Copyright 2006 by Insight Direct, Inc.

All rights reserved. No part of this book may be reproduced or copied in any form or by any means—
graphic, electronic, or mechanic—without the written permission of Insight Direct, Inc.
Table of Contents

Overview ........................................................................................................................................................ 1
   Target Audience ...................................................................................................................................... 1
Development System Requirements ............................................................................................................. 1
System Components ....................................................................................................................................... 2
The ServiceCEO API Web Service ................................................................................................................ 3
   Entities..................................................................................................................................................... 3
   Operations.................................................................................................................................................. 3
   Entity and Operations Table ...................................................................................................................... 3
Getting Started................................................................................................................................................. 4
   Installing ServiceCEO 6.1 ......................................................................................................................... 4
   Installing and Configuring the ServiceCEO API Web Service .................................................................. 5
      The Web.Config File ............................................................................................................................. 5
   Installing the Samples ............................................................................................................................... 6
   Quick Sample Review................................................................................................................................. 6
   Build Your First Application ....................................................................................................................... 7
      Creating a Visual Studio project with a Web Reference to the ServiceCEO API Web Service ............. 7
      Writing the Code.................................................................................................................................... 9
ServiceCEO API Web Service Concepts ....................................................................................................... 11
   “Classic” vs. “API”.................................................................................................................................. 11
   DataItem Base Class ................................................................................................................................ 11
      Top-Level DataItems, Owned DataItems ............................................................................................... 12
   References ................................................................................................................................................ 12
   Login........................................................................................................................................................... 12
   Querying the ServiceCEO API Web Service ............................................................................................ 12
   New, Save, Modify and Delete Operations............................................................................................... 14
      Setting Up and Executing an Operation .............................................................................................. 14
      Operation Results .................................................................................................................................. 15
   Working With DbKey................................................................................................................................. 17
      Comparisons ....................................................................................................................................... 17
      Multi-Keys .......................................................................................................................................... 17
   Jobs............................................................................................................................................................ 17
   Schedules .................................................................................................................................................. 17
   Estimates .................................................................................................................................................... 17
   Job Status .................................................................................................................................................. 18
   Telephone Types ...................................................................................................................................... 18
Overview

This document is designed to provide you with everything you’ll need to start using the ServiceCEO Application Programming Interface (API) web service. It contains the following sections:

- Introducing the ServiceCEO API Web Service
- Quick Start
- Sample Code Walkthrough

Target Audience

This document is intended for a developer who is familiar with ServiceCEO, and has some experience with Microsoft® Visual Basic.NET or C#. Experience consuming web services is preferred but not required.

Development System Requirements

To either develop new applications that use the API, or to run the sample applications (that are included as part of the ServiceCEO API package), you need to install the following items:


  Note: Unless otherwise stated, all instructions in this document are for the full version of Visual Studio 2005. However, most menus and options should have similarly-named counterparts in Express versions.

- A copy of your database. When developing applications and performing sample executions against the API, use a copy of your data that is not being used to run your live business. Do NOT execute any sample or application against your live database until you have thoroughly tested it on an off-line copy.

- ServiceCEO version 6.1+. Although the ServiceCEO client is not required on your development machine, it will come in handy for checking the results of commands issued via the web service.
  - The API can only access databases which have been upgraded to version 6.1 or newer. If your production environment is on an earlier version, you will still need to install 6.1 on your development machine in order to upgrade the database.

- IIS version 5 and above.

  Note: IIS must be installed before either the .NET Framework 2.0 or Visual Studio.
System Components

The following three components are required for proper use of the ServiceCEO API Web Service:

1. **Web Service.** An ASP.NET 2.0 web service being hosted by either the Internet Information Server (IIS) or, for debugging purposes, inside Visual Studio 2005 or Visual Web Developer.

2. **Database Server.** SQL Server hosting the ServiceCEO 6.1 database. This is the same database engine which provides data to the ServiceCEO 6.1 client application; no additional modifications are required for use with the API.

3. **Application.** Your application or a sample application

For development and sample purposes, these three components will probably reside on a single computer. All instructions in this document assume this configuration unless otherwise noted.

For production environments, these components may be distributed to multiple computers. See the Data Security Considerations section for important details and security considerations.
The ServiceCEO API Web Service

The ServiceCEO API Web Service provides programmatic access to your ServiceCEO data using the language and platform of your choice. The current version of the API allows you to access most aspects of ServiceCEO’s Customers, Jobs, Estimates and Tasks.

The ServiceCEO API Web Service can be used to perform many actions, including:

- Integrating ServiceCEO with your .NET application.
- Delivering customer and job data to your users’ desktops.
- Creating intranet web applications that allow access to ServiceCEO data by authorized users from anywhere in the world.
- Building middleware to integrate ServiceCEO with sales force automation systems (such as Salesforce.com, PeopleSoft and Siebel).
- Sending customer and job information to accounting packages.
- Developing middleware to integrate ServiceCEO with Microsoft Office®.

Entities

The ServiceCEO API Web Service interacts with your applications with objects. An object is a programmatic entity that holds a particular chunk of your ServiceCEO data. Each object has a number of properties. These properties handle the fields for that chunk of data.

Your application will use the object properties to interact with the ServiceCEO database through the API.

For details about these entities, please see the table below.

Operations

An operation acts on an input object and returns an output object containing any changes.

Using the API, you can construct web service client applications that use standard Web service protocols to programmatically:

Coming Soon!

Entity and Operations Table

Coming Soon!
Getting Started

This section covers how to get the ServiceCEO API Web Service up and running in the shortest possible time. Please perform any development on a machine specifically delegated for the API development. We suggest that you follow the steps in this order:

1. Setup your development environment:
   A. Install IIS version 5.0 or greater, if not already installed
   B. Install the development tools of choice. Typically, this is Visual Studio Express or Microsoft Visual Studio 2005

2. Setup ServiceCEO applications and the ServiceCEO API Web Service
   A. Install 6.1 on a stand-alone development machine
   B. Install and configure the ServiceCEO API Web Service, including the web.config file
   C. Install the samples
   D. Build your first application
      a. Creating a Visual Studio project with a web reference to the ServiceCEO API Web Service
      b. Writing the Code

Installing ServiceCEO 6.1

1. Select the computer you will use to develop, debug and test your application that uses ServiceCEO API Web Service. This computer should be a stand-alone, development machine.
2. Install the ServiceCEO 6.1 client and database server using either the blank or demo database. See the 6.1 release notes for details.
3. Run the ServiceCEO client on the development computer to ensure it is functioning properly.
   Optional:
4. Create a backup of your main ServiceCEO production database.
5. Restore the backup on the development machine.

Caution:
NEVER connect ServiceCEO on the API development machine to your main production server during development, debugging and testing phases.

You’re now ready to install and setup the ServiceCEO API Web Service!
Installing and Configuring the ServiceCEO API Web Service

Caution:
IIS must be installed before either .NET Framework 2.0 or VB.NET Express. If the order is reversed, a number of error messages such as “Failed to access IIS metabase” may appear. To fix this error, you need to go to Add/Remove programs, select .NET Framework 2.0, click the “Change/Remove” button and select “Repair”. Follow all prompts.

The ServiceCEO API Web Service acts as an intermediary between your application and ServiceCEO’s data and business logic. Follow these steps to install and configure the ServiceCEO API Web Service:

1. Download and extract 6.1.XXX-ServiceCEO_API_Setup.zip. This compressed file contains the following separate files:
   - API Samples.exe. This file installs sample projects for VB.NET, C#, and ASP.Net as well as the help file.
   - ServiceCEOAPI.exe. This file installs the ServiceCEO API Web Service.
   - Documentation.

   Note:
The web service files must be extracted from the compressed file using a file compression tool.

2. Double-click the ServiceCEOAPI.exe file.
3. Click Next. The license agreement screen appears.
4. Select the Accept option button.
5. Click Next. The Select Installation screen appears.
6. Click Next. The Confirm Installation screen appears.
7. Click Next. The installation starts. When the process is completed, a dialogue box appears, informing you of the successful outcome.

The Web.Config File

By default the ServiceCEO Web Service will connect to:

   Server Name: (local)/ServiceCEO
   Database Name: ServiceCEO

If this Server Name and Database Name is not correct for your development environment, you will need to alter the web service’s web.config file:

1. Navigate to the virtual directory where the ServiceCEO API Web Service was installed. This directory is C:\Inetpub\wwwroot\InsightDirect.ServiceCEO.Services.
2. Edit the web.config file in this directory. You can use Visual Studio, or just about any text editor (Notepad or Wordpad, for example).
3. To point the web service to the Server Name and Database Name of your choosing, you need to edit the Connection Strings settings. Find this section in the web.config file:
Getting Started

<connectionStrings>
  <add name="NA1" connectionString="Database=ServiceCEO; Server=(local)\ServiceCEO; UID=insight; PWD=SECRET;" providerName="System.Data.SqlClient" />
</connectionStrings>

4. You will only be editing the portions after “Database=” and “Server=”. Two options are available to you (choose one):

   a. Enter the name of your server and database in the appropriate areas:

      <add name="NA1" connectionString="Database=[ENTER DB NAME]; Server=[ENTER SERVER NAME or IP]; UID=insight; PWD=SECRET;" providerName="System.Data.SqlClient" />

   b. Configure the Web Service to refer to the current DSN settings used by your local ServiceCEO client to connect to the same server and database:

      <add name="NA1" connectionString="Database=UseDSNDB; Server=UseDSNServer; UID=insight; PWD=SECRET;" providerName="System.Data.SqlClient" />

Caution:
NEVER connect ServiceCEO on the API development machine to your main production server during development, debugging and testing phases. Using the section option and changing your client’s database connection to your main production server will cause the web service to be connected to that computer!

5. Save and close the web.config file.

Installing the Samples

A number of sample Visual Studio 2005 projects are located in the 6.1.XXX-ServiceCEO_API_Setup.zip file. These samples demonstrate how to consume the ServiceCEO Web Service.

1. Double-click on API Samples.exe. The installer wizard appears.

2. Click Next. The Select Installation Folder screen enables you to change the location where the ServiceCEO API Samples will be installed.

3. Click Next. The Samples are installed.

4. When complete, click Close to exit the wizard.

After installation, select Start > All Programs > Insight Direct > ServiceCEOAPI. A number of sample Visual Studio projects are located here for your review.

Quick Sample Review

The Customer Sample is a web application written in VB.Net and already includes a web reference to your local ServiceCEO API Web Service. It demonstrates how to retrieve customers based on search criteria as well as how to add a new customer to your database.

1. Select Start > All Programs > Insight Direct > ServiceCEOAPI > Customers Sample. Visual Studio opens.
Getting Started

2. Using the solution explorer, you can open any of the files in the web project. Most of the relevant code for this sample will be in these files:
   - CustomerWeb.aspx.vb
   - frmAddCustomer.aspx.vb
   - CEOHelper.vb

3. To run the sample, click Run or press F5.

The Jobs Sample demonstrates how to retrieve jobs based on the Job Start date and Job Status. It also shows how to save changes to jobs.

The SalesForce Integration Sample demonstrates how the ServiceCEO API can interact with other APIs by connecting to SalesForce (a popular CRM package; see https://www.salesforce.com). Once connected to SalesForce, the API can import all the customers from a SalesForce Demo account into the ServiceCEO database.

**Build Your First Application**

To build an application, you need to:

- Create a Visual Studio Project with a Web Reference to the ServiceCEO API Web Service
- Write the Code for the Project

**Creating a Visual Studio project with a Web Reference to the ServiceCEO API Web Service**

Add a web reference to the ServiceCEO API Web Service by following these steps:

1. Open Visual Studio and create a new C# Windows application. (Note: you can use Visual Basic to do this, but the code sample below is in C#.)

2. On the Project menu, select Add Web Reference.

3. Click on the “Web services on the local machine” link.
4. Click on the ServiceCEOWebService link.

5. In the web reference name box, type ServiceCEOAPI. This is the namespace you will use for the web reference.
6. Click **Add Reference**. The web reference is added to Web References section of your Solution Explorer.

## Writing the Code

This section describes how to write the code for your application.

Before you begin:

Create a new Windows Forms project in Visual Studio 2005. Using the default form provided to you by Visual Studio (typically named Form1), add a button called **Button1** and a DataGridView named **grdCustomers**. Double-click **Button1** to get to the button’s click event:

```csharp
protected void Button1_Click(object sender, EventArgs e)
```

Here’s a VB Sample:

```
' Step #1 Create web service object
Dim ws As CEO.ServiceCEOWebService = New CEO.ServiceCEOWebService()

' Step #2 Pass in authentication to web service. First you need to enter the user name and pw in a soap header
ws.WebServiceSoapHeaderValue = New CEO.WebServiceSoapHeader()
ws.WebServiceSoapHeaderValue.Username = "administrator"
ws.WebServiceSoapHeaderValue.Password = "administrator"

' Step #3 Create a Query Object that returns just corporate customers
Dim custquery As CEO.CustomerQuery = New CEO.CustomerQuery()
custquery.IsCorporate = True

' Step #4 Call the webservice and pass in the query
Dim results() As CEO.DataItem = ws.Retrieve(custquery)

' Step #5 Add columns to the grid
```
Getting Started

```csharp
grdCustomers.Columns.Add("name","Name")
grdCustomers.Columns.Add("primaryContact", "Primary Contact")

' Step #6 Add a new row for each customer in the list
Dim cust As CEO.Customer
For Each cust In results
    grdCustomers.Rows.Add(cust.AccountName,
                           cust.Locations(0).Contacts(0).Name.First + " " +
                           cust.Locations(0).Contacts(0).Name.Last )
Next

Below is a C# code sample with comments inserted for each step:

// Step #1 Create web service object
CEO.ServiceCEOWebService ws = new CEO.ServiceCEOWebService();

// Step #2 Pass in authentication to web service. First you need to enter the user name and pw in a soap header
ws.WebServiceSoapHeaderValue = new CEO.WebServiceSoapHeader();
ws.WebServiceSoapHeaderValue.Username = "administrator";
ws.WebServiceSoapHeaderValue.Password = "administrator";

// Step #3 Create a Query Object that returns just corporate customers
CEO.CustomerQuery custquery = new CEO.CustomerQuery();
custquery.IsCorporate = true;

// Step #4 Call the webservice and pass in the query
CEO.DataItem[] results = ws.Retrieve( custquery );

// Step #5 Add columns to the grid
grdCustomers.Columns.Add("name", "Name");
grdCustomers.Columns.Add( "primaryContact", "Primary Contact" );

// Step #6 Add a new row for each customer in the list
foreach ( CEO.Customer cust in results )
{
    grdCustomers.Rows.Add( cust.AccountName,
                            cust.Locations[0].Contacts[0].Name.First + " " +
                            cust.Locations[0].Contacts[0].Name.Last );
}
```

Once the code is completed, you can run it by pressing F5. Click Button1, and in a few seconds (or minutes depending on the size of the database) a list of all the corporate customers in your database will appear in the grid. Congratulations - you have just written your first ServiceCEO API application!
ServiceCEO API Web Service Concepts

**Note:**
This section expects that you have a basic familiarity with Object Oriented Programming.

**“Classic” vs. “API”**

The web service you are consuming is generated from a different, more modern codebase than the current ServiceCEO application. In addition to its current use as an API, this new codebase will power future generations of ServiceCEO products. To that end, while working on the new code we’ve taken the opportunity to remodel certain aspects of the application.

For example, in the current ServiceCEO application, henceforth referred to as “Classic”, some jobs are projects (jobs with multiple visits) and some jobs are not (one time jobs). In the API, and thus in future products, we’ve elected to remodel this so that all jobs have the potential to have multiple visits. In order to allow the API to interoperate with the Classic product, which still needs to differentiate, the API Job object has a property called ClassicJobType which determines whether it is a project or a one-time job.

**Note:**
Throughout the remaining documentation and code samples you will see “Classic” referring to the current generation ServiceCEO product. Any associated interoperability issues will be identified in a similar manner.

**DataItem Base Class**

A DataItem is an abstract base class representing any record in the ServiceCEO database, such as jobs, customers, tasks or users. Almost all ServiceCEO entities that you will touch derive from DataItem, and inherit some simple properties like:

- **ID** – A unique identifier for this record. Uniqueness is only guaranteed across a single concrete entity type. For example, there is only one Customer with ID “1”, but there may also be an Employee with ID “1”.

  See the section “Working with DbKey” for additional information on IDs.

- **RecTimestamp** – A byte array which is changed by SQL Server every time the record is saved. This can be used while synchronizing records, to detect whether the data has changed since it was last retrieved.

- **CreatedBy*, DateCreated*, LastEditedBy*, DateLastEdited* – The date, time and user who created or last modified the record.

  As of this release, the Classic client does not support these fields on all object types, therefore the values will not always be updated if a user changes a record using the client application.
Top-Level DataItems, Owned DataItems

All entities belong to one of two categories:

1. Top-Level DataItems
   a. Examples: Customer, Employee, Job
   b. Top Level Data Items represent the unit of work for modifying data, and usually for retrieving data as well. For example, a CustomerLocation cannot be directly saved to ServiceCEO using the API. Instead, it must be placed within the Customer.Locations list, and the entire Customer must be saved.

2. Owned DataItems
   a. Examples: CustomerLocation, ScheduledService
   b. Owned DataItems can generally only exist inside other DataItems.
   c. When an Owned DataItem’s part record is deleted, all Owned DataItems are also deleted. For example, if you delete a Customer, all CustomerLocations are implicitly deleted as well.

References

The term “reference” is used to describe two objects which are related to one another, but not in an ownership scenario, where one has direct control over the other. For example, a CustomerLocation is said to “reference” a Zone.

This relationship could be exposed in the API as a simple property, CustomerLocation.ZoneID, leaving API consumers to query for additional information about the Zone, such as its name, on their own. In many cases, we’ve attempted to increase API usability beyond this simple property, by using a complex ReferenceField(Of T) data type.

Thus, CustomerLocation.Zone in the API is declared as CEO.ReferenceFieldOfZone, which exposes the ZoneID, through CustomerLocation.Zone.ReferencedID, but also gives the user access to a short string describing the Zone, through CustomerLocation.Zone.FriendlyString.

Login

To access a ServiceCEO database, you must supply a valid user name and password. However, the API does not currently implement user rights, so, as long as a valid username and password are provided, users will be able to perform all functions, regardless of the actual user permissions that may have been defined in the ServiceCEO Security Options dialog box. In future releases of the API, security will be applied based upon the rights of the specific user logging in.

Querying the ServiceCEO API Web Service

The web service exposes only two methods. The first method, Retrieve, obtains entities from ServiceCEO. This method takes an instance of a Query object as a parameter. For instance, if we want to retrieve jobs we would pass in a JobQuery object or to retrieve Customer objects, we would supply a CustomerQuery.

Every Query class has some number of properties which are used to cooperatively filter the objects returned (supplying multiple filters will execute in a logical AND fashion). Filtering properties take on many different types: integer, string, Boolean, DBKey, and other object types. Remember, because of the nature of the web-service, object types will not be instantiated. You will need to create instances of these types but only if you wish to filter on them.

When you do not wish to include a filter property in the operation, you only need to set it to a null value: String.Empty for strings, DBKey.Empty for ID values and null for all other objects. If a primitive value
would result in an ambiguous filter (did I mean False, True or “don’t include”?) we have wrapped those in a Nullable<T> construct. A good rule of thumb is that unless you specifically set a filter property, you can assume that it will NOT be included in the operation.

**DateRangeSubQuery**

Potentially one of the most important querying objects you will need is the DateRangeSubQuery. You will use this any time you want to return objects of a time specific nature. All Query classes have at least two such filters: DateCreated* and DateLastEdited*.

*As of this release, not all Query classes will support querying on DateCreated and DateLastEdited.

DateRangeSubQuery objects use two Nullable<DateTime> properties to establish a window in time. These are the aptly named the GreaterThanEqual and LessThan properties. If a property is left null, then that portion of the equation is omitted. This allows filtering on a limited range (both limits supplied) or an unlimited range (either limit left null). If you leave both values null, you have effectively instructed the operation to ignore this filter entirely.

Remember you can never really discount the time. If you want “everything for today” you must set the time portions of to midnight of today (GreaterthanEqual) to midnight tomorrow (LessThan).

**Example**

Here is an example of how we might retrieve all Job objects scheduled for today.

1. Declare a variable specific to the object being retrieved:
   ```vbnet
   Dim query As New CEO.JobQuery
   ```

2. Set the ServiceDate filter to reflect that we are only interested in a certain window in time.
   ```vbnet
   query.ServiceDate = New DateSubQuery()
   DateTime dt = DateTime.Today.Date
   query.ServiceDate.GreaterThanEqual = dt
   query.ServiceDate.LessThan = dt + New TimeSpan( 24, 0, 0 );
   'set other filters as needed? CustomerID etc
   ```

3. Declare a variable for the web service proxy, leveraging the CEOHelper method to configure authentication:
   ```vbnet
   Using ws As CEO.ServiceCEOWebService =
   CEOHelper.GetProxy("administrator", "administrator")
   ```

4. Declare a variable to hold the results of the retrieve, which is an array of type DataItem:
   ```vbnet
   Dim results() As ServiceCEOAPI.DataItem
   ```

5. Pass the Query object into the retrieve method of the web service object and set it equal to the results object:
   ```vbnet
   results = ws.Retrieve(query)
   ```

The signature of the Retrieve web method specifies that it returns an array of DataItem objects. While iterating over the items in this array we can cast them to the specific type of object that we queried. For example, if we retrieved some jobs, cast them in this way:

```vbnet
For Each job As CEO.Job In results
    ' Do something here.
Next
```
When setting multiple properties on a query object, the API will query the database using “And” to join the properties (as opposed to or). For example, this job query instructs the API to retrieve every job for today AND has a status of Completed:

```csharp
query.ServiceDateStart = Date.Today
query.ServiceDateEnd = Date.Today
query.Status = CEO.JobWorkStatus.Completed
```

**New, Save, Modify and Delete Operations**

The second method exposed by the web service is DoOperation. Operations can be broken down into the following groups:

1. **New Operations** (example: CustomerNewOperation) which will:
   a. Create a new top-level DataItem in memory (without writing it to the database)
   b. Set default values based on business rules
   c. Set unique ID values for the object and any objects contained therein
   d. Return the object to the caller for further manipulation

2. **Save Operations** (example: CustomerSaveOperation) which will:
   a. Take as input a top-level DataItem object (example: Customer) which represents either a new record or an existing record in the database
   b. Write the input DataItem, as well as any owned DataItems (example: CustomerLocations), to the database
   c. Return a duplicate of the input object, containing the latest values written to the database, including the latest LastEdited timestamps, along with any ID values generated for new records

3. **Delete Operations** (example: CustomerDeleteOperation) which will delete records corresponding to the input DataItem, as well as any owned DataItems.

4. **Modify Operations** (example: JobAddEmployeeOperation) which:
   a. Take as input a top-level DataItem
   b. Modify the input DataItem in some way
   c. Return the resulting, modified, top-level DataItem

   Note that adding a new owned DataItem somewhere inside of a top-level DataItem is considered a Modify Operation, rather than a New Operation, since we are modifying the top-level item by adding a new item inside of it

Operations themselves are objects (which derive from OperationBase) which you instantiate and then configure using properties.

**Setting Up and Executing an Operation**

This section will walk you through setting up and executing an operation. This code sample below will configure and execute a JobSaveOperation:

1. Declare a variable specific to the object that is being saved:
   ```csharp
   Dim saveOperation As New CEO.JobSaveOperation()
   ```

2. Set the Input property of the save operation equal to the object you are saving (the job variable referenced here would have been obtained earlier, through a call to the Retrieve web method):
3. Get a web service proxy object:
   ```csharp
   Using ws As CEO.ServiceCEOWebService = CEOHelper.GetProxy("administrator", "administrator")
   ``

4. Declare a variable to hold the results of the operation, which is always of type OperationResult:
   ```csharp
   Dim saveResult As CEO.OperationResult
   ``

5. Pass the Operation object into the DoOperation method of the web service object and set it equal to the results object:
   ```csharp
   saveResult = ws.DoOperation(saveOperation)
   ```

**Operation Results**

The OperationResult class has a few important properties:

- **Success** (saveResult.Success) – A Boolean property. If true, the operation did what you asked it to do. If false, it did not. For performance reasons, many instances of operation failure do NOT throw an exception, and instead return Success as false. Examples of Success = false include:
  - Required fields missing.
  - You tried to edit a record which is not editable, like a completed Job.

In many cases, the code you write may want to abort processing if an unsuccessful OperationResult is encountered. If this is the case, instead of manually checking the Success property, you can call CEOHelper.AssertSuccess(), which will throw an Exception if a failure was encountered. This will ensure that your application’s flow stops immediately.
**ServiceCEO API Web Service Concepts**

- **ValidationIssues** *(saveResult.ValidationIssues)* Returns zero or more Prompt objects, each representing an issue that was encountered during the operation. Typically this array will be empty, unless Success was returned as false. To prevent the return of non-critical (informational) ValidationIssues, you can set the operation object’s LightValidation property to true before calling DoOperation.

The Prompt data type has properties that allow user interaction. For example, a Prompt could be returned while saving a job, with its Message property reading “There is a scheduling conflict for the team you’ve selected. Would you like to save anyway?” This message could be presented to the user.

Any New, Modify or Save Operation will return to the caller an updated copy of the object passed as input. This is a subtle detail of all web service development. For example, when you pass a Customer object into the web service in order to add a new CustomerLocation to it, your Customer object was serialized to XML by the .NET Framework. The XML is sent to the web server. The web server generates an XML response which includes XML describing the modified Customer object. The web server sends this XML back to us, where .NET creates a NEW Customer object from the XML. .NET does NOT update the original Customer object! Our first object, the one we sent in, hasn't changed; it still has only one location.

Since different types of operations will return different types of data (and some will return multiple types of data), you will need to cast the basic OperationResult class to a more specific data type in order to obtain specific results, such as the new Customer object in our example above.

In summary, it’s wise to do the following after EVERY DoOperation call:

2. As long as Success = True, cast your OperationResult to a specific type. For example, CustomerAddLocationOperation will return a OperationSingleResultOfCustomer. This class derives from OperationResult, but adds properties that enable the result to also contain a single Customer object.
3. Overwrite your old input variable with the result of the last web service call.

These steps look like this in VB.NET:

```vbnet
' Execute an operation using local variable ‘customer’ as input:
Dim newLocOp As New CEO.CustomerAddLocationOperation
newLocOp.Input = customer
newLocOp.PostalCode = "90210"
Dim results As CEO.OperationResult = ws.DoOperation(newLocOp)

' If it failed, something unexpected happened and we want to stop processing.
CEOHelper.AssertSuccess(results)

' Overwrite the local customer variable w/ the results.
' It’s safe to DirectCast here, because we know results.Success = True.
customer = DirectCast(results,
CEO.OperationSingleResultOfCustomer).ResultItem
```

Similarly, when calling a Save Operation, it’s important to grab the ResultItem property. For example, saving a Job will assign any new charges on that Job an ID that identifies the new record in the database. In addition, the LastEditedBy and DateLastEdited fields for all modified objects will be updated.
Working With DbKey

Every DataItem’s ID property is encapsulated inside an object called a DbKey. This is an implementation detail that will help us adapt a more modern database schema in the future. Unfortunately, as it is exposed through the ServiceCEO API Web Service, the DbKey object requires some special handling in your code. Specifically, you will always access the DbKey’s underlying value, which is a string, rather than the DbKey itself:

Example: `Customer.ID.Value = “123”`

Any new object which has not been saved to the database yet will be assigned a DataItem.ID.Value containing a Globally Unique Identifier (GUID). This value will be used until the object is saved to the database, at which point it will be replaced by the newly generated ID value from the database.

Example: `Customer.ID.Value = “{NEW:0d763a16-9f15-4eae-83b3-fa8ce1e65845}”`

Comparisons

Take special care when comparing two DbKeys. Two DbKey instances that have the same DbKey.Value will NOT be equal in code (e.g., "customer1.ID = customer2.ID") because, although the underlying values are the same, they are contained in two different DbKey objects. With this in mind, you should compare the underlying values rather then the IDs (e.g., “customer1.ID.Value = customer2.ID.Value”).

Multi-Keys

In certain instances, the API has modeled entities differently than they were in Classic so that a single logical item might be stored in multiple database tables. In these cases, a special DbKey.Value string may be assembled containing multiple values from the actual database tables, separated by periods. Example: “123.345.567”. Entities with these “multi-keys” will document the meaning of each section of the string under their section of the help file.

Jobs

Three types of jobs exist in Classic:

1. One-time jobs
2. Recurring jobs (a schedule; a job with a recurrence)
3. Projects (a job that requires multiple visits)

In the API, this has been remodeled. Every Job in the API contains one or more Visit objects. Thus, when you retrieve a one-time job in the API, it looks like a Classic project, even though it is not. Along these lines, when you save the job record using the API, it will be saved as a one-time job in Classic, even though it looks like a project. If you attempt certain edits to that one-time job that can’t be stored as a one-time job in Classic, the save operation will fail (a Prompt object will let you know what you did wrong). To see how a retrieved Job is stored in the Classic database, you can check its ClassicType property.

Schedules

In the current version of the ServiceCEO API Web Service, recurring jobs are not supported in the API. You can open a single occurrence of a recurring job, and you can edit and save it, but you can not create a new schedule, or edit an entire schedule at once. Similarly, you cannot open or create an Estimate for recurring service. These features will be available in a future version.

Estimates

In Classic, an estimate can be created for a one-time job, recurring job or a project. But what exactly is an estimate? An estimate in ServiceCEO represents a document given to a customer that can include expected
ServiceCEO API Web Service Concepts

charges (products, services, and kits) as well as expected service dates (visits). A common point of
confusion in Classic is that an estimate is an entirely separate entity when in fact it is simply a job
decorated with some extra information (date sent, estimate number etc). Since we have two different types
of “things which can be estimated” -- Jobs (both projects and non-projects, in Classic) and Schedules -- two
types of estimates exist in the API: a JobEstimate and a ScheduleEstimate. Since Schedules will be
available in a future release, the ScheduleEstimate will be available at that time as well. For now, API users
only have access to the JobEstimate class.

Job Status

Note that the way Job Status is handled in the API is a bit different then Classic. The JobWorkStatus
enumeration is used by both Jobs, Visits and Estimates. Supported statuses include:

- Active
- Cancelled
- Completed
- LostEstimate
- OpenEstimate
- WonEstimate
- Unknown

Please note that “Pending” status is no longer supported. Any jobs in Classic that have a status of pending
will have a JobWorkStatus of Active in the API. When retrieving records using the API, you can use the
“Active” JobWorkStatus to retrieve pending Jobs/Estimates.

Telephone Types

In Classic there are ten different phone types: Home1, Home2, Work1, Work2, Fax, Business Fax, Mobile,
Car, Other, Other2. These types have been simplified in the API and have been narrowed down to just five
types: Home, Work, Fax, Mobile and Unknown. Any phone number in Classic that has the type of Home1
or Home2 will have a type of Home in the API. Work1 and Work2 will be Work, Fax and Business Fax
will be Fax, Mobile will be Mobile and all others will be Unknown.

CEOHelper Class

The ServiceCEO API Web Service samples include a CEOHelper class, available in both VB.NET and C#,
which you can include in your own project.

For more information, see the ServiceCEO Model.chm file that is installed in the same directory as the API
samples and this document.

NewLines

In Windows systems, the new line consists of a carriage return character followed by a line feed character.
In Visual Basic 6, this is accessed by vbCrLf constant. In C#, the recommended way to represent a newline
is via Environment.NewLine constant, although a string literal “\n\r” can be used as well.

In Microsoft web services, if a multi-line string is returned through a web service, the \n\r is automatically
replaced by a single \n. This can negatively affect the display of text spanning multiple lines. To remedy
this problem, replace all instances of ‘\n’ in a string with Environment.NewLine before displaying them to
the user. We have provided a helper function (CEOHelper.EnsureCrLf) to assist you with this.
**Backward Compatibility**

The ServiceCEO API is an evolving product that will change over time with releases of new code. As new versions of the API are made available, the types and methods used in the web services may change. Until the API is stabilized, Insight Direct will **NOT** guarantee backward compatibility.

We highly recommend that you follow these best practices to minimize the impact of changes in the ServiceCEO API on your application:

1. Whenever possible, use the methods in the provided CEOHelper class.

   Do not modify the CEOHelper class. Since Insight Direct will provide a new version of it with each release, any changes you make to this file will be overwritten.

2. Avoid persisting multi-key ID values (those formatted like “123.345.567”) into your own data source since they may change with a future API release. If you must relate items in external storage back to ServiceCEO records, opt for something more durable like Visit.VisitNumber for a Visit, or add a post to the Insight Direct Discussion Board for a recommendation from Insight Direct developers.
Appendix A: Recommended Reading

- **Web Services Basics:**

- **Introducing Microsoft Visual Basic 2005 for Developers:**

- **How to Consume a Web Service Using Visual Basic or Visual C#:**
  Note that some of the step-by-step instructions are invalid for Visual Studio 2005, so review this page for general process information only.
Appendix B: Configuration Files

Microsoft.NET applications store their configuration information in special files. For web services and applications, these files are called web.config. For Windows applications, the file’s name consists of the executable file name followed by “.exe.config” (e.g., ServiceCEO.exe.config).

Each configuration file has important properties that can alter the behavior of the program or its database settings.

Web Service Configuration File

Follow these steps to access the web service configuration file. (Note that this is only one path; there are other ways to access this file that are not listed here.)

1. Select Start > Run. The Run dialog box appears.
2. Type “inetmgr”
3. Press Enter. The Internet Information Service window appears.
4. In the IIS window, navigate to: Web Sites > Default Web Site > (path where you installed ServiceCEOAPI web service). Select the folder web service folder in the left pane.
5. Right-click the web.config in the right pane and select Open from the pop-up menu. Visual Studio will open, displaying the contents of the web.config file.

Note:
Do not modify any settings in the web.config file except for the Database Configuration Parameters.

Database Configuration Parameters

The following settings control database configuration parameters:

```xml
<connectionStrings>
  <add name="NA1" connectionString="Database=UseDSNDB;Server=UseDSNServer;UID=insight;PWD=SECRET;"
       providerName="System.Data.SqlClient" />
</connectionStrings>
```

You can change the Server and Database settings to point to your database server. Other settings are currently not configurable.

The recommended method of accessing a database from the ServiceCEO API Web Service is via the DSN connection manager. Simply ensure that the configuration values for Database and Server are set to the keywords UseDSNDB and UseDSNServer, as shown in the example above.
Appendix C: Deployment Considerations

**Data Security Considerations**

The following best practices should be followed to ensure data security.

- If you are developing web pages to access ServiceCEO data, you should either:
  - Host the web service on a separate machine from the one servicing your web pages. This separate machine should be behind a firewall so it cannot be accessed via the Internet.
  - If you must host the web service on the same machine hosting your web pages, secure the web service folder in Internet Information Services (IIS) so it cannot be accessed by remote IP Addresses.

- If you are developing a Windows Forms (thick client) application to access ServiceCEO data, you should either:
  - Ensure that the application is run on your LAN only, so that unencrypted traffic between your application and the web service is not sent over the Internet. In this scenario, you should ensure that the machine hosting the web service is behind a firewall so it cannot be accessed via the Internet.
  - If you must run your application outside of your LAN, you should either:
    - Place the machine hosting the web service behind a firewall, and use a Virtual Private Network (VPN) to connect the remote machine to your local network.
    - Expose the machine hosting the web service to the Internet on the HTTPS port only, leveraging IIS encryption and a security certificate.

If you have any questions, please post them on the Discussion Boards.
Appendix D: Getting Help

Before contacting technical support, you need to know the version of the ServiceCEO API.

Please follow these steps in determining the version number of your API installation:

1. Select **Start > Run**. A Run dialog box appears.
2. Type **inetmgr** and press Enter. The IIS configuration screen appears.
3. Navigate to the web folder of ServiceCEO API (by default, InsightDirect.ServiceCEO.Services).
4. Expand to show the bin folder.
5. Right click on the bin folder and select **Explore** from the pop-up menu. The right side pane of inetmgr will display the files.
6. In the default view (“Tiles”), look for InsightDirect.ServiceCEO.dll. A version number is displayed underneath it. This is the version of the ServiceCEO API.

For more information or help with other issues, please post on the API forum in the Insight Direct Discussion Boards. For more information on accessing the Discussion Boards please visit the Member Center at [http://www.insightdirect.com/member-login/](http://www.insightdirect.com/member-login/)