and collaboration on user-generated Business Intelligence solutions in a Microsoft SharePoint Server 2010 environment.

Additional Reading: *For more information about PowerPivot, see http://www.powerpivot.com/.*

In addition to the AtomPub format, the OData implementation in Microsoft Dynamics NAV 2013 also supports the JSON format. This is a somewhat less verbose format that may perform better in low-bandwidth environments.

Whereas SOAP web services expose a WSDL document, OData web services expose an EDMX document that contains metadata for all published web services. The Entity Data Model (EDM) is a specification for defining the data that is used by applications that are built on the Entity Framework. EDMX is an XML-based file format that is the packaging format for the service metadata of a data service.

Obtaining the Service Metadata Document

To obtain the service metadata (EDMX) document for the OData web services that are published by a Microsoft Dynamics NAV 2013 server instance, use the following URI.

```
http://<Server>:<Port>/<ServerInstance>/OData/$metadata
```

The following table explains the components of the OData service metadata URI.

Component	Remarks
Server	The same as that used with SOAP web services.
Port	The port on which the OData web services run. The default port is 7048. However, you can configure a different value.
ServerInstance	The same as that used with SOAP web services.

Note: *By default, the Company entity type is always present in the EDMX document, and Microsoft Dynamics NAV 2013 always publishes it.*

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Obtaining AtomPub Documents and Feeds

When you publish a page or query web service, you expose an OData service that can be accessed from a uniform resource identifier (URI) by using a web browser or any other HTTP client. OData clients can use Atom Publishing Protocol (AtomPub) documents to interact with Microsoft Dynamics NAV data. AtomPub documents and feeds are XML.

To obtain a list of companies that use an AtomPub document, use the following URI.

```
http://localhost:7048/DynamicsNAV70/OData/Company
```

Note: *You must modify your Internet Explorer settings to display the actual XML for a feed instead of the feed content that has changed. In Internet Explorer, click Tools* **>** *Internet Options. On the Content tab, in the Feeds and Web Slices group, click Settings, and then click to clear the Turn on feed reading view check box. Restart Internet Explorer to enable the new setting.*

Additional Reading: *To learn more about AtomPub URI syntax and possibilities, refer to the How to: Use OData to Return/Obtain an AtomPub document topic in the Microsoft Dynamics NAV 2013 Developer and IT Pro Help.*

Handling UI Interaction

In Microsoft Dynamics NAV 2013, web services are stateless. Each call must either fully succeed or fail. Microsoft Dynamics NAV 2013 cannot interact with a web services user in the same manner that the RoleTailored client can. For example, if the C/AL code in a page opens a confirmation dialog box, a user in the RoleTailored client can respond to that confirmation. However, the web service call fails.

When you publish a web service, you must make sure that the code that you are publish does not assume the ability to interact with a user through the UI. You can use the **GUIALLOWED** function to suppress the UI. For example, you can use this function to determine whether a codeunit is called from the RoleTailored client or from a web service. You must make sure that you suppress any user interaction, such as CONFIRM, STRMENU, or the running of pages or reports, when a codeunit is called from a web service client.

When you implement a conditional code check in Microsoft Dynamics NAV, you should implement the check only around code that could cause an error. You should not encapsulate the whole business logic.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Variables of the Dialog data type or any of the functions that are listed as dialog functions can cause callback not allowed exceptions when they are called from a web service application. The **MESSAGE** function is the only function in this category that does not cause an exception, and it is suppressed.

The following are other keywords that you should not use:

- FORM.RUN
- FORM.RUNMODAL
- ACTIVATE
- REPORT.RUN
- REPORT.RUNMODAL
- HYPERLINK
- FILE.UPLOAD
- FILE.DOWNLOAD

Also, if you access any client-side Automation or .NET Framework interoperability objects from the C/AL code during a web-services call, a run-time error occurs.

Registration Web Service

CRONUS International Ltd. wants to automate the seminar registration process as much as it can. It plans to deploy a website that lets web users see the list of available seminars and register for scheduled seminars.

Solution Design

The functional requirements document for the CRONUS International Ltd. implementation includes the following requirements:

- External applications must be able to retrieve the list of scheduled seminars. The information that is available to the external applications must include at least the following:
  - Name of the seminar
  - Starting date of the seminar
  - Duration of the seminar
  - Status of the seminar
  - Maximum participants
  - Number of registered participants

- External applications must be unable to delete or modify information about scheduled seminars.
- External applications must be able to register new users for a seminar if the maximum number of participants has not been reached.

Your obvious choice is to use the SOAP web services and expose the necessary functionality.

Retrieving List of Scheduled Seminars

External applications must be able to retrieve the list of scheduled seminars. Reading data is best achieved through a page web service, where you can use the ReadMultiple method to retrieve multiple records from a page. Because the data must only be read, your simplest choice is to expose the **Seminar Registration List** page. This page already includes most of the information that is specified in the requirement. The only information that CRONUS requires, and that the page does not expose, is the number of registered participants. You can solve this by adding a new flow field to the table, and to the page.

You should try to expose only necessary information to external applications. Make sure that no sensitive data is available to anyone who does not absolutely require such data. Because the Seminar Registration List page includes more information than is specified in the requirement, consider creating a completely new page specifically for integrating with web services.

Note: *It is a good practice to always create a new page specifically for the web services integration, unless you explicitly want to expose the identical functionality of the page itself. The most important reason to do this is data sensitivity and to avoid exposing more information than needed. Another important reason is to avoid regression issues. If you delete or rename fields, or modify page or field properties, such as Editable, you may easily cause all external applications to stop working. By providing a separate page for web services, you completely avoid these issues.*

Note: *In the labs in this chapter you will not create a new page, but will reuse the existing* ***Seminar Registration List*** *page.*

For CRONUS, you decided that the risk of exposing the existing page is low. Because no sensitive data will be made available, you decide not to develop a new page. Instead, you decide to expose the **Seminar Registration List** page as a web service.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Preventing Users from Modifying or Deleting Records

Another functional requirement requires you to prevent external users from modifying or deleting information about scheduled seminars. Because the **Seminar Registration List** page is noneditable, you do not have to take any steps specifically to support this requirement.

Registering through Web Services

When registering students for seminars, seminar managers create new Seminar Registration Line records through the **Seminar Registration** page. Therefore, your first design choice might be to use the **Seminar Registration** page, and use its **Lines** subpage to create new Seminar Registration Line records. However, this design choice is not a good one, because it violates the previous requirement. If you expose the Seminar Registration page as a web service, you automatically enable external applications to perform any operations that seminar managers can do by using the user interface.

Following are several other alternatives to provide registration functionality:

- Expose the subpage directly. This is not a good choice because you have to add the **Document No.** and **Line No.** fields to the **Seminar Registration Subform** page. In standard Microsoft Dynamics NAV 2013, these fields are never exposed in the document pages because they easily could enable users to cause data-related errors or inconsistencies.
- Provide a separate page specifically for web services use. Although it is a slightly better choice, it opens the door to a poorly written (or malicious) external application. This could cause the same data-related errors or inconsistencies for users as the first option.
- Provide an extension codeunit that performs the task through C/AL logic. We recommend this option as the cleanest one. Therefore, you will provide a new codeunit, and expose this codeunit as an extension codeunit for the web service that publishes the **Seminar Registration List** page.

The codeunit must contain a function that receives the Customer No. and Contact No., and inserts a new Seminar Registration Line record. The function must make sure not to enable the registration of more participants in a seminar than is specified in the **Maximum Participants** field.

Solution Development

To meet the requirements that are defined in the functional requirements document, you must create and customize several objects.

Tables

You must customize the following table.

Table	Changes
123456710 Seminar Registration Header	Add a new flow field that is named Registered Participants, and configure it to show the count of registration lines.

Pages

You must customize the following page.

Page	Changes
123456710 Seminar Registration	Add the **Registered Participants** field.

Codeunits

You must create the following codeunit.

Codeunit	Remarks
123456710 Seminar Web Services Ext.	Contains a function that registers a participant for a seminar. This codeunit will be used as an extension codeunit for the **Seminar Registration List** page.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Lab 11.1: Creating a Web Service

Scenario

You are a developer working on a project to implement Microsoft Dynamics NAV 2013 for CRONUS International Ltd. Your goal is to develop the necessary changes to enable external applications to perform select functions of the Seminar Management module. After you develop changes, you must publish the web services and verify that they were correctly exposed.

Exercise 1: Customize the Objects

Exercise Scenario

Customize the tables and pages to support CRONUS International Ltd. requirements.

Task 1: Customize the Table

High Level Steps

1. Add a flow field to the table 123456710, **Seminar Registration Header** that shows the count of corresponding Seminar Registration Line records.

Detailed Steps

1. Add a flow field to the table 123456710, **Seminar Registration Header** that shows the count of corresponding Seminar Registration Line records.
   a. Design table 123456710, **Seminar Registration Header**.
   b. Add the field 61, **Registered Participants** of type Integer.
   c. Set the Editable property to **No**.
   d. Set the FieldClass property to FlowField.
   e. Set the CalcFormula property so that it applies the Count method to the **Seminar Registration Line** table, and filters the table to only include those records with the same Document No. as the **No.** field of the header.

The **CalcFormula** field should contain the following expression.

```
Count("Seminar Registration Line" WHERE (Document No.=FIELD(No.)))
```

   f. Compile, save, and then close the table.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Task 2: Customize the Page

High Level Steps

1. Add the **Registered Participants** field to the **Seminar Registration List** page.

Detailed Steps

1. Add the **Registered Participants** field to the **Seminar Registration List** page.
   a. Design the page 123456713, **Seminar Registration List**.
   b. Under the repeater control, add a new field control for the **Registered Participants** field.
   c. Compile, save, and then close the page.

Exercise 2: Configure and Test the Web Service

Exercise Scenario

After you customize the table and the page, you are now ready to configure the web service for the **Seminar Registration List** page.

Task 1: Publishing Seminar Registration List

High Level Steps

1. Publish page 123456710, **Seminar Registration** as a web service of name ScheduledSeminar.

Detailed Steps

1. Publish page 123456710, **Seminar Registration** as a web service of name ScheduledSeminar.
   a. Start Microsoft Dynamics NAV 2013 client for Windows.
   b. In the **Search** field, enter "Web Services".
   c. Click the **Web Services** page in the results.
   d. Click **New**.

In the **New – Web Services** page, enter the following information.

Field	Value
Object Type	Page
Object ID	123456710
Service Name	ScheduledSeminar
Published	(Selected)

   e. Close the **New – Web Services** page.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Task 2: Test the Web Service

High Level Steps

1. Use Internet Explorer to verify that the ScheduledSeminar SOAP web service is available in the DynamicsNAV70 instance on the localhost computer.

Detailed Steps

1. Use Internet Explorer to verify that the ScheduledSeminar SOAP web service is available in the DynamicsNAV70 instance on the localhost computer.
   a. Start Internet Explorer.
   b. Browse to the following URL.

```
http://localhost:7047/DynamicsNAV70/WS/CRONUS%20International%20Ltd/Pag
e/ScheduledSeminar
```

   c. Verify that you receive a valid XML response.

Exercise 3: Extend the ScheduledSeminar Web Service with an Extension Codeunit

Exercise Scenario

Your next task is to create a new codeunit with a function that creates a new **Seminar Registration Line** record for the specified Seminar Registration Header, customer, and contact. Then you must expose the codeunit as the extension codeunit for the ScheduledSeminar web service.

Task 1: Create the Codeunit

High Level Steps

1. Create the codeunit 123456710, Seminar Web Services Ext.
2. Add the **RegisterParticipant** function that creates a new **Seminar Registration Line** for the specified Seminar Registration Header, customer and contact.

Detailed Steps

1. Create the codeunit 123456710, Seminar Web Services Ext.
   a. Create a new codeunit.
   b. Save the codeunit as 123456710, Seminar Web Service Ext.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

2. Add the **RegisterParticipant** function that creates a new **Seminar Registration Line** for the specified Seminar Registration Header, customer and contact.
   a. Create a new global function and name it "RegisterParticipant".
   b. Define the following parameters for the function.

Name	DataType	Subtype	Length
SemRegHeader	Record	Seminar Registration Header	
CustNo	Code		20
ContNo	Code		20

   c. In the **RegisterParticipant** function, create the following local variables.

Name	DataType	Subtype
SemRegLine	Record	Seminar Registration Line
LineNo	Integer	

   d. Define a global text constant Text001, with the following value: "You cannot register for this seminar. Maximum number of participants has already been registered."
   e. In the function trigger for the **RegisterParticipant** function, enter the following C/AL code.

```
LineNo := 10000;

SemRegLine.LOCKTABLE;

SemRegLine.SETRANGE("Document No.",SemRegHeader."No.");

IF SemRegLine.FINDLAST THEN

  LineNo := LineNo + SemRegLine."Line No.";

SemRegHeader.CALCFIELDS("Registered Participants");

IF SemRegHeader."Registered Participants" >= SemRegHeader."Maximum
Participants" THEN

  ERROR(Text001);

SemRegLine.RESET;

SemRegLine.INIT;

SemRegLine.VALIDATE("Document No.",SemRegHeader."No.");
```

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

```
SemRegLine."Line No." := LineNo;

SemRegLine.VALIDATE("Bill-to Customer No.",CustNo);

SemRegLine.VALIDATE("Participant Contact No.",ContNo);

SemRegLine."Registration Date" := TODAY;

SemRegLine.INSERT(TRUE);
```

f. Compile, save, and then close the codeunit.

Task 2: Configure the Extension Codeunit

High Level Steps

1. Configure the Seminar Web Service Ext. codeunit as the extension codeunit for the ScheduledSeminar web service.

Detailed Steps

1. Configure the Seminar Web Service Ext. codeunit as the extension codeunit for the ScheduledSeminar web service.
   a. In the **Web Services** page in the Microsoft Dynamics NAV 2013 client for Windows, click **New**.
   b. Enter the following information for the new web service.

Property	Value
Object Type	Codeunit
Object ID	123456710
Service Name	ScheduledSeminar
Published	(not selected)

   c. Close the **New** – **Web Service** page.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Task 3: Verify the Web Service

High Level Steps

1. Use Internet Explorer to verify that the ScheduledSeminar web service now includes the RegisterParticipant method.

Detailed Steps

1. Use Internet Explorer to verify that the ScheduledSeminar web service now includes the RegisterParticipant method.
    a. Start Internet Explorer.
    b. Browse to the following URL.

```
http://localhost:7047/DynamicsNAV70/WS/CRONUS%20International%20Ltd/Pag
e/ScheduledSeminar
```

    c. Verify that the following node exists.

```
<message name="RegisterParticipant">
```

    d. Close Internet Explorer.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Lab 11.2: Create a Windows Forms Application to Test the Web Service

Scenario

You must create a simple application to test the newly published web services. You will create a new C# Windows Forms Application by using Microsoft Visual Studio. Connect it to the ScheduledSeminar web service. Enable it to display the list of scheduled seminars, and let users create new registrations.

Objectives

Now that Web service functions are published (exposed), the next step is to create an application to use (consume) that published service in an external application.

Exercise 1: Create a new Windows Forms Application

Exercise Scenario

Create a new Windows Forms Application and connect it to the ScheduledSeminar web service to display the list of scheduled seminars. Let users register for seminars. At the end of the registration process, display a message that states whether the registration succeeded or failed.

Task 1: Create the New Application

High Level Steps

1. In Microsoft Visual Studio, create a new C# Windows Forms Application.
2. Add a data grid control to the Form1 form.
3. Add the web service reference for the ScheduledSeminar web service.
4. When the application starts, bind the ReadMultiple results of the ScheduledSeminar function to the dataGridView1 grid control.
5. Add two text box controls and a button to the form.
6. Add the code to run the RegisterParticipant method on the ScheduledSeminar web service when a user clicks **Register**.
7. Test the web service.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Detailed Steps

1. In Microsoft Visual Studio, create a new C# Windows Forms Application.
   a. Start Microsoft Visual Studio.
   b. Click **File > New > Project**.
   c. Under Visual C#, select Windows.
   d. In the list of project types, select Windows Forms Application.
   e. In the **Name** field, enter "TestSchedSemWS".
   f. Click **OK**.

2. Add a data grid control to the Form1 form.
   a. In the **Toolbox**, under the Data group, select the DataGridView control.
   b. Drag the DataGridView control to the Form1 form.

3. Add the web service reference for the ScheduledSeminar web service.
   a. In **Solution Explorer**, right-click **References**, and then click Add Service Reference.
   b. In the **Add Service Reference** window, click **Advanced**.
   c. In the **Service Reference Settings** window, click **Add Web Reference**.
   d. In the **Add Web Reference** window, in the **URL** field, enter the following: http://localhost:7047/DynamicsNAV70/WS/CRONUS%20International%20Ltd/Page/ScheduledSeminar,
   e. In the **Web reference name** field, enter "ScheduledSeminar".
   f. Click **Add Reference**.

4. When the application starts, bind the ReadMultiple results of the ScheduledSeminar function to the dataGridView1 grid control.
   a. Double-click **Form1** to open the Form1.cs editor.
   b. At the end of the **using block**, enter the following C# code.

```
using TestSchedSemWS.ScheduledSeminar;
```

   c. In the **Form1_Load** function, enter the following C# code.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

```
ScheduledSeminar_Service svc =

    new ScheduledSeminar_Service

    {

        UseDefaultCredentials = true

    };

dataGridView1.DataSource =

    svc.ReadMultiple(

        new ScheduledSeminar_Filter[] { },

        null,

        0);
```

   d. Press CTRL+SHIFT+S to save all changes.

5. Add two text box controls and a button to the form.
   a. From the **Toolbox**, drag the TextBox control to the Form1 form two times.
   b. From the **Toolbox**, drag the Button control to the Form1 form.
   c. Set the Text property on the button1 control to "Register".

6. Add the code to run the RegisterParticipant method on the ScheduledSeminar web service when a user clicks **Register**.
   a. Double-click the button1 control.
   b. In the **button1_Click** function, enter the following C# code.

```
if (dataGridView1.SelectedRows.Count > 0)

{

    ScheduledSeminar_Service svc =

        new ScheduledSeminar_Service

        {

            UseDefaultCredentials = true
```

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

```
        };

    try

    {

        svc.RegisterParticipant(

            dataGridView1.SelectedRows[0].Cells["Key"].Value.ToString(),

            textBox1.Text,

            textBox2.Text);

        MessageBox.Show("Registration completed.");

    }

    catch (Exception ex)

    {

        MessageBox.Show(

            "Web service call has failed , and you receive the following error
message:\n" +

            ex.Message);

    }

}

else

{

    MessageBox.Show("Please, select the scheduled seminar first.");

}
```

c. Press CTRL+SHIFT+S to save.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

7. Test the web service.
   a. Press F5 to run the application.
   b. Verify that the application displays the list of available Seminar Registrations.
   c. Enter “10000” and “CT100140” into the two text fields.
   d. Click **Register**.
   e. In the Microsoft Dynamics NAV 2013 client for Windows, verify that the participant was registered for the seminar.

Note: *These steps only work if you already created seminar registrations. If the list is empty, then you must use the Seminar Registration page to create a seminar registration. Follow the steps in Appendix to create test records.*

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Module Review

Module Review and Takeaways

Web services are an industry-standard concept that enables applications to interact regardless of their platform, programming language, or technology. Microsoft Dynamics NAV 2013 supports SOAP and OData web services, and lets you publish pages, codeunits, and queries for consumption from SOAP or OData clients.

When you expose a page as a web service, you can access it through SOAP to create, read, update, and delete records in the page. This guarantees that all business logic that is contained in page and table triggers execute in the same manner as if a user performed the operation in the RoleTailored client.

You can expose codeunits as web services and call their global methods. You can also use codeunits to extend the functionality of page web services by using new methods, such as registering participants for scheduled seminars.

Finally, you can use pages and queries as sources for the OData web services, and then use the OData web services from an OData client. One of the best available options for OData web services consumption is the PowerPivot add-on for Microsoft Office Excel.

Test Your Knowledge

Test your knowledge with the following questions.

1. Which two types of web services can you publish from Microsoft Dynamics NAV 2013?

2. Which types of objects can you publish as OData web services?

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

3. What happens when the code execution in web services encounters a CONFIRM function?

4. Which of the following functions is not a valid operation for a page SOAP web service?

   ( ) CreateMultiple

   ( ) DeleteMultiple

   ( ) UpdateMultiple

   ( ) ReadMultiple

5. What are extension codeunits?

6. How do you properly expose an extension codeunit?

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the page you want to extend. Set Service Name field to the name of the codeunit. Leave the Published field unselected.

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the codeunit. Set Service Name field to the name of the page web service you want to extend. Leave the Published field unselected.

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the page you want to extend. Set Service Name field to the name of the codeunit. Select the Published field.

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the codeunit. Set Service Name field to the name of the page web service you want to extend. Select the Published field.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Test Your Knowledge Solutions

Module Review and Takeaways

1. Which two types of web services can you publish from Microsoft Dynamics NAV 2013?

   MODEL ANSWER:

   SOAP and OData

2. Which types of objects can you publish as OData web services?

   MODEL ANSWER:

   Pages and Queries

3. What happens when the code execution in web services encounters a CONFIRM function?

   MODEL ANSWER:

   A run time error occurs.

4. Which of the following functions is not a valid operation for a page SOAP web service?

   ( ) CreateMultiple

   (√) DeleteMultiple

   ( ) UpdateMultiple

   ( ) ReadMultiple

5. What are extension codeunits?

   MODEL ANSWER:

   Extension codeunits allow adding operations to page SOAP web services.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

6. How do you properly expose an extension codeunit?

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the page you want to extend. Set Service Name field to the name of the codeunit. Leave the Published field unselected.

   (√) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the codeunit. Set Service Name field to the name of the page web service you want to extend. Leave the Published field unselected.

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the page you want to extend. Set Service Name field to the name of the codeunit. Select the Published field.

   ( ) Create a Web Service entry for a codeunit. Set Object ID field to the ID of the codeunit. Set Service Name field to the name of the page web service you want to extend. Select the Published field.

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement

Microsoft Official Training Materials for Microsoft Dynamics ®
Your use of this content is subject to your current services agreement