Currency Server™

CLOANTO®

Currency Server 6.3

by Cloanto Corporation

Table of Contents


Chapter 1

1 Introducing Currency Server

Welcome to Currency Server

Thank you for choosing Currency Server, which is part of Cloanto's Currency System family of products and services. It is our mission to deliver not only the most advanced currency-enabling component on the market, but also the finest documentation and support. Should you ever have questions, comments or special requirements, please do not hesitate to let us know. We always appreciate your feedback.

Your Currency System Team

Getting Started

The following sections provide the most important information to get started with Currency Server and to quickly configure the software for first use:

- For information about system requirements and deployment, see System Requirements and Deployment Options.
- For step-by-step configuration instructions and advice, see Initial Configuration and Code Examples.
- For information about best practices, see Quality Checklist and Operational Procedures.
- For information on currency-related topics, see Working with Currencies and Recommended Reading.
- For information about the different interfaces which can be used to access and control Currency Server, see Accessing the Software.

Context Help

Context help is always available during use of Currency Server Manager by pausing the pointer over an option or feature (or by clicking the item for which you need instructions, and then pressing F1).

Examples and Other Information

To download examples of code which accesses the functionality of Currency Server, or if you need more specific information about technical, administrative or troubleshooting issues, please refer to the Currency Server Homepage (shortcut: currencyserver.com).

Related Topics

- For more information about new features of this version of Currency Server, see New Features.
- For more information about software, licensing, service and support options, see Options.

1.1 New Features

Currency Server includes the following new features.

Version 6

- Opens the door to new features while retaining compatibility with version 5 and previous programming interfaces
- Simplified Enterprise Edition licensing, no longer requiring optional modules or client or CPU core licenses (the Standard Edition remains supported, but is deprecated)
- New Currency Server Manager features (Changelog in **Software** tab, daily scheduling in **FX Feeds** tab, etc.)
- Support for new FX feeds and data formats
- Native 64 bit code and COM interface for x64 systems

- Improved **Post-Update Actions**, including Amazon S3 output
- Enhanced support for latest Windows Server and Windows 10/11 releases

Version 5

- New Standard and Enterprise editions. The Enterprise Edition includes all optional modules (Multiple FX Feeds, Unlimited Web Service Clients) and has no limit on CPU cores. Optional add-ons remain supported in the Standard Edition (which is now licensed for 4 CPU cores).
- New JavaScript output for lightweight and compatible Ajax-like web integration (convert-as-you-type fields, widgets, etc.)
- **Post-Update Actions** can now be defined directly in the **Clients** tab of Currency Server Manager (rather than via external scripts). The actions include exporting to file, uploading to server (FTP, WebDAV, Azure Blob, etc.), and executing a script or application.
- Increasing list of data export formats, including XML, CSV, TSV, INI, JavaScript, as well as emulation of third-party feed formats for customers who need to replace legacy feeds without rewriting their code
- Easier initial setup via Populate button (in the **Active Currencies** tab of Currency Server Manager)
- Support for new FX feeds and data formats
- Each currency may now be hard-assigned to a specific FX feed. This allows for more precise aggregation of data from multiple feeds.
- Added pervasive support for FX feed subscription expiration and renewal, if supplied by the data provider (commercial feeds only, does not apply to free feeds). Support for these new feed properties was added to the universal filter. Service expiration reminders were included among the administrative notifications.
- New monitoring and logging options are included in the **Notifications** tab of Currency Server Manager
- Currency World Monitor added as a supported feed provided as part of the Currency System platform. The (optional) feed is currently provided as a free service to all software customers. It provides non-exchange rate data updates (e.g. notification of new currencies, name changes, EMU status changes, etc.) for all currencies of the world.
- The COM and .NET interfaces have been both streamlined and extended, while preserving compatible interfaces for legacy code. New features include the ability to programmatically set FX feeds and their access mode, to export data in XML and other formats on the fly, and to access the properties of all currencies (opposed to active currencies only)
- New currency properties include Popularity (High, Medium, Low), localized names for the entity (official and short country or institution name), subunit, and plural names
- Enhanced support for Windows Server 2008 to 2019 and Windows Vista to Windows 10 versions

Version 4

- Built-in support for all currencies and locales of the world (with manual or automatic updates)
- Specific support for EMU currencies as their status changes (includes support for past and future euro currencies)
- Internationalization features (Unicode support, multi-language currency names, symbols, formats, etc.)
- Find most likely currency by internet TLD, country code, Windows LCID and numerical currency code
- Output formatted currency values, names and symbols as HTML code
- DCOM-compatible COM interface
- Managed .NET code, including SOAP interface for remote Web service clients on any platform
- New Currency Server Manager user interface (with quick access via notification area icon)
- New options to support multiple FX servers (for data aggregation, failover, cross-checks, etc.)
- New monitoring features (fluctuations, lack of updates, software responsiveness, etc.)
- New and improved notification functionality (SMTP, redesigned single-window on-screen

messages, etc.)

- Security sandbox
- Automatic post-update action (e.g. to export to XML, INI, CSV or SQL)
- Plug-in filters (with sample source code) offer virtually unlimited support for different data providers, protocols and formats
- New XPath-enabled universal filter (for easier support of new providers)
- Built-in support for new data formats and for additional providers of exchange rate data
- Advanced deployment options via Windows Installer (MSI) technology or by copying .NET assembly and Web service files
- Deployment profiles can be saved to file for one-step configuration (e.g. from staging to production system)
- Easier integration with existing Quality Management System and Euro Changeover procedures
- Enhanced support for Windows Server 2003, Windows XP, .NET Framework 1.1, Terminal Services, Office System 2003 and hyper-threading CPUs
- Simple activation of optional add-ons from the **Software** tab in Currency Server Manager

1.2 Options

Currency Server comes in a Standard Edition (with optional add-on modules) and in an Enterprise Edition (which includes all add-ons). Standard technical support is provided via web/email. Additional licensing, software and support options are available. Please feel free to contact us for more information.

Add-On Licenses

The following components are included in the Enterprise Edition, and may also be added and activated to the Standard Edition either individually or as an upgrade to the Enterprise Edition by entering the respective license key in the **Software** tab of Currency Server Manager:

- CPU Upgrade (support for servers with more than four CPU cores)
- Multiple FX Feeds (support for multiple FX servers, e.g. for data source redundancy and aggregation, to synchronize and monitor clusters, etc.)
- Unlimited Web Service Clients (unrestricted, or with support for custom Web service license keys via external validation code)

Special Licenses

Additional licensing options include:

- Volume, site and worldwide licenses
- ISP standard license (ISP acquires software license, full software functionality is free of charge to end users)
- ISP value license (free of charge to ISP, end users acquire Web service license)

Data Feeds

Currency Server includes built-in support for numerous independent providers of exchange rate data, both free and subscription-based. Currency System additionally provides the following data feed services:

- Daily exchange rate updates (selection of currencies, verified and monitored by Currency System)
- Weekly updates of static currency properties (all currencies of the world, including EMU changes, currency names, new currencies, etc.)

Additional Software

The following software components are available or can be created and/or customized:

- Custom filters (to connect to any provider of data of your choice, if not already supported)
- Custom software versions for data provider bundles (with locked support for a single data source and easy configuration during setup)
- Customized calculator clients (for corporate deployment, free distribution via website, etc.)

Support and Services

In addition to web/email support, the following services are available:

- Premium support (24x7 telephone/pager plus monitoring of your Currency Server notifications)
- FX feed filter maintenance and source code license
- Consulting (software deployment, data feeds, etc.)
- Active monitoring (your server is remotely monitored for lack of currency data updates, fluctuations, etc.)
- FTP updates (data is uploaded daily to your server)
- Currency Server Web service (i.e. service only, with software managed by Currency System)

Related Topics

- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information about FX feeds and automatic update options, see The FX Feeds Tab.
- For more information about creating custom FX feed filters, see FX Feed Filters.

Web Links

- For more information about available options, see Currency Server Options on the Currency System website.


Chapter 2

2 Getting Started

This section covers:

- Initial Configuration
- Code Examples
- Quality Checklist
- Operational Procedures

2.1 Initial Configuration

Currency Server needs a one-time configuration through Currency Server Manager (e.g. where to get the exchange rate updates from, how many currencies to support, who to notify in case of problems), after which it is ready for use by COM and Web service clients.

Important Information

If you decide to set up the software without going through the rest of this documentation you should keep in mind a few important technical considerations:

- Out of the box, Currency Server "knows" about all the currencies of the world (**All Currencies** tab of Currency Server Manager), i.e. their names, currency codes and other static properties (but not their exchange rates). Updates to this information can be applied both manually and automatically.
- The availability of dynamic information such as exchange rate data depends on the FX feed(s) set in the **FX Feeds** tab. To help you prioritize items, Currency Server Manager uses star symbols to show "currency rank" and "feed rank" values (three stars indicate the most popular and trusted currencies and feeds). The less popular a currency is on the foreign exchange market, the less likely it is to be supported by some FX feeds. This is related to the trading volume of the given currency and/or to the availability of official reference data (e.g. provided by a national central bank). If there is no such data, it is more difficult to provide a meaningful exchange rate, i.e. one that has any practical use. Currencies for which exchange rate data is available are listed in the **Active Currencies** tab. Although you can manually add currencies to this list, and then manually update their exchange rates every day (which you probably don't want to do, as this is what Currency Server is supposed to do), we recommend that you click **Populate** (or **Update Now**) to let the software automatically build a list of currencies which are supported by the selected FX feed(s), without adding unsupported currencies.
- Currency Server provides methods both for exchange rate information purposes (e.g. "What is the rate of the Japanese yen?") and for currency conversion purposes (e.g. "How much is $5 in euros?"). In theory, once you have the exchange rate of a currency relative to another, you could do your own multiplications or divisions to convert between these two. There are however two reasons why you may prefer to use the software's Convert() method to let Currency Server do the conversions for you: a procedure called triangulation is required to convert to and from EMU currencies which are a sub-unit of the euro, and there may be rounding issues involved where you need to know what the smallest unit for the destination currency is (e.g. it may not be practical to provide results with a precision of 0.01 when you convert to a currency which has no cents at all). Currency Server takes care of triangulation and rounding for you. Currency Server can also output formatted results as HTML, which is useful to support international currency names and symbols on websites.

Up and Running in 5 Minutes

To quickly configure Currency Server for first use:

1. Open **Currency Server Manager** (e.g. by double-clicking the Currency Server notification area icon).
2. Make sure that the **Connection** settings give the software access to the internet. By default, Currency Server uses the system network and proxy server settings.
3. In the **Active Currencies** tab, click **Populate**.
4. If you see any currencies that you do not need (from a usability and maintenance point of

view, more is not necessarily better), select them (using Ctrl or Shift to select multiple entries or a range of entries) and click **Remove** to delete the corresponding entries.

5. Set **Updates may add or remove currencies** to **No (lock all)**. If you don't select this option, the currencies that you have removed (plus possibly more currencies) will be added again at the next update; also, you will not receive any warning messages should a currency unexpectedly be removed from the list maintained by your FX feed(s).
6. In the **Notifications** tab, make sure that Currency Server is configured in such a way that you will become aware of any information, warning and error conditions (this includes connection problems, currencies ceasing to be legal tender, etc.) Use **Test Now** to verify that you actually receive the test message. You don't need to worry about the messages you may receive, as these will usually be self-explanatory and the software will keep running fine even in case of problems, however it is very important that you receive them.
7. In order for the **Monitoring** functionality to be meaningful from the very beginning, the initial set of exchange rate data should be double-checked either by loading the data from two different FX feeds and letting the software monitor the differences, or by manually comparing the data with another external foreign exchange reference source. If this is not done, it is theoretically possible that an incorrect rate provided by a FX feed is loaded, accepted (due to lack of previous reference information) and then retained. Also see the Quality Checklist.
8. To enable unassisted operation of Currency Server set the **Auto-Update** option in the **FX Feeds** tab, and decide how often you would like the exchange rate data to be refreshed by accessing the FX feed(s). A refresh interval between 1 and 4 hours is in general considered sufficient, since the official reference exchange rates used in commerce are usually published only once every day.
9. If your list of active currencies includes currencies for which rates are not provided or updated every day, you may adjust the **Inactivity Warning** options in the **Monitoring** tab accordingly. If the warning is enabled for **Each currency**, it is recommended to set its value to **97:20:00** (three days plus margin for a change in summer time and a small delay) or higher. Once you are comfortable with the timer set for **Each currency**, you can disable the **Warn immediately when partial data is received** option (you will still receive inactivity warnings based on the timer).
10. Currency Server is now configured and running. You can back up the entire configuration by using **SaveConfiguration.vbs** (from the **Actions** menu of the Currency Server notification area icon, or in the "Actions" subdirectory of the Currency Server installation directory). You may also want to check the Code Examples, Quality Checklist and Operational Procedures sections.

Resetting Previous Configuration Changes

If you need to reset any configuration changes which you may have applied:

1. Open **Currency Server Manager** (e.g. by double-clicking the Currency Server notification area icon).
2. **FX Feeds** tab: select a single server using the default Currency Server Feed by Currency System.
3. **Locales** tab: click **1033 English (United States)**, then **Set as default**, then **Edit...**, then **Reset All**.
4. **All Currencies** tab: click **Reset All**.
5. **Active Currencies** tab: click **Reset EMU**, then set **Updates may add or remove currencies** to **Yes (lock only EMU)**, **Conflicting EMU rates** to **Use internal rates without warning** and **Unrecognized currency codes** to **Ignore and warn**.
6. **Monitoring** tab: set the **Application Monitor** timer to no more than **01:00:00**, the two **Fluctuation** settings to no more than **15%**, the **Inactivity Warning** for the **Entire set** to **25:20:00** and for **Each currency** to **00:00:00**, and enable **Warn immediately when partial data is received**.
7. Check all Currency Server Manager tabs for other options you may want to change based on your needs.

Advanced Options

1. The default settings in the **FX Feeds** tab refer to data made available by Currency System. You can set a different data provider if you wish, but we recommend to leave this default setting unchanged to test the initial configuration. If you purchased the Multiple FX Feeds component you can add Currency System's secondary server (and set **Multiple servers** to **Failover**) or you can merge data from different sources (set **Multiple servers** to **Merge**).

2. If you need to configure more complex currency data update schedules than those made possible by the **Auto-Update** functionality (**FX Feeds** tab) you can disable Auto-Update and use Windows Task Scheduler instead, or you can invoke updates via the COM or Web service (.NET and SOAP) interface. Permission to access the functionality required to invoke data updates has to be granted in the **Administrative Functions** section of the **Clients** tab (e.g. **Grant access to COM clients** if you decide to use **UpdateNow.vbs**).
3. If you configured the software so that logging or other data is written to files (e.g. **Write to log** file option in the **Notifications** tab and external client validation and custom action code which may be launched as set in the **Clients** tab), make sure that such files are rotated, transferred or otherwise maintained in consideration of the available disk space or storage quota.
4. If you need to export or upload data in XML, CSV, INI or another format (e.g. for further processing by another application, for Currency System calculator clients, for logging and monitoring purposes, etc.) add a **Post-Update Action** in the **Clients** tab.
5. If so desired, you can write your own filters to import currency data, and/or create custom user authentication functionality to offer a subscription-based Web service based on Currency Server.

Managing Currency Server on Multiple Systems

If you are the administrator of several computers running Currency Server:

1. You can configure one system to monitor the other servers by setting **Multiple servers** to **Cross-Check**. Assuming that the clients wrote the currency data to a file at a known location (e.g. using a **Post-Update Action**), add the address of each file as if it were a different server, using the **Generic Fallback Filter**.
2. You can use configuration files (saved and loaded via COM, or created with **SaveConfiguration.vbs** and loaded with a double-click) to apply identical configurations from one system to other systems (setup details like software installation information and license keys are not contaminated by this procedure).
3. You can deploy a "master" instance of Currency Server for central administration of multiple "slaves" (e.g. on a web server farm). In this scenario, you could configure the master to fetch the rates as recommended in "Up and Running in 5 Minutes". The slaves would then have **Updates may add or remove** currencies set to **Yes (full refresh)** in the **Active Currencies** tab, and a single server in the **FX Feeds** tab using the **Generic Fallback** Filter and referencing a file written by the master (e.g. using a **Post-Update Action**). In this way the slaves will automatically adjust to whatever currencies are set in the master.

Managing Calculator Clients

Currency Server can be used to feed data to customized currency calculator clients, which are also available from Currency System. Please refer to Client-Server Configurations for additional information concerning configuration and operations.

Related Topics

- For examples of programmatic access to the Currency Server methods and properties, see Code Examples.
- For more information about system requirements and configuration, see System Requirements.
- For more information about software installation, see Deployment Options.
- For more information about services and files, see Services and Executable Files.
- For more information about best practices, see Quality Checklist and Operational Procedures.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.
- For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about the different interfaces which can be used to access and control Currency Server, see Accessing the Software.

2.2 Code Examples

Although in some cases you may only need to use Currency Server to write an XML, CSV or other file via a **Post-Update Action**, more in general you will probably want to write some scripts or other programs to access the COM and Web service (.NET and SOAP) functionality of the software.

Frequently Used Methods and Properties

Currency Server offers a variety of methods and properties which are available over the COM and Web service interfaces. The most frequently used ones are:

- ActiveCurrencies collection, which contains all active currency objects with their names, symbols and other properties
- Convert(), to convert a currency amount with optional rounding and formatting
- Rate(), to get the exchange (conversion) rate of a currency, which is equivalent to invoking Convert() for an amount of 1
- Name(), to get the extended name of a currency, using the desired language and capitalization style
- Symbol(), to get the symbol (not the official code) conventionally used for a currency in the desired language
- UpdateNow(), to force a connection to the FX feed(s) and load new currency data

Tips

When implementing your first currency-related code, keep in mind the following general recommendations:

- Strings are better than numerals to store results, because exchange rates and rounded currency amounts have trailing 0s (e.g. "$1.50"), with a precision which is different from currency to currency (some currencies use cents, some use other fractions, while other currencies only use round numbers, etc.) When you use methods such as Convert(), these automatically take care of the proper rounding, and trailing 0s are preserved if you specify the string output format. Microsoft's Currency data types which only support up to four decimals do not meet EMU requirements on precision and triangulation.
- Instead of using exchange rate values and performing conversions yourself with a simple multiplication or division, use methods such as Convert(), which take into consideration the appropriate triangulation and rounding requirements.
- Keep it simple: if you use Currency Server to currency-enable a website, you normally don't need to use multiple locales if your website is in English only. If you don't explicitly specify any language or formatting parameters, the default locale settings (Locales tab of Currency Server Manager) will be used.
- More is not always better: if you are tempted to support more and more currencies, consider whether you need each currency, the risk of overwhelming users, and the administration overhead.
- After you have chosen a list of currencies that you need to support and keep up-to-date, set **Updates may add or remove currencies** to **No (lock all)** in the **Active Currencies** tab of Currency Server Manager.
- Do not invoke administrative functions such as UpdateNow() in the same code where you also interact with users, e.g. to provide currency information and conversion functionality. Exchange rate data updates are relatively time consuming operations which are best performed by the software itself (e.g. via **Auto-Update** in the **FX Feeds tab**), via Task Scheduler, or with some separate code controlled by you.
- Invest some time in reading through the Currency Server documentation and website, which contain additional tips, and don't forget to let us know if you have additional questions, suggestions or requirements.

A Simple Conversion

The following examples show how to do a simple conversion (in this case 1000.00 US dollars to Japanese yen). The three main parameters we need to set are a source currency ("From"), a

destination currency ("To"), and the desired amount. The source and destination currencies are expressed using ISO currency codes (e.g. "USD", "EUR", "GBP", "JPY", etc.) You can get the codes from the **Active Currencies** tab of **Currency Server Manager**, or using the Code property of the Currency object.

Note: these first examples do not include error handling. Also, for simplicity, the MsgBox function is used, which may cut output after 1 KB of text.

Using Visual Basic/VBScript:

```
Dim application, conv, ver
Const curncsrvReturnRateString = 1
Set application = CreateObject("CurrencyServer.Application")
conv = application.Convert("USD", "JPY", 1000, True, "",
curncsrvReturnRateString)
ver = application.Version
MsgBox "USD 1000.00 = JPY " & conv
```

Using Microsoft Visual C++ native COM support:

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#import "C:\Program Files (x86)\Cloanto\Currency Server\CURNSRV5.DLL"
...
CurrencyServer::IApplicationPtr pApplication;
pApplication.CreateInstance(OLESTR("CurrencyServer.Application"));
_variant_t vConv = pApplication->Convert(OLESTR("USD"), OLESTR("JPY"), 1000,
VARIANT_TRUE, OLESTR(""), CurrencyServer::curncsrvReturnRateString);
_bstr_t bVer = pApplication->GetVersion();
_tprintf(_T("USD 1000.00 = JPY %s\r\n"), (const _TCHAR *)vConv);
```

Using C# (to add the reference to the Currency Server COM interface in Visual Studio, select Project/Reference... and select the Currency Server Type Library in the COM tab):

```
CurrencyServer.ApplicationClass application = new
CurrencyServer.ApplicationClass();
String result = (String)application.Convert("USD", "JPY", 1000.0, true, "",
CurrencyServer.curncsrvReturnRate.curncsrvReturnRateString, DBNull.Value,
DBNull.Value);
String version = (String)application.Version;
MessageBox.Show("USD 1000.00 = JPY " + result);
CurrencyServer.IActiveCurrencies activeCurrencies =
application.ActiveCurrencies;
String currencies = new String('\0', 0);
foreach (CurrencyServer.ICurrency curr in activeCurrencies)
{
currencies += curr.Code;
currencies += " - ";
currencies += curr.get_Name("", true);
currencies += "\r\n";
}
MessageBox.Show(currencies);
```

A Simple Information Table

The following VBScript code shows how to create a simple list of exchange rates for information purposes, first showing amounts of foreign currencies for 1 US dollar, and then US dollars for 1 foreign currency unit. If you need to do the same for different amounts (e.g. 100 or 1000 currency units) just replace Rate() with Convert() and set the desired Amount (e.g. 100 or 1000). Again, for simplicity we use the MsgBox function, which may cut output after 1 KB of text.

```
Dim application, currencies, curnc, curr_str
Const curncsrvReturnRateString = 1
Const curncsrvReturnRateNumber = 2

Set application = CreateObject("CurrencyServer.Application")
Set currencies = application.activecurrencies
curr_str = ""
for each curnc in currencies
        curr_str = curr_str + "USD 1.00 = " + curnc.Code + " " +
application.Rate("USD", curnc.Code, true, "", curncsrvReturnRateString) + " (" +
curnc.Name + ")" + vbLf
next
MsgBox curr_str, , CStr(currencies.Count) + " Currencies"
curr_str = ""
for each curnc in currencies
        curr_str = curr_str + curnc.Code + " 1.00 = USD " +
application.Rate(curnc.Code, "USD", true, "", curncsrvReturnRateString) + vbLf
next
MsgBox curr_str, , CStr(currencies.Count) + " Currencies"
```

A Simple SQL Interface

The following code could for example be invoked as a custom action module to feed exchange rate data into a database. Just set the base unit used by your system (e.g. replace "USD" with "EUR"), if necessary replace LOCALHOST with the remote SQL server name or address, set the DATABASENAME, USERNAME and PASSWORD fields (or set "Password=;" if no password is required) and complete the INSERT part with appropriate index columns and data (e.g. you may want to add a date/time stamp).

The "On Error Resume Next" statement is required to avoid error dialogs, which would be invisible yet blocking if running under a context that is not allowed to interact with the desktop. The return value is required in order for Currency Server to know whether the action succeeded or not, and to issue an appropriate notification action. The output of the Err.Source e Err.Description parts helps log meaningful details in case of an error.

```
On Error Resume Next

Main()
If Err.Number <> 0 Then
    If Err.Description <> "" Then
       WScript.Echo Err.Source & ": " & Err.Description
    End If
    If Err.Number > 0 And Err.Number < 4096 Then
        WScript.Quit -2146828288 + Err.Number   '0x800A0### VB error
    Else
        WScript.Quit Err.Number
    End If
End If
WScript.Quit 0

Sub Main()
   Dim lObjConn
   Dim lStrSQL
   Dim application
   Dim currencies
   Dim currency
   Dim basecurrency
   Dim rate
   Const curncsrvReturnRateString = 1
   Const curncsrvReturnRateNumber = 2
   basecurrency = "USD"
```

```
   Set application = CreateObject("CurrencyServer.Application")
   Set currencies = application.activecurrencies
   Set lObjConn = Server.CreateObject("ADODB.Connection")
   lObjConn.Open _
   "Provider=SQLOLEDB.1;" & _
   "Data Source=LOCALHOST;" & _
   "Initial Catalog=DATABASENAME;" & _
   "User ID=USERNAME;" & _
   "Password=PASSWORD;"
   For Each currency In currencies
          rate = application.Rate(basecurrency, currency.Code, False, "",
curncsrvReturnRateNumber)
          lStrSQL = 'INSERT INTO TABLENAME (COL1,COL2,...)
VALUES(currency.Code,rate,...)'
          lObjConn.Execute lStrSQL
   Next
   lObjConn.Close
   Set lObjConn = Nothing
   Set currencies = Nothing
   Set application = Nothing
End Sub
```

An ASP Conversion Form

The following ASP example ("csexample1.asp") shows a simple conversion form where the user can pick a source and destination currency, and enter an amount to be converted:

```
<%@ language="VBScript" %>
<html>
<head>
<title>Currency Server ASP Example 1</title>
</head>
<body>
<%
Const curncsrvReturnRateString = 1
Const curncsrvReturnRateNumber = 2
Dim objApplication, objCurrencies
Set objApplication = Server.CreateObject("CurrencyServer.Application")
Set objCurrencies = objApplication.ActiveCurrencies
%>
<form action=csexample1.asp method="POST">
<p>Convert from:</p>
<p><select name=CurrFrom style="HEIGHT: 22px; WIDTH: 166px">
<% For i = 1 To objCurrencies.Count
if Request.Form("CurrFrom") = objCurrencies.Item(i).Code then %>
        <option selected><%= objCurrencies.Item(i).Code%>
<% else %>
        <option><%= objCurrencies.Item(i).Code%>
<% end if%>
<% Next %>
</select></p>
<p>To:</p>
<p><select name=CurrTo style="HEIGHT: 22px; WIDTH: 166px">
<% For i = 1 To objCurrencies.Count
if Request.Form("CurrTo") = objCurrencies.Item(i).Code then %>
        <option selected><%= objCurrencies.Item(i).Code%>
<% else %>
        <option><%= objCurrencies.Item(i).Code%>
<% end if%>
```

```
<% Next %>
</select></p>
<p>Amount:</p>
<p><input name=FromValue value="<%=Request.Form("FromValue")%>" size="20"></p>
<p><input type="submit" name="Convert" value=Convert></p>
<% dim ToValue %>
<% dim FromValue %>
<% FromValue=Request.Form("FromValue") %>
<% if (FromValue > "") and (IsNumeric(FromValue)) then
        ToValue = objApplication.Convert(Request.Form("CurrFrom"),
Request.Form("CurrTo"), Request.Form("FromValue"), true, "",
curncsrvReturnRateString)
end if %>
<% if FromValue = "" then ToValue = "" end if %>
<% if not IsNumeric(FromValue) then
        FromValue = ""
        ToValue = ""
end if
%>
<p>Result:</p>
<p><input name=ToValue Value="<%=ToValue%>" size="20"></p>
<% Set objApplication = Nothing %>
</form>
</body>
</html>
```

Invoking an Update

The following VBScript example shows how an exchange rate data update can be invoked. Because the ActiveCurrencies collection object contains a snapshot of the currency list, which is not updated during the lifetime of the object, the old object is released and a new object is created after the update.

```
Dim application
Set application = CreateObject("CurrencyServer.Application")
Set currencies = application.ActiveCurrencies
' ... (omitted): access to the currencies collection
' the following forces an update of the exchange rate data
application.UpdateNow
' the following releases the 'currencies' object
Set currencies = nothing
' the following gets a new ActiveCurrencies object with
' the updated information
Set currencies = application.ActiveCurrencies
```

Related Topics

- For step-by-step configuration instructions and advice, see Initial Configuration.
- For more information about the different interfaces which can be used to access and control Currency Server, see Accessing the Software.
- For SOAP-specific examples, see The SOAP Web Service Interface.

Web Links

- For more detailed and practical examples in different programming languages, see Code Samples and Source Code on the Currency System website.

2.3 Quality Checklist

	Step	Reference
1	Once your configuration has been finalized, set **Updates may add or remove currencies** to **No (lock all)** in the **Active Currencies** tab to ensure that no new currencies are automatically added without your knowledge, and that a notification is issued if some currencies are not updated by a scheduled refresh.	Initial Configuration, The Active Currencies Tab
2	If exchange rate data is fetched from multiple FX feeds (merge mode), consider whether you prefer the exchange rate data for each currency to always be sourced from the same FX feed. To apply this to each active currency, in the **FX Events** tab of the **Edit Currency** dialog, set the **FX feed** option to a specific FX feed or to Any. Locking the data to a feed ensures that exchange rates are not fetched from different feeds over time if a feed is allowed to be skipped due to a temporary failure, or if the number of currencies covered by a feed changes.	The Edit Currency Dialog
3	Make sure that the application, fluctuation and inactivity monitoring and warning options in the **Monitoring** tab are active and reflect your needs, so that the appropriate notifications are issued if necessary.	The Monitoring Tab
4	Monitoring of excessive fluctuations and of lack of activity only works between updates, not when exchange rate data is first loaded. Unless you are loading the data from multiple providers by using the **Cross-check** option in the FX Feeds tab, you may want to manually double-check the accuracy of the data which is first loaded into the system. To do this you can either load the data (**Update Now** in the **Active Currencies** tab), then change the FX feed (in the **FX Feeds tab**), then load the data again (at which point monitoring takes place, comparing the rates provided by the different FX feeds), and then restore the original FX feed, or compare the rates listed in the **Active Currencies** tab with a known reference source (e.g. a financial newspaper).	The Active Currencies Tab, The FX Feeds Tab
5	In the **Information and Error Messages** group of the **Notifications** tab, make sure that one or more notification options are enabled. Use **Test Now** to confirm the proper delivery and handling of messages. Notifications include warnings (new data is accepted), errors (new data is discarded, while the software keeps using cached data) and information messages (e.g. availability of new currencies, changes in EMU status, etc.) It is very important that you make sure that if something fails during an exchange rate update somebody will know about it and act appropriately (even if the software will keep running and satisfying client requests using cached data).	The Notifications Tab, Services and Executable Files
6	Fine-tune the monitoring and notification options to take into account currencies that have less frequent updates and FX feeds that are not always responsive. Currency-specific inactivity tolerance options can be set in the **FX Events** tab of the **Edit Currency** dialog. In a scenario where data is merged from multiple FX feeds, feed-specific notification options can be set via the **On merge error** option in the **Edit FX Feed** dialog.	The Edit Currency Dialog, The Edit Feed Dialog
7	Especially on e-commerce sites, and with consideration to your specific public, judge whether offering a choice of certain less common currency units represent a useful service or if instead it just makes the list unnecessarily long and more difficult to use. If your currency data provider	Currency Popularity Charts (web link), The All Currencies Tab

	Step	Reference
	offers a long list of currencies (and maybe even non-currency units, such as the value of gold and silver) you may want to consider manually removing less-used ones. In the **All Currencies** tab, sort the list by the Active column, and consider the specific reasons for maintaining active currencies where the Rank is a single star (i.e. Rank 1, Low Popularity).	
8	In the **All Currencies** tab, sort the list by the Active column, and review any active currencies where the Legal Tender status indicates No. Do you need to use these currencies at all? Sometimes it is helpful to support a currency for a few months after it has been replaced by a newer one, however after a while it usually is no longer necessary, and the currency can be removed from the list of **Active Currencies**.	The All Currencies Tab, The Active Currencies Tab
9	If you created a post-update custom action module, test both the success and failure scenarios. The success case should return 0, while the failure case should return a meaningful HRESULT exit code, which should be detected and notified as an error by Currency Server. Neither case should open a user prompt (which would be blocking in an unattended scenario).	The Clients Tab, Custom Action Modules
10	If you defined your own currency unit names, is your use of capitalization and singular vs. plural forms consistent?	Money and Style (web link)
11	If there are payments involved, and the charge will be in a currency other than the one displayed and/or a currency other than the customer's credit card account, make sure that the customer is reminded of the fact that the card company and/or bank may add a few percentage points of commission (e.g. use wording such as "Approximate price in", "Approximate equivalent", "Your card will be charged in currency XYZ", or provide additional information in notes, etc.)	From Exchange Rates to Actual Charges (web link), Exchange Rate Information and Disclaimers (web link)
12	If you make your data available to the public, make sure that your legal notices include specific currency-related information, disclaimers and limitation of liability sections. You may want to have a look at the documents which are linked in the FAQ section on the Currency System website and the legal notices which are published by major financial, travel and information websites.	Exchange Rate Information and Disclaimers (web link)
13	Currency Server is designed to issue a notification when a currency joins the EMU, or is replaced by the euro. Usually such a message will include a detailed list of changes, and it will remind you that you can select **Reset All** (**All Currencies** tab) and **Reset EMU** (**Active Currencies** tab) in order to apply all changes in a single step. If your organization has a euro transition plan in place, share this Quality Checklist and the Operational Procedures with your euro changeover personnel. If you are the person responsible for the operation and maintenance of the software we recommend that you do not ignore messages by Currency Server informing you about a change in the EMU status of one or more currencies. Similar messages will let you know about the existence of new non-EMU currencies, or about changes in the official name of a currency. You may want to deal with these events in a way similar to the handling of EMU changes.	Automatic Updates, The EMU and the Euro, The Euro (web link)
14	If Currency Server is running on a stateless system (e.g. a "disposable" virtual machine) you may want to enable the **Update on startup** (in the **FX Feeds tab**) and **Auto-**	The All Currencies Tab, The FX Feeds Tab

	Step	Reference
	apply updates (**All Currencies** tab) options.	
15	Define software maintenance tasks and roles and/or share this Quality Checklist and the Operational Procedures with your quality system personnel. If you are the person responsible for the operation and maintenance of the software make sure that you are familiar with the checklist and with the procedures.	Operational Procedures

2.4 Operational Procedures

We recommend that you review the following roles and tasks, also for possible inclusion in the Quality Management System and Euro Changeover procedures which your organization may have. Most of the recurrent tasks and "what if" scenarios outlined here also apply if you are the only person responsible for the operation and maintenance of Currency Server.

Roles

The following roles define how Currency Server tasks can be assigned to individuals.

- Project Manager: defines responsibilities and coordinates the deployment and maintenance of the software; this is usually, but not necessarily, the main contact person for Currency System.
- Project Contacts: individuals who may contact Currency System for technical issues, and usually include the Project Manager; Currency System may also contact these individuals for important software or currency-related information (e.g. major EMU news, etc.)
- Event Screeners: individuals who receive notifications issued by the Currency Server software (typically the same engineers who get called or paged when a web server, firewall or database has a problem).
- Administrative Contact: the main contact person for administrative and billing matters.

One-Time Tasks

It is recommended that the following tasks be performed at least once when Currency Server is deployed.

- Configure the software as outlined in the Initial Configuration and Code Examples sections.
- Verify that your deployment meets the recommendations included in the Quality Checklist.
- Create a customized set of Currency Server Roles, Tasks and Event Management Procedures. E.g. complete the procedures outlined in this section by adding steps for logging events and related actions, documenting who has to be contacted when events occur, etc.
- Make sure that you have the appropriate Currency System support plan information and contacts on file, and that Currency System has your project and administrative contact information. Please refer to currencysystem.com/contact/ if you are not sure who to contact.

Daily Tasks

Currency Server was designed to automatically take care of daily tasks such as the collection of exchange rate data, database updates, creation of logs, etc. Once set up, these tasks should not require human intervention.

Monthly Tasks

The Initial Configuration and Quality Checklist sections describe how to set up the software so that important warning and error conditions are promptly notified. Nevertheless, a chain of unexpected events may lead to a condition where the software keeps running unattended in spite of a previous warning or error condition. The following manual tasks can be performed to add an additional layer of quality monitoring. The recommended frequency of these tasks is monthly, however you can change it as appropriate, possibly also merging this list with the yearly tasks.

- In the **Active Currencies** tab of Currency Server Manager, verify that no currency has a

warning flag (red exclamation point). For each warning flag, check the logs to see when the warning or error condition affecting that currency occurred, and why the warning flag was not cleared. Click **Reset Warnings** to clear all warning flags.

- In the **Active Currencies** tab of Currency Server Manager, click **Information** and check the date appearing at the top of the status information. This is the date of the most recent successful exchange rate data update. If this date is older than a few days (considering week-ends and bank holidays), check the logs to see whether an inactivity warning condition was raised by the software, and what type of action was taken (or not taken) to address this condition. If no warning notification was sent, double-check the recommendations outlined in the Quality Checklist. As a minimum, verify the **Inactivity Warning** settings in the **Monitoring** tab, and the **Notifications** tab.
- In the **Notifications** tab, click **Test Now** to issue a test notification. Verify that the message recipients respond as per your procedures.
- The **Write to log file** option (**Notifications** tab of Currency Server Manager) and the external client validation and custom action code which may be launched (**Clients** tab) write data to disk. Verify that the files are being maintained as per your policies (e.g. moved or deleted after a certain period, etc.), and that the available disk space or storage quota is sufficient for continued operation.

Yearly Tasks

Yearly tasks mostly involve possible contract and subscription renewals.

- If you are using Currency Server in combination with a premium support, yearly free upgrade or similar program, verify the renewal of the contract.
- If you are using Currency Server to collect data from a commercial provider of exchange rate data under a yearly subscription contract, verify the renewal of the contract.
- Verify whether Currency System has your up-to-date project and administrative contact information.

Event Management

You may want to have procedures in place to handle certain specific scenarios. The following are some of the events which are notified by the software, listed in order of frequency (based on monitoring performed by Currency System). In most cases the notification messages include detailed information to help you address the specific condition. On special occasions, e.g. to confirm unusual currency exchange rate fluctuations or to provide extended information about major EMU changes, Currency System may also issue an email advisory and/or add an entry to the product log.

- Test message: if you receive a message saying "This is a test message from Currency Server. A response may be required by your quality system verification procedures." respond as per your procedures.
- No information about one or more specific currencies: it is normal that some providers of exchange rate data do not provide daily updates for certain less common currencies, in which case either they provide the previous day's rate(s), or they do not provide the rate(s) at all, so that Currency Server issues a warning and keeps using cached data for the missing rate(s); if this condition persists for several days this could mean that the currency is not supported any more by the FX feed(s), and you may want to consider removing it from the list of **Active Currencies**.
- Connection timeout, host not found and other FX server connection errors: occasional failures caused by sporadic network or server problems can be considered normal, however if the updates fail too frequently you may want to double-check the system's internet connectivity and then consider changing the **Servers** and/or **Connection** settings; if connections to a specific server keep failing make sure that you are using the latest version of the FX feed filter and double-check the **Address** field in the settings of the server which is giving an error; as a last resort, consider using a different FX feed.
- FX feed filter errors: occasional server problems may result in HTML error message pages being delivered instead of a data file, which would be reported by the filter; if the updates from a specific FX feed keep failing make sure that you are using the latest version of the FX feed filter; as a last resort, consider using a different FX feed.
- Extended inactivity warning: during week-ends and bank holidays the data source(s) may not update the exchange rate data, which should be considered normal; this warning is also normal

as a side effect of other warning or error conditions, which may cause cached exchange rate values to be used instead of updated data; if you did not receive any other warning or error messages (e.g. in relation to a connection problem or to excessive fluctuations) you should consider whether the **Inactivity Warning** settings in the **Monitoring** tab are excessively severe; if exchange rate data updates are normally triggered via Task Scheduler or via the COM, .NET or SOAP interfaces, verify whether the process responsible for invoking the updates is running properly; as a last resort, consider using a different FX feed.

- Excessive exchange rate fluctuation: special circumstances may cause unusual fluctuations. For example, in January 2001 the Turkish lira fluctuated by more than 40% on a single day, and the Icelandic krona fell repeatedly in October 2008, losing about 25% of its value in a single day. A good thing to do when Currency Server issues such a notification is to search for the currency name and code on Google News or a similar service. Unless the condition is also confirmed by major news and financial sources you should consider whether the cause could be an error in the data, especially if the fluctuation affects multiple currencies. By default Currency Server treats excessive fluctuations as an error which requires manual approval of the data, and keeps running using cached values for all currencies (not just the currencies affected by the unusual fluctuation); you may want to consider whether the **Fluctuation** settings in the **Monitoring** tab are excessively severe
- Service or executable does not respond error: open the Services tool in the Control Panel or in Computer Management and check the status and properties of the Currency Server services; this event may be the result of a low memory condition; starting or restarting the Currency Server service(s) and/or rebooting the system usually solves the problem, however you may also want to try to understand the underlying issue (e.g. a possible memory leak).
- One or more currencies have joined the EMU: back up the entire configuration by using SaveConfiguration.vbs; open Currency Server Manager; to apply the updated currency data click Reset All (or edit the currencies manually and set their Regime property to EMU) in the All Currencies tab, then select Reset EMU in the Active Currencies tab to make sure that Currency Server starts using the appropriate procedures which apply to EMU currencies (constant conversion rates, triangulation, etc.) If you are feeding rates to calculator clients such as those created with Currency System's Calculator Builder, also refer to the procedures which apply to your specific configuration.
- One or more currencies have completed the euro transition phase: back up the entire configuration by using **SaveConfiguration.vbs**; open Currency Server Manager; to apply the updated currency data click **Reset All** (or edit the currencies manually and unselect their **Legal Tender** checkbox) in the **All Currencies** tab, then select **Reset EMU** in the **Active Currencies** tab to remove the currencies which have ceased to be legal tender (having been replaced by the euro); your organization may have specific Euro Changeover procedures specifying for how long after a currency has ceased to be legal tender it should be retained in data provided by systems such as Currency Server, in which case you may have to postpone the last step described here. If you are feeding rates to calculator clients such as those created with Currency System's Calculator Builder, also refer to the procedures which apply to your specific configuration.
- Unknown currency code: unknown currency codes which may be found in the exchange rate data are either ignored or added to the list or **All Currencies**, depending on the **Unrecognized currency codes** setting in the **Active Currencies** tab; we recommend to set this option to **Ignore and warn**, and to manually add new codes to the list or **All Currencies** after verification of their intended purpose; usually these are private or otherwise non-standard codes used by some commercial FX feeds, which sometimes include one or more fictitious or obsolete codes for tracking purposes (they act as a watermark if the data is republished); when new official codes are introduced the software issues a different notification.
- Other changes to currencies of the world (e.g. new currency introduced, change of name, currency ceases to be legal tender, etc.): back up the entire configuration by using **SaveConfiguration.vbs**; open Currency Server Manager; to apply the updated currency data click **Reset All** (or edit the currencies manually) in the **All Currencies** tab; answer **Yes** when asked whether to apply the changes to the active currencies.
- Other events: follow your procedures and the instructions in the warning or error message; contact Currency System Support or use the contact information provided with your premium support plan if you need assistance.

Related Topics

- For more information about automatic currency data updates and notifications, see Automatic

Updates.

- For more information about the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about software, licensing, service and support options, see Options.
- For more information about the administration of calculator clients, see Client-Server Configurations.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.
- For more information about related web pages and sites, see Web Resources.

Chapter 3

3 Working with Currencies

This section covers:

- Currencies of the World
- Currency Codes
- The EMU and the Euro
- Rates and Conversions
- Internationalization
- Automatic Updates
- Recommended Reading

3.1 Currencies of the World

Currency Server contains static data (i.e. information like currency names and symbols, which, unlike exchange rates, is not subject to daily updates) about all currencies of the world. This data can be edited in the **All Currencies** tab of Currency Server Manager, or it can be updated automatically.

General vs. Localized Data

Static data for each currency includes both general properties, which are the same in all languages (e.g. the currency code and whether the currency it belongs to a regime such as the European Economic and Monetary Union) and localized information (e.g. the currency's name in different languages).

All Currencies vs. Active Currencies

The currencies listed in the **Active Currencies** tab of Currency Server Manager are a subset of the currencies listed in the **All Currencies** tab. To be listed in the **Active Currencies** section, all of the following conditions must apply:

1. The currency must be known to Currency Server, i.e. be listed in the **All Currencies** section;
2. Foreign exchange (FX) rate data for the currency must be available from the FX feed(s), or via manual input;
3. Your organization decided that the specific currency is useful within the context where Currency Server is operating.

Related Topics

- For step-by-step configuration instructions and advice, see Initial Configuration and Code Examples.
- For more information about localized currency properties, see Internationalization.
- For more information about ISO 4217 currency codes, see Currency Codes.
- For more information about the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.

3.2 Currency Codes

Currency Server uses official ISO 4217 currency codes. The codes covered by this standard, as well as the registered trademark "ISO", are the property of the International Organization for Standardization (ISO), and are used under license.

Alpha-3 vs. Numerical Codes

Currency Server supports both Alpha-3 three-letter codes (e.g. "EUR", "USD", "JPY", etc.) and numerical codes (e.g. 978, 840, 392, etc.) Unless you need to work with systems that use numerical codes, we recommend the use of Alpha-3 codes in any software and scripts you may develop.

Alpha-3 codes are unique for each currency, and are usually available even for older 20th century currencies. Numerical codes, instead, are sometimes reused for different currencies (e.g. after a country is split or has its name or currency changed), and may not be available for some older currencies.

Alpha-3 strings are case-insensitive when they are accepted as input (e.g. in COM and .NET and SOAP Web service methods and properties), and uppercase when they are output by the software. When currencies are referenced by their numerical code, the software only considers the first matching currency, even if multiple currencies have the same numerical code. The software always uses uppercase Alpha-3 code strings (not numerical codes) in return values.

When defining a new currency (e.g. in Currency Server Manager), Alpha-3 codes are required, and must be unique. In order to support both older currencies which do not have a numerical code, and currencies where the same numerical code was reused for a different currency, numerical codes are optional and do not need to be unique.

Related Topics

- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.

Web Links

- International Organization for Standardization Homepage

3.3 The EMU and the Euro

Because of the impact of the changes and the technical requirements which the European Economic and Monetary Union (EMU) brings with it, Currency Server includes special support for EMU currencies.

The European Union

The European Union (EU) is a dynamic institution, which also reflects on the EMU process. Several waves of accessions followed the original community of six countries, bringing the total to 27 member states on February 1, 2020:

- 1957: Belgium, Germany, France, Italy, Luxembourg and the Netherlands;
- 1973: Denmark, Ireland and the United Kingdom;
- 1981: Greece;
- 1986: Spain and Portugal;
- 1995: Austria, Finland and Sweden;
- 2004: Cyprus, the Czech Republic, Estonia, Hungary, Latvia, Lithuania, Malta, Poland, the Slovak Republic and Slovenia;
- 2007: Bulgaria and Romania;
- 2013: Croatia;
- 2020: withdrawal of the United Kingdom ("Brexit").

The EMU

The process of achieving economic and monetary union was set out in the 1957 Treaty of Rome and in subsequent treaties (Maastricht Treaty of 1992, Amsterdam Treaty of 1997 and Lisbon Treaty of 2007) and decisions. The euro (currency code EUR) is the official currency of the European Union, and the EMU is the process by which EU member states replace their national currency with the euro and transfer management of monetary policy to the European Central Bank.

When a state joins the EMU its national currency becomes a sub-unit of the euro, at a fixed conversion rate with respect to the euro. During the transition phase, in which the national currency and the euro co-exist, a process called "triangulation", which is supported by Currency Server, is required to convert to and from the national currency and any non-EMU currencies. At the end of the transition phase the national currency is first replaced by euro banknotes and coins, and then ceases to be legal tender. A first group of twelve EU states completed this process between the end of 2001 and the first half of 2002, after a transition phase which lasted between two and three years. Member states that adopt the euro after this point can choose from a number of scenarios (e.g. a shorter transition period, including a "big bang" option) that provide for additional flexibility, also in consideration of the fact that euro banknotes and coins are already in circulation. Following the initial introduction, the euro replaced the former national currencies of Slovenia in 2007, Cyprus and Malta in 2008, Slovakia in 2009, Estonia in 2011, Latvia in 2014, Lithuania in 2015 and Croatia in 2023. Bulgaria is expected to adopt the euro in 2025.

Denmark (and the United Kingdom, which later withdrew from the EU) was granted special "euro opt-out" status in the Amsterdam Treaty, while Sweden decided not to meet the EMU exchange rate criteria. These EU member states, like some of the countries which joined the EU after the introduction of the euro, still use their national currencies and to different degrees conduct their own monetary policies.

The Euro outside the EU

The euro plays a role in the exchange rate regime of more than 50 countries outside the EU. The solutions adopted by these countries range from very close or even full links to the euro, such as the formal entitlement to use the euro as legal tender, to looser types of anchoring, such as peg arrangements and crawling fluctuation bands.

Non-EU countries such as Andorra, the Principality of Monaco, the Republic of San Marino and Vatican City have not only adopted the euro as their official currency, but are also minting euro coins on the basis of formal arrangements with the European Union.

Currency Server and the EMU

Currency Server specifically supports the EMU by means of:

- Definition of EMU as a regime a currency may belong to;
- Triangulation and rounding applied to EMU currencies (as required by law);
- Handling of official EMU conversion rates and precision values (as required by law);
- Preservation of internal exchange rate values expressed in EUR (as required by law);
- Differentiation between constant EMU conversion rates and fluctuating exchange rates (e.g. for inactivity warnings);
- One-click reset of official EMU data (**Active Currencies** tab of Currency Server Manager);
- Manual changes to regime and legal tender status (**All Currencies** tab of Currency Server Manager);
- Automatic notification and one-click approval of changes to regime and legal tender status (e.g. beginning and end of EMU transition phase).

Your Organization and the EMU

If your organization is based in a country which is scheduled to join the EMU or which is nearing completion of the EMU transition phase, you probably already have a corporate euro changeover plan. Similar transition plans are in place at larger organizations outside the EU that are indirectly affected by the EMU. If you have such a changeover department, be sure to share at least the Currency Server Quality Checklist and Operational Procedures with your euro changeover

personnel.

If you are the person responsible for keeping your organization and system up-to-date with EMU and with more general currency-related matters, you can use this documentation as a starting point to make sure that you are prepared for the changes which occur from time to time both within the EMU and with currencies worldwide. Currency Server includes an automatic notification and update system designed to inform you about changes with clear and detailed messages guiding you through any actions which may be required (e.g. removal of a currency which ceased to be legal tender, etc.)

Some of the actions that can be taken depend on your changeover plans. For example, if your activity involves payments, you have to decide for how long you decide to keep accepting and/or making payments or posting pricing information in a given currency after it has completed the EMU transition phase and has been replaced by the euro.

Related Topics

- For more information about EMU change scenarios supported by automatic updates, see Automatic Updates.
- For more information about EMU-related operational procedures, see Operational Procedures.
- For more information about precision, triangulation and rounding, see Rates and Conversions.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.

Web Links

- European Central Bank

3.4 Rates and Conversions

Currency Server takes into consideration a number of rules which apply to the storage and conversion of currency exchange rate data.

EMU-Specific Rules

Article 235 of the Maastricht Treaty and regulations 1103/97 and 974/98 of the Council of the European Union lay down rules concerning the euro, the handling of currency amounts, and conversions to, from and between European Economic and Monetary Union (EMU) national currency units. Currency Server was designed to comply with these rules, which are in general equal to or stricter than other non-EMU specifications, with which they are not known to conflict.

The software has also been tested by Currency System against version 2 of the Business Application Software Developers Association (BASDA) triangulation test data. To pass this test, the smallest unit for the BEF and LUF currencies has to be set to 1 franc (smallest unit = 1) instead of 50 centimes (smallest unit = 0.50, as set in the default data used by Currency Server, in consideration of the smallest coin in circulation and production in accordance with report II/717/97 of the European Union Directorate General II).

Precision

EMU regulations require the use of six significant figures (i.e. a maximum of six decimals) for expressing conversion rates, and not less than three decimals for intermediate calculations (e.g. as part of triangulation). Currency Server satisfies EMU precision requirements both when storing rates and when performing conversions. Intermediate calculations are neither rounded nor truncated.

It should be noted that Microsoft's default Currency data types (e.g. as originally implemented in OLE Automation) only support up to four decimals, and therefore do not meet EMU requirements. As of Visual Basic .NET, Microsoft appears to have phased out the Currency data type, preferring the use of the Decimal data type instead.

If you are not sure whether your conversion procedures and data types meet EMU rules, it is recommended that you use Currency Server's Convert() method for conversions.

Base Unit

Currency Server stores exchange rates and constant conversion rates using the euro as a reference currency in order to satisfy EMU regulations, which forbid the use of inverse rates and bilateral rates (unless they lead to identical results, which may not be guaranteed to be the case). This internal base unit is completely transparent to COM and Web service clients which use the Convert() and Rate() methods.

In Currency Server Manager rates are displayed using the base unit set in the **Active Currencies** tab, which is independent from the internal base unit.

Triangulation

When a country joins the EMU as a new member its national currency becomes a sub-unit of the euro, at a fixed conversion rate with respect to the euro. During the transition phase, in which the national currency and the euro co-exist, a process called "triangulation" is required to convert to and from the national currency and any non-EMU currencies. At the end of the transition phase the national currency is first replaced by euro banknotes and coins, and then ceases to be legal tender.

Currency Server applies triangulation as appropriate in consideration of the applicable regime when the Convert() method is used for conversions to or from EMU units.

Rounding

The static properties of each currency include the smallest unit for rounding purposes, which can be set either through the **All Currencies** tab of Currency Server Manager, or automatically. Rounding is important for the results to take into consideration what the smallest unit is for each currency, for example some countries don't use cents or fractions, but just round numbers for monetary amounts, while others use three decimals.

The Convert() and Rate() methods include a rounding option, which causes Currency Server to optionally round results to the nearest smallest unit. The rounding option only affects the result, not intermediate calculations. Although EMU regulations allow the optional rounding of intermediate results to not less than three decimals, Currency Server does not round intermediate calculations.

When rounding is enabled, Currency Server applies the "standard" rounding rule to results, which is consistent with EMU regulations (e.g. when rounding to the nearest cent 1.234 is rounded to 1.23, and 1.235 is rounded to 1.24). In cases where a conversion between a national currency unit and the euro unit leads to a result which is exactly half way, the sum is rounded up. If different national laws and practices apply to the scope of your implementation you should not set the rounding option, and round results using an appropriate custom procedure.

Related Topics

- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.

Web Links

- European Central Bank
- Business Application Software Developers Association (BASDA)

3.5 Internationalization

Currency Server was designed to offer complete support for the internationalization of currency-related applications, both when deployed as a stand-alone solution and when integrated as part of a broader internationalization effort.

The internationalization of a currency-enabled application, such as a database or an e-commerce system, is most likely to require support for more than just the conversion of currency amounts from one unit to another. Regardless of whether you need to build a simple list of daily reference

rates, or if instead you have to convert thousands of prices in a database, you will probably want to take into account rules such as those which apply to the formatting of currency amounts, which vary from locale to locale.

Locales

A locale is a set of cultural preferences which are assigned to a given region. Within a specific locale the same settings apply, i.e. a locale defines the largest possible region sharing the same country name, currency name, language and other localization settings.

A locale does not necessarily exactly match with a language or with a country. For example, the English language is used in multiple countries and regions (e.g. United States, Canada, United Kingdom, Australia, etc.) which are covered by multiple locales. Even when the language is the same, details such as currency symbols and the formatting of currency amounts may change between one locale and another. Similarly, countries may have multiple locales, e.g. Canada has different locales for its English and French language regions, where not only currency names change, but also other settings, e.g. the placement of the currency symbol relative to the amount. Associating the user selection of languages and/or locales to flag graphics, as is sometimes seen on websites, shows neither equal respect for different countries where the same language is spoken, nor consideration for the fact that a country may have more than one official language.

Currency Server not only fully supports the Windows locale functionality, which it uses to set the default initial software configuration, but it also adds software-specific properties to each locale (e.g. currency names, capitalization style, etc.)

The default locale used by Currency Server is US English. This locale is guaranteed to be made available by Currency Server on all versions of Windows, even if Windows itself does not provide it. A different default locale can be set in the **Locales** tab. When a COM or Web service client omits to specify a locale, the default locale is used.

Non-Localized Properties and Rules

Locales are independent from exchange rate data. Exchange rates and numerical results are not affected by locale settings. If you only use the software to obtain numerical results, you do not need to use locales at all. If you only use Currency Server to output formatted currency amounts on an English language website you can in general use the default locale and settings without further changes.

In the user interface, the clear separation between currency-specific data and locale-specific data becomes evident when looking at the first three tabs of Currency Server Manager: **Active Currencies** groups the dynamic properties of active currencies, **All Currencies** is dedicated to the static properties of all currencies, while **Locales** covers the localized properties of all currencies as well as other locale-specific options.

The rules which apply to the storage and conversion of currency exchange rate data, such as those governing precision, triangulation and rounding are not locale-specific, but rather they are associated to certain currency-specific properties such as the regime (used, for example, for triangulation) and the smallest unit used for rounding (when requested by the COM or Web service client).

While currency-specific rounding is supported (by taking into account the smallest unit of each currency), locale-specific rounding is not supported by Currency Server. If you require locale-specific or other rounding procedures different from the "standard" rounding rule, you should apply an appropriate custom procedure to the non-rounded amounts returned by Currency Server.

Locale Identifiers

When Currency Server COM and Web service (.NET and SOAP) clients need to reference a specific locale (e.g. as an argument to a method returning a localized currency name), the recommended identifier for the desired locale is the Windows locale identifier, or LCID, which is a 32-bit numerical value, as used in the Windows National Language Support API (NLSAPI). The LCID of US English is 1033 (decimal). The LCID column of the **Locales** tab lists the codes of all LCIDs which are available.

In addition to numerical LCID codes, Currency Server also supports the referencing of locales by means of three-character Windows locale codes (e.g. "DEU", i.e. two-character ISO 639-1 Alpha-2

language codes plus an additional character, as specified for use with the Windows "LOCALE_SABBREVLANGNAME" type, which is not necessarily the same as the three-character ISO 639-2 Alpha-3 standard). Two-letter ISO 639-2 Alpha-2 language codes (e.g. "de") are also supported, however they should only be used as a last resort, because there can be multiple locales for a given language, in which case Currency Server would resort to the first matching locale.

Currency Names, Symbols and Custom Data

Currency Server supports the following localized properties for each currency:

- Currency Name (e.g. "Swiss franc", "Schweizer Franken", "Franco svizzero", etc.);
- Currency Symbol (e.g. "$" or "US$");
- Custom String (e.g. "usd.gif" to indicate the location of a flag image file).

For example, the US English locale contains all US English names for all currencies of the world. It also contains the currency symbols of all currencies of the world, as used in the United States (the same currency may be referenced using a different symbol in different regions).

Each currency may also have a localized custom string, which is undefined by default, and which may be set as required by the application.

Other Currency Properties

Currency Server also associates a unique currency code (e.g. "USD") with each currency. Unlike currency symbols (which are localized, i.e. different locales may use different symbols for the same currency, e.g. "öS" vs. "S" for the Austrian schilling), currency codes are standard, and are not localized (they are the same all over the world). Be sure not to confuse localized currency symbols, which are meant for presentation purposes only, with standard currency codes, which can be used both to uniquely identify currencies and for presentation purposes.

Currency Server supports several other currency-specific properties, such as the exchange rate, the regime, the smallest unit, etc., however these properties, like the currency code, are not locale-specific, but rather they are independent of the locale. They only have to be set once for each currency (and not once for each currency for all locales which are used).

Capitalization

Currency names are usually written in lower case as a rule. In some languages (e.g. German) all nouns, including currency names, are always capitalized (both in body text and in titles). In some languages (e.g. English) titles are capitalized, which style may also be appropriate in drop-down lists, etc. For example, the Currency Server user interface uses title style in most contexts.

Internally, Currency Server stores currency names using the singular form and body text style, i.e. without title capitalization (headline style). For example, the default internal US English name of the USD currency is "US dollar", not "US dollars" or "US Dollar". When a COM or Web service client requests the name of the USD currency indicating the US English locale (LCID 1033) and the desired title style (True or False), Currency Server returns either "US Dollar" or "US dollar", as requested by the client. In title style the word "dollar" is capitalized as specified by the **Capitalize titles** setting in the **Edit Locale** dialog, which is enabled by default for US English.

Character Sets

Currency Server supports both Unicode and CP-1252 (a superset of ISO 8859-1 also known as ANSI or Latin 1) 8-bit encoding of the currency name, symbol and custom string.

The COM and Web service methods which output formatted currency strings, i.e. Convert() and Rate(), support the output of HTML character entity references instead of characters having decimal codes greater than 127 (e.g. "£ 12.34" instead of "£ 12.34").

The default currency names preset for the US English locale do not use Unicode characters. Unicode characters with no equivalent CP-1252 encoding may be present in the currency names when certain Asian and certain other locales are used.

The default currency symbols preset for the US English locale use Unicode characters for some Asian and certain other currencies. If your application does not support Unicode, you may want to edit these currency strings in the properties of the locale(s) you plan to use.

Formatting of Currency Amounts

The formatting of currency amounts for presentation purposes depends on several locale-specific (not currency-specific) currency formatting settings. For example, the same dollar amount written as "$12,345.67" in the US would be "$ 12.345,67" in Italy. These regional options include the decimal symbol (also known as fractional separator), the digit grouping symbol (also known as group separator), the placement of the currency code or symbol (before or after the amount, with or without a separator) and the type of digit grouping (e.g. "123456789" vs. "123 456 789" vs. "12 34 56 789"). These preferences can be modified in the **Edit Locale** dialog.

The Convert() and Rate() COM and Web service methods include powerful formatting functionality which takes into account both the requested locale and additional formatting options (e.g. output using standard currency code vs. localized currency symbol, etc.)

Locales, Countries, Domains and Currencies

The COM and Web service interfaces include internationalization functions such as LocaleToCurrency(), CountryToCurrency() and DomainToCurrency() which may make it easier for example to automatically set an initial default currency, e.g. on an e-commerce site based on the internet domain of the customer.

While useful, these functions have some limitations:

- Since there is no exact relationship between locales and currencies (e.g. a two-letter language locale code may match different countries with different currencies, and a three-letter code may match a country which is in a transition between two currencies), the result may be approximate (especially if two-letter language codes are used, whereas LCID codes or three-letter Windows locale codes lead to more focused results). If the input string is empty or could not be resolved, the function returns a currency for the default locale set in Currency Server. If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code) is returned.
- Since there is no exact relationship between countries and currencies (in some countries more than one currency may be in practical use, e.g. during the transition from an old currency to a new currency, or because a "stronger" currency is preferred), the result may be slightly approximate (but correct in most cases).
- Since there is no exact relationship between internet Top Level Domains (TLDs) and countries and/or currencies (e.g. a currency or internet top level domain may be used in multiple countries), the result may be approximate.

In the worst case, i.e. if multiple or no matches occur, Currency Server either returns the first currency matching the argument, or the first currency for the default locale, or, as a last resort, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code).

Related Topics

- For more information about precision, triangulation and rounding, see Rates and Conversions.
- For more information about editing locale settings, see The Locales Tab and The Edit Locale Dialog.
- For more information about currency properties, see Currencies of the World.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.

3.6 Automatic Updates

An important functionality of Currency Server is the ability to automatically collect data from external FX feeds, verifying it and applying it and/or issuing administrative notifications.

Currency Server can automatically update several types of data:

- Exchange rates of active currencies (via one or more FX servers)
- Names and other properties of all currencies of the world (via Currency World Monitor service)

- Status of European Economic and Monetary Union (EMU) currencies (via Currency World Monitor service)
- Availability of software updates
- Email and fax advisories

Some types of updates may require a subscription with a service provider. All information which can be set via automatic updates can also be set or changed manually, e.g. via Currency Server Manager.

Exchange Rate Updates

Official exchange rates, e.g. rates published by institutions such as national central banks, usually change once a day during weekdays (with no updates during week-ends). Major institutions generally publish their reference rates once a day at around noon or early afternoon in their local time. Private providers of exchange rate data, which for example may base their data on real time trading sources rather than on official reference data, may publish new rates more frequently.

After Currency Server has been configured to load new exchange rate data at a given time or interval, the software will automatically take care of updating the exchange rates of the active currencies, issuing notifications if data is missing, or fluctuates beyond a certain limit, etc.

The Currency Server software is, by design, independent from any given provider of data. Although Currency System itself also provides a currency data feed service, Currency Server not only supports several different providers of exchange rate data (both free and subscription-based), but it is also possible to create custom FX feed filters. Should there be a problem with a given FX feed which cannot be resolved, a different provider of data can be chosen with one or two mouse clicks in the FX Feeds tab of Currency Server Manager.

Currency World Monitor Service

Exchange rates are not the only currency properties which change over time. For example, when a state joins the European Economic and Monetary Union (EMU), its national currency may become a sub-unit of the euro (during a transition phase which may last months or years) before being replaced by the euro. During this timeframe the software must employ a triangulation procedure to convert to and from the national currency. Also, countries themselves may change name (and their currency with them), or split into two or more new countries (with new currencies), or form new unions with other countries, or simply replace an old currency with a new one.

While exchange rate updates occur at least once every weekday, modifications to "static" currency properties (like changes to currency names, or EMU status changes) are considerably less frequent, i.e. they occur about once every few months. Nevertheless, it can be easy to lose oversight of such changes, and longer lists of currencies can quickly get obsolete even if they are not contaminated with old data (e.g. wrong currency names, inclusion of currencies which ceased to be legal tender, etc.) to begin with, as is often the case with lists which can be found on the internet and which are sometimes just a sum of unverified data from different sources.

Access to the Currency World Monitor service, which also causes appropriate notifications to be issued when necessary, is set in the All Currencies tab of Currency Server Manager.

Worldwide Currency Data Updates

Currency System is a licensee of ISO 4217 currency codes and is immediately notified by the ISO 4217 Maintenance Agency whenever a change occurs. Currency System also actively monitors other changes such as the status of the EMU, internationalization trends, new currency symbols, rounding procedures, etc. This information is incorporated into the Currency Server software at compile time. Additionally, updates to this static data are made available as part of the Currency World Monitor service provided by Currency System.

Currency Server specifically supports the following change scenarios for all currencies of the world (not just the currencies listed as active):

- New currency is introduced
- Currency ceases to be legal tender
- Change of regime status (e.g. EMU)

- Currency changes code
- Change of other currency properties (e.g. smallest unit, localized names, symbols)

These special updates are independent of exchange rate updates (which are provided by different sources), and are provided by Currency System as part of the Currency World Monitor service.

EMU Currency Data Updates

Currency Server specifically supports the following change scenarios for currencies of the European Economic and Monetary Union:

- Currency joins EMU (conversion rate is locked, triangulation begins)
- Currency is replaced by euro (ceasing to be legal tender)
- Currency abandons EMU (triangulation ceases, fluctuating exchange rates as provided by external sources)

These special updates are independent of exchange rate updates (which are provided by different sources), and are provided by Currency System as part of the Currency World Monitor service.

If the EMU status of a currency has changed since the previous update an appropriate notification is issued by the software, advising to select **Reset EMU** in the **Active Currencies** tab, so that the new list of EMU currencies and the corresponding triangulation settings can be applied as appropriate. No changes are applied until **Reset EMU** is selected. The **Reset EMU** function includes an option to retain currencies which have ceased to be legal tender, if data for such currencies is available.

Software Updates

The Currency Server setup procedure includes the option to enable automatic software update checks via Software Director. When Software Director checks for updates, neither personal information nor data about installed software is sent to Currency System. From the system where Currency Server is installed, you can select **Check for Software Updates** in the Currency Server notification area icon, or **Check Now** in the **Software** tab of Currency Server Manager.

Email, RSS Feed, and Fax Advisories

From time to time, and only in relation to major changes, Currency System may issue currency and software-related advisories. This service is provided, for example, to better explain some unusual currency fluctuations which the software may have issued a warning about, or to introduce planned changes to the European Economic and Monetary Union, etc. You have the option to select email and/or fax as the preferred medium to receive such notifications, or to access an RSS feed. This service may be linked to certain online update services and support plans.

Related Topics

- For more information about quality management and euro changeover procedures, see Operational Procedures.
- For more information about editing the static properties of all currencies of the world, see The All Currencies Tab.
- For more information about editing the dynamic properties of active currencies, see The Active Currencies Tab.
- For more information about automatic update options, see The FX Feeds Tab and Use with Task Scheduler.
- For examples of programmatic access to the Currency Server methods and properties, see Code Examples.
- For more information about ISO 4217 currency codes, see Currency Codes.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.
- For more information about the possible privacy implications of updates, see Privacy Information.

Web Links

- For more information about the Currency World Monitor service, see Currency World Monitor on the Currency System website.

3.7 Recommended Reading

Just as we tried to design the software side of Currency Server for maximum performance, reliability, and ease of use, we listened to our customers' most frequently asked questions about currencies, and we prepared the following online information and guidelines. We hope that this will help you feel absolutely confident about your implementation of the software, resulting in impeccable information and quality on the end user side.

- Foreign Exchange Markets and Terminology. This article provides a general overview of foreign exchange markets and rates, explaining the meaning of words such as "interbank", "fx", "wholesale", "spot", "ask", "bid", "cash" and "swift".
- From Exchange Rates to Actual Charges. This article explains some of the differences between "interbank" ("official") exchange rates and the conditions which in practice are applied by banks and credit card companies when buying and selling small amounts of currency.
- The Euro. This article explains some basic concepts about the European Economic and Monetary Union's euro currency.
- Considerations on Exchange Rate Data Providers. This article explores some of the differences between different types of data providers based on considerations like update frequency, reliability and perceived fairness and authoritativeness.
- Exchange Rate Information and Disclaimers. When supplying currency-related data it is important to try to provide accurate information on how the system works and to set clear limits on scope and liability.
- Money and Style. Uppercase or lowercase? Singular or plural? Style guidelines for authors and editors.
- Currency Popularity Charts. More is not always better: optimizing the list of available currencies can improve usability and accuracy.

Related Topics

- For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about currency-related topics, see Working with Currencies.


Chapter 4

4 Administrative Tasks

This section covers:

- System Requirements
- Deployment Options
- Accessing the Software
- Services and Executable Files
- Use with Task Scheduler
- Security Sandbox

4.1 System Requirements

The following are the system requirements to install and run Currency Server.

Hardware

Currency Server was designed to run on any Intel Pentium (x86 or x64) and compatible system. The software is licensed per CPU core (4 cores for the Standard Edition, or more if upgraded, and unlimited cores for the Enterprise Edition).

Operating System

Currency Server has been tested on systems running Windows Server 2022, Windows Server 2019, Windows Server 2016, Windows Server 2012 (R2), Windows Server 2008 (R2), Windows 11, Windows 10, Windows 8, Windows 7 and Windows Vista. It works on both x64 and x86 systems.

Windows Installer (MSI)

Installation of Currency Server requires Windows Installer (MSI) 2.0 or higher. This component is part of newer versions of Windows, and is also available as a free download from Microsoft.

For additional information on MSI please refer to this support page on the Currency System website.

.NET Framework

Installation of the optional Currency Server Web service feature requires version 2.0 or higher (tested up to version 4.5) of the Microsoft .NET Framework.

Version 2.0 of the .NET Framework is also required by the Export() function of the COM interface.

Without version 2.0 of the .NET Framework, the Web service feature cannot be installed, and the Export() function, if used, will return an error.

The Azure Blob storage and Amazon S3 functionalities of the Post-Update Action feature require version 4.5 or higher of the Microsoft .NET Framework. Without version 4.5 of the .NET Framework, saving data to Azure Blob storage or to Amazon S3 will fail.

Internet Information Services (IIS)

Installation of the Currency Server Web service feature (serving .NET and SOAP clients) requires that the Internet Information Services (IIS) Windows component be installed and running, and that ASP.NET be installed and enabled.

In an IIS 7 or higher environment, the IIS 6 Metabase Compatibility feature must be present on the system.

Proxy Servers

Currency Server supports standard HTTP, WinSock, SOCKS and TIS FTP proxy servers, with or

without authentication (including Windows NT Challenge/Response).

Terminal Services

Terminal Services clients are specifically supported. The **Display on screen** notification option causes messages to be displayed to all Terminal Services users until the first user acknowledges the notification.

SMTP

Currency Server includes a built-in SMTP client to directly send email notifications to an SMTP server. This functionality is unrelated from Windows Messaging and does not require special privileges or optional operating system features.

Windows Messaging

Currency Server also supports the Windows Messaging Application Programming Interface (MAPI), which is part of Windows Messaging. Windows 2000, Windows XP, Windows Server 2003 and higher do not include Windows Messaging. On these systems, the Microsoft Mail PostOffice and Mail icons are not listed in Control Panel. MAPI support can be added by installing a client such as Microsoft Outlook in Corporate or Workgroup mode. Windows Messaging for Windows 2000 may also be available from Microsoft Product Support Services. For additional information on the availability of Windows Messaging for Windows 2000, Windows XP, Windows Server 2003 and higher please refer to the Microsoft Knowledge Base (e.g. articles 254458 and 314491).

In order to use the Windows Messaging notification options, the Currency Server service must be logged on as a user having access to the desired Windows Messaging Profile and belonging to the Administrators group on the computer. By default, Currency Server is installed to log on as System Account, which does not make available these messaging features (SMTP notification functionality is however always available). An option to configure the service to log on as a different user is available during setup.

Protocols and Ports

Currency Server normally collects currency information from the internet via HTTPS over TCP port 443, or via HTTP over TCP port 80. Some FX feed filters (e.g. Oanda FXP) support connections via different protocols and ports. The FX feed filters refer to the FX servers by host name, therefore requiring DNS client functionality for name resolution (usually TCP and UDP port 53 to and from port 53 and random port numbers greater than 1023). It is possible to manually edit the FX feed address or filter configuration to reference the servers by IP address rather than by host name, so that the software will run on systems with tighter security settings (e.g. where only port 80 may be open), however in this case it is strongly recommended to set up appropriate Notification options in order to be able to promptly correct any connection problems (e.g. a change of IP address).

For additional security-related information please refer to this white paper on the Currency System website.

Other Platforms

Additional (non-Windows) platforms are supported through the COM, Web service (.NET and SOAP) and JavaScript interfaces, and by means of the **Post-Update Actions** functionality, which makes it possible to write and upload data files in any file format (XML, CSV, etc.) and to any system.

The JavaScript (also known as ECMAScript or JScript) code generated by Currency Server is compatible with version 1.2, and as such works on major "version 4" browsers released since 1997-1998, and newer versions, including the most popular widget platforms.

Related Topics

- For more information about software installation, see Deployment Options.
- For more information about security, see Security Sandbox.

4.2 Deployment Options

Overview

Currency Server uses Windows Installer (MSI) technology to support advanced software management options. The software is normally distributed in a compact self-extracting "cssetup.exe" archive (which contains and normally automatically opens a setup.exe file and a Windows Installer package file).

When run as a command from the command prompt, **cssetup** includes a variety of extraction and installation options, as well as the possibility to access the Windows Installer package file (.msi) which is contained in the distribution archive and which is normally launched automatically when the setup archive is opened.

Additionally, the Currency Server .NET assembly and Web service files can also be installed manually.

Setup Using Command Line Parameters

The **cssetup** command can be used to initiate a silent (no user interface) installation using command line options in the following format:

> **cssetup [/s] [/j] [/x] [/f] [/w] [/v"/qn USERNAME**=\"username\" **USERCOMPANY**=\"companyname\" **LICENSEKEY**=licensekey **USERID**=userid **INSTALLDIR**=\"targetdirectory\" **SERVICEACCOUNT**=\"userid\" **SERVICEPASSWORD**=\"userpassword\" **WEBSERVICEVIRTUALDIR**=\"virtualdirectory\" **WEBSERVICESITE**=\"sitename\""]

The backslash (\) character is used when quotes (") appear inside the quoted **/v** parameter.

Example of installation:

```
cssetup /s /v"/qn USERNAME=\"John Doe\" USERCOMPANY=\"Acme Inc.\"
LICENSEKEY=12345-12345-12345-12345 USERID=\"myfxaccount\" INSTALLDIR=\"C:
\Program Files\Currency System\Currency Server\" SERVICEACCOUNT=\"ACMEINC\jd
\" SERVICEPASSWORD=\"johnpwd\" WEBSERVICEVIRTUALDIR=\"webservices\"
WEBSERVICESITE=\"My Website\""
```

Example of uninstallation:

```
cssetup /s /x /v/qn
```

The following setup options are available in the **cssetup** command.

Option	Description
/s	Silent extraction (no progress indication).
/j	MSI Advertise mode.
/x	MSI Uninstall mode.
/f	MSI Repair mode.
/w	Wait until MSI is finished and return any return codes generated by MsiExec.exe.
/v *parameters*	MSI command line parameters.
/qn	No user interface (MSI option).
USERNAME	User name (MSI property, defaults to current user name data).
USERCOMPANY	Organization name (MSI property, defaults to current user company data).
LICENSEKEY	One or more semicolon-separated license keys (MSI property).
USERID	Optional User ID assigned by the data source (MSI property, unused by default).
INSTALLDIR	Installation directory (MSI property, defaults to "[ProgramFilesFolder]\Cloanto\Currency

Option	Description
	Server").
SERVICEACCOUNT	"Currency Server" service logon username (MSI property, defaults to "Local System account" data).
SERVICEPASSWORD	"Currency Server" service logon password (MSI property, defaults to "Local System account" data).
WEBSERVICEVIRTUALDIR	Name of virtual directory for Web service (MSI property, defaults to "CurrencyServer").
WEBSERVICESITE	Destination site name, also referred to as Description in IIS Properties (MSI property, defaults to first site).

Windows Installer Package Files

In order to extract the .msi package file and other files required for deployment using Windows Installer run **cssetup** as in the following example:

```
cssetup /s /e /f "C:\Currency Server Admin"
```

The following table shows the **cssetup** command options used.

Option	Description
/s	Silent extraction (no progress indication).
/e	Do not start setup.
/f	Target directory.

On Windows 2000, Windows XP, Windows Server 2003 and newer versions of Windows, Windows Installer can be configured using Group Policy and Active Directory to manage installations.

For more information please refer to the Windows Installer sections of Windows Help.

Microsoft Systems Management Server (SMS)

Please contact us if you need a package definition file (PDF) to deploy the software using Microsoft Systems Management Server (SMS).

Manual Installation of .NET Assembly and Web Service Files

The Currency Server Web service interface is an optional feature of the Currency Server setup procedure. The Web service is not required for the operation of the COM interface, and can be installed at any time either with the installer or manually.

The functionality of the Web service is provided by two files: "CurrencyServer.dll" and "CurrencyServer5.asmx". These files are stored in the "Web Service" subdirectory of the Currency Server installation directory.

The "CurrencyServer.dll" file is known as the .NET assembly, and contains the assembly manifest. The Currency Server .NET assembly is strong-named to allow for installation as a shared assembly into the Global Assembly Cache (GAC). A shared assembly can be used from other assemblies on the same system, regardless of their location. The following example shows how to install the assembly into the GAC:

```
C:\Program Files (x86)\Cloanto\Currency Server\Web Service>gacutil /i
CurrencyServer.dll
```

The "CurrencyServer5.asmx" file is the Web service entry point in IIS. You may rename this file if you wish. The .asmx file contains a reference to the .NET assembly file, which must be located either in the "bin" subdirectory, or in the GAC.

To manually install the Currency Server Web service copy "CurrencyServer5.asmx" to the root of the desired IIS virtual directory and "CurrencyServer.dll" to the "bin" subdirectory of the virtual

directory (unless you installed the .NET assembly in the GAC). Make sure that the Execute Permissions for the virtual directory are set to either "Scripts Only", or "Scripts and Executables". Instead of copying the files, you can mount the "Web Service" subdirectory of the Currency Server installation directory as an IIS virtual directory, which is what the Currency Server setup procedure does.

Windows Installer Security Warning

Some of the Windows Installer options described in this chapter include the possibility to specify the "Log on as" settings of the "Currency Server" service. If a user other than "Local System account" (which is the default) is set, then, under certain conditions listed below, the specified service logon username and password may be temporarily stored as plain text, posing a potential security threat if a malicious user has access to the system:

- If the installation logging option is enabled, either explicitly (Windows Installer /L [flags] log file command line option) or as a system policy, then depending on the logging flags the log file may include the service logon details.
- If a reboot is required to complete the installation (which is normally not the case), then the installation options (which include the service logon details) are stored in the system registry until the installation is complete.

Related Topics

- For more information about system requirements and configuration, see System Requirements.
- For more information about services and files, see Services and Executable Files.
- For more information about custom FX feed filters, see FX Feed Filters.
- For more information about client validation executables, see Client Validation Modules.
- For more information about custom action executables, see Custom Action Modules.
- For more information about the administration of calculator clients, see Client-Server Configurations.
- For more information about security, see Security Sandbox.

4.3 Accessing the Software

Currency Server can be accessed and controlled using different interfaces:

- Currency Server Manager is used to configure the currencies and other software options which are then used for daily updates, exchange rate information and conversion services, error notifications, etc.
- The Currency Server icon in the notification area gives quick access to Currency Server Manager, to Currency Server custom action files (i.e. scripts and other executables stored in the "Actions" subdirectory of the Currency Server installation directory) and to other frequently used functionality.
- The COM interface (officially named COM, but also referred to as Automation or ActiveX, depending on the context and the implementation) exposes the internal objects of the software, so that other programs and scripts can request exchange rate information, conversions and other currency-related services.
- The .NET Web service interface exposes the internal objects of the software using Microsoft's XML Web services platform.
- The SOAP (Simple Object Access Protocol) Web service interface exposes the internal objects of the software using XML requests and answers over HTTP, making it ideal for cross-platform applications and for access by remote clients.
- The Windows Task Scheduler can be used to define advanced schedules for the collection of exchange rate data or to initiate other tasks which can communicate with Currency Server over the COM, .NET and SOAP interfaces.

Related Topics

- For more information about configuration, see Initial Configuration.
- For more information about best practices, see Quality Checklist and Operational Procedures.

4.4 Services and Executable Files

On Windows NT and higher versions of Windows applications that need to start and do their work independently from the users who may or may not be logged on to the computer are implemented as services. Currency Server uses two different services, which both also verify the responsiveness of each other at the intervals specified by the **Application Monitor** setting. Other executable files provide additional functionality.

"Currency Server" Service

This is the main service, which makes sure that the currencies are updated as scheduled. It is also used to provide a default user context when executing external code as part of the **Custom Action** functions, and to provide the functionality required by the Message() and UpdateNow() methods of the Admin object (COM) and the AdminMessage() and AdminUpdateNow() Web service methods (.NET and SOAP). Other than for these two methods, COM support in general is not implemented in this service, but as a considerably faster in-process server DLL.

By default, the service "Log on as" option is set to "Local System account". It does not matter whether "Interact with desktop" is enabled or not, as this functionality is not required by this service.

If a Windows Messaging notification option is selected in the **Notifications** tab, then the service must be set to log on as a user having access to the selected profile. Similarly, if a user-specific configuration is required for internet access (e.g. to access exchange rate data through a firewall), then the service must be set to log on as a user having the necessary internet access privileges. This user must also belong to the Administrators group of the local machine, or otherwise the software will not be able to write to the system registry, which is required for proper operation. The user must also have "Log on as service" rights (these rights are enabled automatically, if necessary, when the "Log on as" credentials are set in the Currency Server installer or in the Services tool in the Control Panel or in Computer Management). An error entry is added to the system application log if the settings do not give the service both access to the selected Windows Messaging profile and write access to the registry.

Windows Messaging notification option is selected in the **Notifications** tab, then the service must be set to log on as a user having access to the selected profile. This user must also belong to the Administrators group of the local machine, or otherwise the software will not be able to write to the system registry, which is required for proper operation. The user must also have "Log on as service" rights (these rights are enabled automatically, if necessary, when the "Log on as" credentials are set in the Currency Server installer or in the Services tool in the Control Panel or in Computer Management). An error entry is added to the system application log if the settings do not give the service both access to the selected Windows Messaging profile and write access to the registry.

"Currency Server Messenger" Service

This service takes care of the display of on-screen information and error messages.

The service requires that the service "Log on as" option be set to "Local System account". This is the default setting, and does not need to be changed. An error entry is added to the system application log if the default settings are modified.

Executable Files

Currency Server makes use of the following executable files:

- "curncsen.dll" and "curncsen_64.dll" contain localized data (x86 and x64 versions). The files are stored in the Currency Server installation directory.
- "curncsmn.exe" contains the code for Currency Server Manager (excluding the Control Panel GUI), notification area icon and on-screen messages. The file is stored in the Currency Server installation directory.
- "curncsmp.exe" contains the code for the Currency Server Manager Control Panel GUI. The file is stored in the Currency Server installation directory.
- "curncsrv.dll" is the legacy COM in-process server DLL, included for compatibility with clients

written for previous versions of Currency Server. The file is stored in the Currency Server installation directory.

- "curncsrv.exe" is run by the Currency Server service to handle updates and to monitor currency data. The file is stored in the Currency Server installation directory.
- "curncsvm.exe" is the Currency Server Messenger service executable file. The file is stored in the Currency Server installation directory.
- "curncsvs.exe" is the Currency Server service executable file. The file is stored in the Currency Server installation directory.
- "curncsvx.exe" is run by the Currency Server service to export currency data after an update. The file is stored in the Currency Server installation directory.
- "curnsrv5.dll" and "curncrv5_64.dll" are the Currency Server COM in-process server DLLs (x86 and x64 versions). The files are stored in the Currency Server installation directory.
- "CurrencyServer.dll" is the Web service assembly, and contains the assembly manifest. This file is stored in the "Web Service/bin" directory inside the Currency Server installation directory.
- "CurrencyServer5.asmx" is the Web service entry point. This file is stored at the root of the "Web Service" subdirectory of the Currency Server installation directory.
- The "Actions" subdirectory of the Currency Server installation directory is the recommended location to store executables and scripts which access the software functionality via its interfaces. Executables placed in this directory are automatically accessible via the Currency Server notification area icon menu.
- The "Filters" subdirectory of the Currency Server installation directory contains the FX feed filters.
- The "Validators" subdirectory of the Currency Server installation directory contains the Web service client validation executables.

Related Topics

- For more information about system requirements and configuration, see System Requirements.
- For more information about software installation, see Deployment Options.
- For more information about custom FX feed filters, see FX Feed Filters.
- For more information about client validation executables, see Client Validation Modules.
- For more information about custom action executables, see Custom Action Modules.
- For more information about security, see Security Sandbox.

4.5 Use with Task Scheduler

The built-in **Auto-Update** option in the **FX Feeds tab** can be set to schedule either a relatively simple "every X hours and Y minutes" schedule, or a schedule based on expiration information contained inside the data itself (if provided by the data source). If you prefer to take advantage of an equivalent automatic update procedure initiated by the Windows Task Scheduler, disable the **Auto-Update** option in Currency Server and set the scheduler to run the **UpdateNow.vbs** command file, which is located in the "Actions" subdirectory of the Currency Server installation directory (in the English version of Windows this usually defaults to "C:\Program Files\Cloanto \Currency Server").

The update is performed by the "Currency Server" service, so no special account requirements apply to run **UpdateNow.vbs**, which only sends a message to the service. Task Scheduler requires that the user account have "Log on as a batch job" privileges on the local system.

A possible application of this procedure could for example be the creation of a schedule where Currency Server collects new exchange rates every Monday to Friday every hour between the exact time at which new exchange rates are expected to be published and the evening (to allow for possible corrections which could be published by the data source).

This procedure uses the COM interface and will not work if the **Grant access to COM clients** option is disabled.

Related Topics

- For more information about automatic update options, see The FX Feeds Tab.
- For more information on post-update actions, see The Post-Update Action Dialog.

4.6 Security Sandbox

Currency Server includes a security sandbox to restrict access to internal objects and to limit interaction with the local file system. These measures were designed to prevent the software from being reconfigured or otherwise used to further damage an already compromised system on which Currency Server is running.

The security sandbox includes the following measures:

- FX feed filter executables, client validation executables and custom action executables must be stored inside the "Filters", "Validators" and "Actions" subfolder of the Currency Server installation folder, respectively. Any such files cannot be accessed or executed outside of these folders. This prevents other files (e.g. system files or other known files) from being accessed through Currency Server.
- Log files written by Currency Server must have a name ending with ".log". The suffix is automatically appended if missing or different. This prevents abuse of the logging functionality in order to overwrite other types of files.
- Configuration files read and written by Currency Server must have a name ending with ".cs-cnf". The suffix is automatically appended if missing or different. This prevents abuse of the configuration load/save functionality to overwrite or otherwise access files other than those intended.
- The license key parameter provided by Web service clients is normalized (""", "/", "\" and ":" are converted to underscore characters) and truncated after 254 characters. This helps prevent possible buffer overrun, directory traversal and other command line argument exploits when the handling of license key validation is performed by external code (not under the control of Currency Server).
- The URLs which are passed to external FX feed filter executables are truncated at 254 characters. This helps prevent possible buffer overrun exploits when the initial processing of currency data is performed by external code (not under the control of Currency Server).
- Standard objects exposed via the COM and Web service (.NET and SOAP) interfaces are read-only.
- Access to administrative objects (loading and saving of configuration settings, and invocation of exchange rate updates and notification messages) via the COM and Web service interfaces is disabled by default. Configurations loaded via this administrative functionality remain subject to the restrictions of the security sandbox.
- Sample code is not distributed and installed with Currency Server, but rather is available for separate installation from the Currency System website.
- All executable files which are part of Currency Server are digitally signed with Authenticode. Currency Server was designed to be installed and run on systems where policies prohibit the execution of unsigned code.

Web Links

- Currency Server Security Analysis on the Currency System website.

Chapter 5

5 Currency Server Manager

This section covers:

5.1 Overview

Currency Server Manager provides access to those software configuration options which allow the software to automatically refresh currency information and, without further maintenance, provide up-to-date currency information to COM and Web service (.NET and SOAP) clients.

The Currency Server options can only be modified by a user having administrator privileges. If a specific user account was assigned to the "Currency Server" service (which is required for the service to be able to use the Windows Messaging options for notification purposes), then that user account must have both administrator privileges and access to the same messaging profile as the user who accesses Currency Server Manager.

While Currency Server Manager is open, it has exclusive write access to the currency data (**Active Currencies** tab). This causes scheduled updates and updates requested by COM and Web service clients to fail (other operations keep working normally, only updates are blocked). An appropriate message is displayed in either case after Currency Server Manager has been closed, giving the option to perform the update and the related custom action, if any.

Related Topics

- For more information about additional configuration options, see Advanced Registry Settings.

5.2 The Active Currencies Tab

Active currencies are currencies for which up-to-date exchange rates are available. The **Active Currencies** tab allows you to add, edit or remove currencies from the list of currencies which have their exchange rates updated and which are available to COM and Web service (.NET and SOAP) clients. These currencies are a subset of **All Currencies**.

For rapid initial configuration, click the **Populate** button. This will populate the list of active currencies and exchange rates using the default FX feed. You can then remove currencies that you don't need.

Alternatively, to manually set up Currency Server for first use with one or more new FX feed(s) you may want to select **Updates may add or remove currencies: Yes (full refresh)** and click **Update Now** to adjust the list to the available exchange rates, then click **Reset EMU**, then remove all currencies that are not needed, and finally select **Updates may add or remove currencies: No (lock all)**.

Selecting **Update Now** establishes a connection to the FX feed(s), updating the available exchange rates. If **Updates may add or remove currencies** is enabled, **Update Now** refreshes the list of active currencies accordingly, building a new list of the currencies supported by the FX feed(s). If a **Post-Update Action** is set in the **Clients** tab, the execution of the custom action is deferred until Currency Server Manager is closed and intent to execute the action is confirmed.

The **Base Unit** setting allows you to change the currency unit relative to which exchange rates are displayed in this tab of Currency Server Manager. The internal exchange rate data is always preserved as originally collected from the FX feed, and is not affected by this setting. In particular, please note that if you need to change the unit relative to which amounts to be converted are indicated, that is done by setting the FromCurrency parameter in the Convert() method (COM, .NET and SOAP).

You can reset the default values for the EMU currencies by clicking **Reset EMU**, which adds these currencies to the list, assigning them the official constant conversion rate. If the software has information about currencies which have ceased to be legal tender (having been permanently replaced by the euro) you will be asked whether you would like to also have those old currencies included, which is usually only of interest if you need to work on historical data.

The **FX Feed** column displays the name of the FX feed from which the exchange rate data is set to be sourced for the currency, or "Any". Click **Edit** to change the source. If the source is set to "Any" (which is the default), the data from the first FX feed that supports that currency is used, in the order listed in the **FX Feeds** tab.

The **Feeds** column lists the number of FX feeds which provided exchange rate data for each currency during the last update.

The **Age** column indicates the number of days since the last exchange rate update.

The **Hits** column displays the number of hits (accesses) from COM, .NET and SOAP clients.

The **!** column displays a warning symbol for currencies which were referenced by recent warning or error messages (e.g. lack of updates, excessive fluctuations, etc.) Click **Reset Hits** or **Reset Warnings** to respectively reset the hit or warning information.

The **Rank** column displays currency popularity information. The most popular currencies are marked by three stars (currency rank 3). Rank 3 currencies are included in more than 50% of worldwide foreign exchange transactions. Rank 3 and Rank 2 currencies combined cover more than 95% of international transactions. Rank 1 currencies should be added with special care and limited to an as-needed basis, as it can be more difficult to source exchange rate data that is meaningful for a broad variety of contexts.

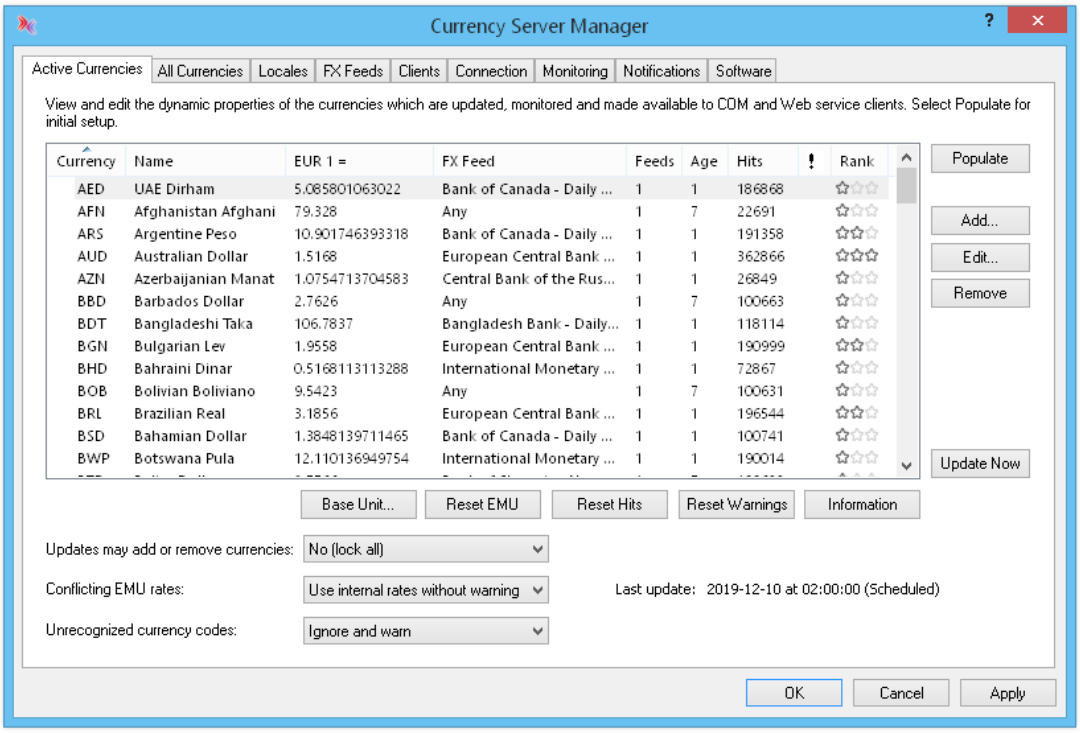
Click **Information** to open a dialog containing information fields collected by the software together with the exchange rate data. The fields include the name(s) of the FX feed(s), the copyright information, the creation and expiration date and time of the currency information, and the combined contents of one or more messages. If more than one creation or expiration date and time were found during the collection, then the oldest data is considered, in order to provide refresh indications based on the first data which expires. If only a creation date, but no creation time information was loaded, then the creation time field is not displayed, but internally the software considers a time of 23:00 UTC, which is the earliest time at which major Asian, European and American currency reference rates are assumed to have been published. This assumption is used if no expiration data was provided and if the **Auto-Update** configuration (**FX Feeds** tab) does not otherwise force an update by means of the **Update at least every** field.

Updates may add or remove currencies specifies whether manual, scheduled or client-invoked exchange rate updates not only update the actual rates, but also may add currencies for which rates are available and remove currencies for which rates are not available. If **No (lock all)** is selected, then no new currencies are added or removed by the updates, but only the exchange rates for the currencies already appearing in the list are updated. If no exchange rate is found for one or more locked currencies (except for EMU currencies), then a warning message is displayed, and cached exchange rate values are used. EMU (European Economic and Monetary Union) currencies can be locked separately via the **Yes (lock only EMU)** setting, because their official rates are constant and do not require daily updates, so they can be retained in the list even if they are not available from the FX feed(s). A warning notification is issued if no exchange rate data is available for non-EMU currencies which are locked. When the list is locked currencies can still be added or removed manually by using the **Add** and **Remove** buttons. If **Yes (full refresh)** is selected, then new currencies may automatically be added or removed, potentially

resulting in unexpectedly short or long lists of currencies. This setting is useful for initial configuration.

Conflicting EMU rates specifies how to deal with EMU currencies for which an official constant rate has been set, but a different value is provided during an update. The recommended setting is **Use internal rates without warning**. When this option is selected, no warning messages are issued if the data contains euro conversion rates which are different from the official rates.

Unrecognized currency codes specifies the action to be taken when updates are allowed to add currencies to the list set in the **Active Currencies** tab, and an FX feed provides data for a currency which is not listed in the **All Currencies** list, which by default includes all currency codes of the world and can be automatically updated. The recommended setting is **Add to All Currencies and warn**.



Related Topics

- For more information about editing the static properties of all currencies of the world, see The All Currencies Tab.
- For more information about editing locale settings, see The Locales Tab and The Edit Locale Dialog.
- For more information about ISO 4217 currency codes, see Currency Codes.
- For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.
- For more information about the possible privacy implications of updates, see Privacy Information.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

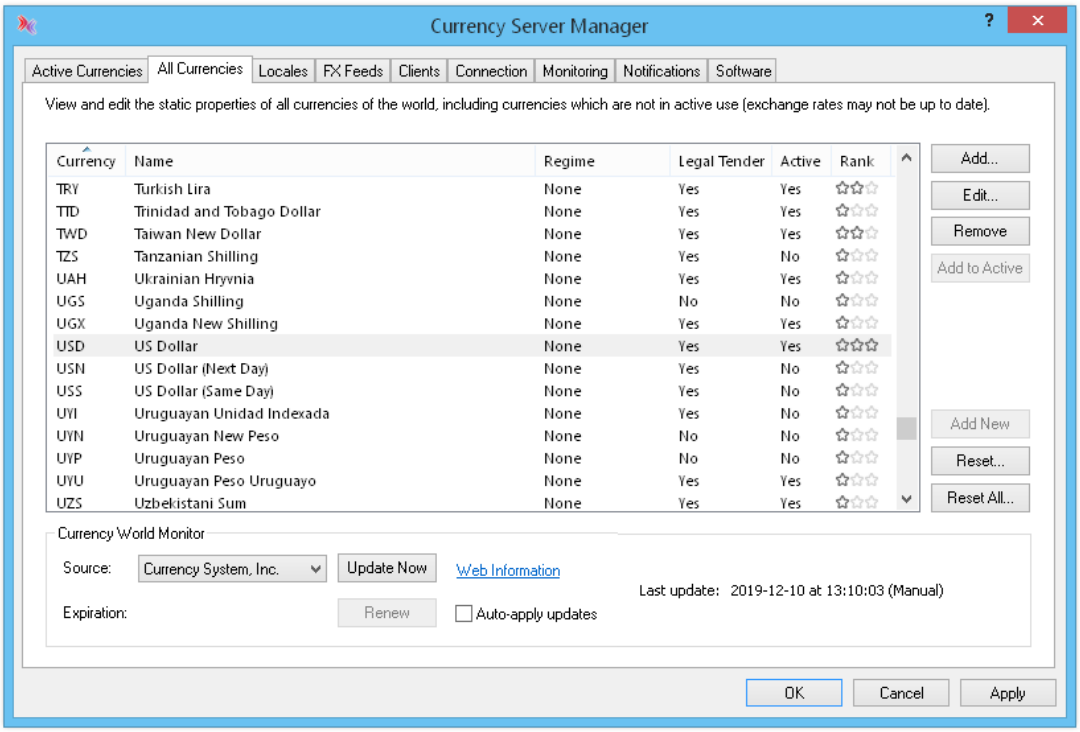
5.3 The All Currencies Tab

The **All Currencies** tab allows you to edit the static properties, i.e. all attributes other than the exchange rates, of all currencies of the world. This information does not change as frequently as the exchange rates, and can be updated both automatically and manually (by using this tab).

ISO 4217 Alpha-3 codes are used to uniquely identify each currency. Codes are also provided for some non-currency commodities, e.g. gold and silver.

The **Currency**, **Name**, **Regime** and **Legal Tender** columns list the corresponding fields from the **Edit Currency** dialog. The Active column indicates whether currencies are included in the list of **Active Currencies**. Use the **Rank** information to get a feeling of how "popular" a currency is (the most popular currencies are marked by three stars). Sort by **Legal Tender** or **Active** to group the currencies in the list to more easily visually check if you are using any obsolete currencies.

Add New adds new currencies to the list, based on preset internal software tables or Currency World Monitor updates, without affecting currencies that are already in the list. **Reset** resets the selected currencies, while **Reset All** resets all currencies (also adding or removing entries). A confirmation dialog allows you to choose whether to reset only the general currency properties (currency code, smallest unit, etc.), or also the localized properties (default currency names and symbols, etc.) and the FX feed-specific properties (feed and inactivity exceptions), mirroring the three tabs of the **Edit Currency** dialog.



Currency World Monitor Service

Once a week or if necessary to process new data, and only during exchange rate data updates, Currency Server also verifies whether non-exchange rate updates are available from Currency System as part of its Currency World Monitor service. A connection to the Currency World Monitor service can also be forced by clicking **Update Now**.

If new data is available, it is loaded (but not applied) and an appropriate notification is issued (additional details may be available in a "currencyserver-cwm-details.txt" file in the log directory set in the **Notifications** tab). The message provides information about the nature of the change (e.g. a currency joining the EMU, or ceasing to be legal tender, etc.) and instructions on how to apply the change (e.g. by clicking **Reset All** in the **All Currencies** tab and/or **Reset EMU** in the

Active Currencies tab of Currency Server Manager). If this update process fails, the software continues to function normally (updates can still be applied manually, if so desired).

This functionality, which is enabled by default, can be disabled, if so desired (set **Source** to **None**).

Related Topics

- For more information about editing currency-specific properties, see The Edit Currency Dialog.
- For more information about editing the dynamic properties of active currencies, see The Active Currencies Tab.
- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information about editing locale settings, see The Locales Tab and The Edit Locale Dialog.
- For more information about ISO 4217 currency codes, see Currency Codes.
- For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

Web Links

- For more information about the Currency World Monitor service, see Currency World Monitor on the Currency System website.

5.4 The Edit Currency Dialog

When you **Add** or **Edit** a currency, the **Edit Currency** property sheet allows you to change the general currency properties separately from the localized information.

The **Currency Code** fields indicate the ISO 4217 currency codes. This information is preset in the software and can be updated automatically.

The **Rate** field indicates the exchange or conversion rate of the currency, relative to the indicated base unit. Unless the currency is part of a regime you can usually leave this field blank, as exchange rates of active currencies are automatically updated by the FX feed(s). Click **Base Unit**... in the **Active Currencies** tab if you prefer to use a different base unit.

The **Smallest unit** field indicates the smallest unit for rounding purposes (e.g. 0.01 = one cent), which usually is either the smallest coin for the given currency or the smallest unit for accounting purposes, if different. Trailing zeros can be used to customize the display of rounded values. If for example the **Smallest unit** is set to 0.50, an amount of 3.51 would be rounded to 3.50. If instead **Smallest unit** is set to 0.5, then the rounded amount would be 3.5. This difference only applies to results returned as strings, as in the Convert() and Rate() COM and Web service (.NET and SOAP) methods. A **Smallest unit** of 1 followed by a subtraction of 0.05 (performed by the client) would result in all amounts ending in .95. Remember to consider possible legal and consumer protection implications when configuring this option for "creative rounding" of product prices. This information is preset in the software and can be updated automatically.

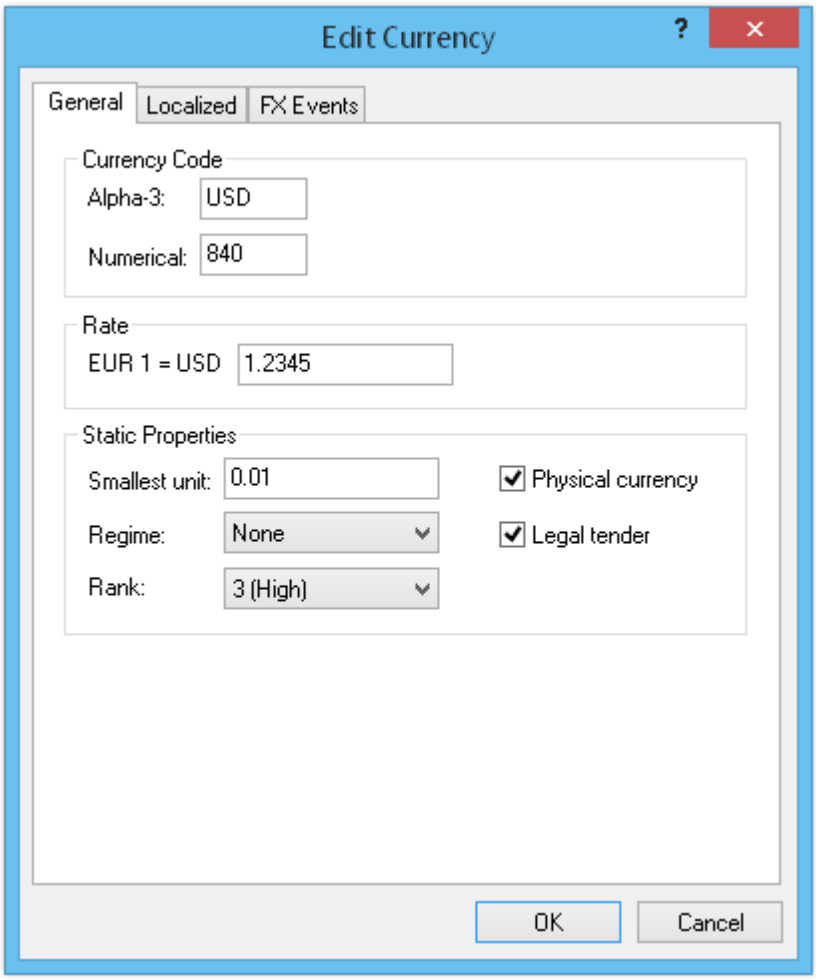
The **Physical currency** checkbox indicates whether the currency is a physical currency (i.e. "real" notes and coins) or not (e.g. a non-currency commodity, or a unit used only for accounting or other purposes). This information is preset in the software and can be updated automatically.

The **Legal tender** checkbox indicates whether the currency is legal tender or not. If this checkbox is selected, the currency is legal tender. Currency Server's global list of currencies includes codes and names of currencies which ceased to be legal tender. This information is preset in the software and can be updated automatically.

The **Regime** field indicates the regime this currency belongs to, if any. Supported regimes include **EMU** (for currencies which are part of the European Economic and Monetary Union) and **Pegged to EUR** (for non-EMU currencies which are pegged to the euro at a constant rate). This

information is preset in the software and can be updated automatically.

The **Rank** field indicates currency popularity information. Rank 3 currencies are among the most frequently used, and are included in more than 50% of worldwide foreign exchange transactions. Rank 3 and Rank 2 currencies combined cover more than 95% of international transactions.



Related Topics

- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.5 The Locales Tab

The **Locales** tab allows you to customize currency names, symbols and formats in different languages. Locales do not affect ISO 4217 Alpha-3 or numerical currency codes (which are universal), or the availability of currencies or their exchange rates (which are set in the Active Currencies tab).

If you use Currency Server on a website or other system in English language only (regardless of the number of currencies that are available), then you normally only need to configure the desired English language locale (the default settings are usually fine). If you don't use currency-related strings for display purposes, then you usually don't need to configure or use locales at all.

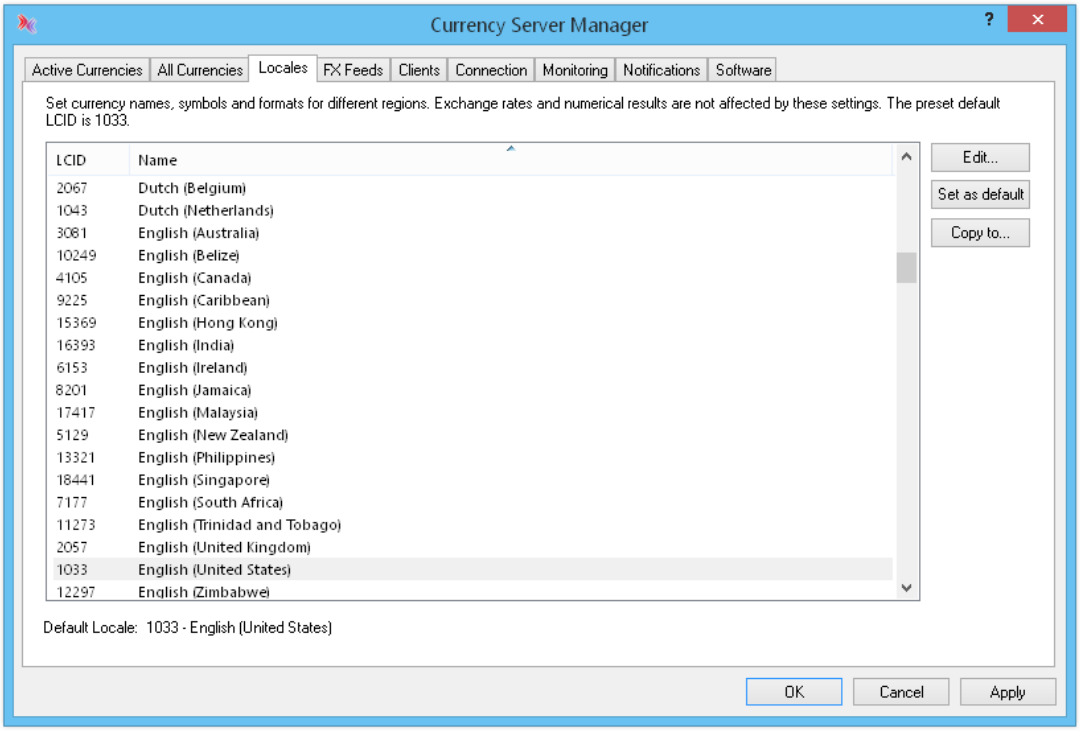
Windows locale identifier (LCID) codes are used to uniquely identify each locale. The preset default locale, which is used for currency names, symbols and formats when a locale is undefined or unavailable, is 1033 (US English). Although Currency Server provides its own custom locale

properties, support for LCIDs is provided by Windows. On some systems not all LCIDs may be installed by default. If this is the case, you can enable support for the desired locale in the Control Panel (**Regional and Language Options**, or **Regional Options** depending on the version of Windows).

Click the **Edit**... button to open the **Edit Locale** dialog, which allows you to modify the settings of the selected locale.

Click **Set as default** to set the default locale.

Click **Copy to**... to copy the currency names, symbols and formatting information from one locale to another locale.



Related Topics

- For more information about editing locale properties, see The Locale Dialog.
- For more information about localized currency properties, see Internationalization.
- For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about currency-related topics, see Working with Currencies and Recommended Reading.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.6 The Edit Locale Dialog

The **Edit Locale** dialog allows you to modify the settings for a given locale. The settings include the names, symbols and a custom field for all currencies, as used in the specified language and region (currency names and symbols may be different from language to language, and country to country).

The **Active** column indicates whether a currency is in active use (as in the **Active Currencies** tab), making it easier to only set information for currencies which are in use, if so desired.

Edit allows you to modify the currency name, symbol and custom string information of the selected currency for the current locale.

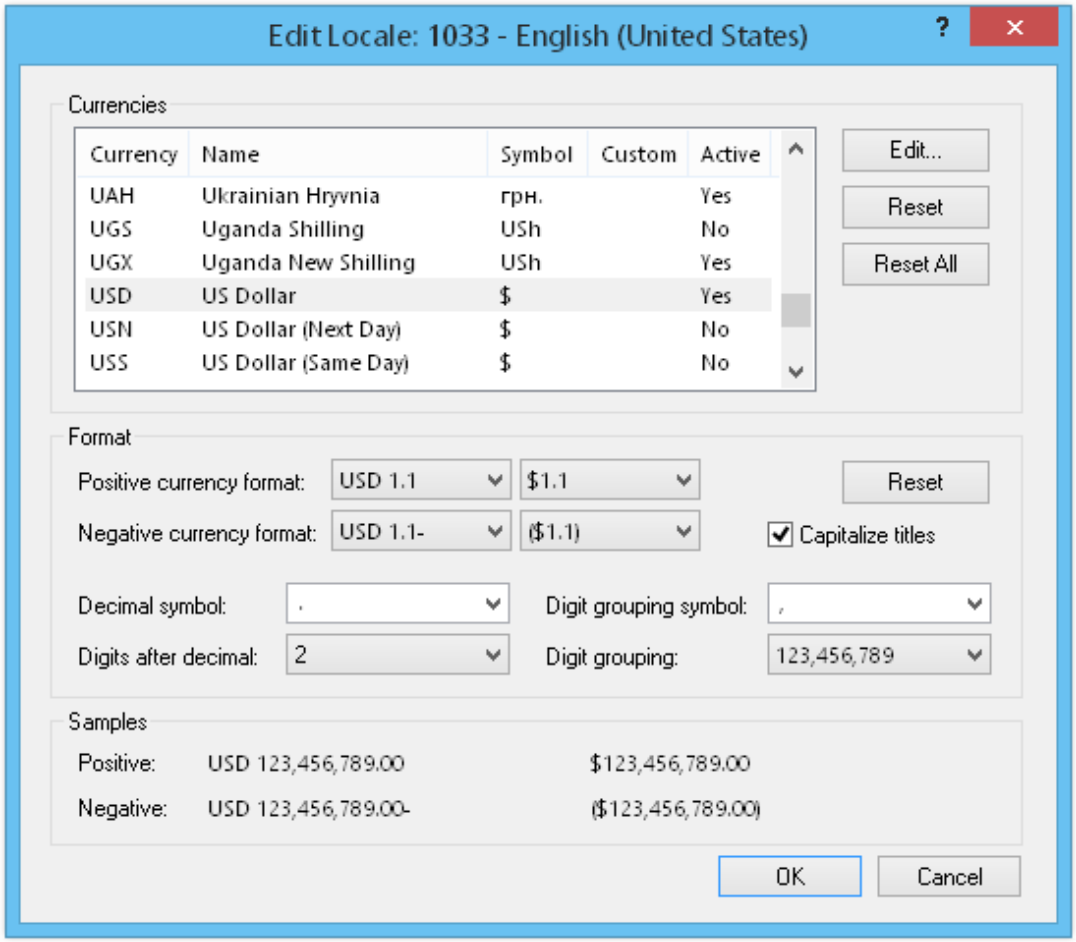
The **Reset** button, under **Currencies**, resets the currency name, symbol and custom string information of the selected currency for the current locale. Only localized currency properties are reset (general properties, such as exchange rate and smallest unit, are not reset). **Reset All** resets the same data for all currencies.

The **Reset** button, under **Formats**, reset the formatting of currency values for the current locale.

Various options are available to set and the preview the formatting of positive and negative currency values both when they appear next to their ISO code and when they are used with the local currency symbol (the two usages are often, but not always, identical).

Capitalize Titles specifies whether nouns and adjectives in currency names are always capitalized in titles (headline style, e.g. as used in English). Currency names (e.g. dollar, euro) are written in lower case in most languages, unless they appear at the beginning of a title, or if a language requires capitalization for all nouns (e.g. German).

Digits after decimal specifies how many digits appear after the decimal symbol in the formatting of currency values output by the Convert() and Rate() methods (COM, .NET and SOAP) when Round is false. If Round is true the number of digits is specified by the **Smallest Unit** property of the given currency (which is a general property of the currency, valid for all locales, and not a localized setting). This setting does not affect the precision of ConvertToNum(), RateNum() and other non-string methods.



Related Topics

- For more information about locale settings, see The Locales Tab.

- For more information about localized currency properties, see Internationalization.
- For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.
- For more information about currency-related topics, see Working with Currencies.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.7 The FX Feeds Tab

This tab contains the settings which indicate where to fetch the foreign exchange rate data from, how to apply the data, and whether and when the software should collect the data automatically.

The servers are listed in the order in which they are queried (from the top to the bottom).

Depending on the installed options and licenses, your software may support a single server, or multiple servers. If Currency Server supports only a single server, the **Add...** button may become unavailable after one server has been added. The Multiple FX Feeds component can be purchased and added in the **Software** tab, without reinstalling the software.

When multiple servers are used, the **Multiple servers** option specifies the operating mode (**Merge**, **Cross-check** or **Failover**).

In **Merge** mode unique data from multiple FX feeds is aggregated. Individual currencies may be set to fetch the data from any feed, or from a specific feed via the **FX feed** option in the **FX Events** tab of the **Edit Currency** dialog. By default, exchange rates for each currency are allowed to be fetched from any feed, and differing exchange rate entries for the same currency are ignored (the first rate found in any feed has priority). Feed-specific notification options can be set via the **On merge error** option in the **Edit FX Feed** dialog.

In **Cross-check** mode exchange rates for each currency must be provided by all servers and are cross-checked as specified by the **Fluctuation** settings in the **Monitoring** tab.

In **Failover** mode data is collected from the first server which does not return an error; if collection from a server returns an error, collection is attempted from the next server in the list.

Special circumstances, such as lack of network connectivity, server unavailability, unusual fluctuations, lack of updates, etc., may lead to warning or error conditions. Some of these conditions can be set in the **Monitoring** tab. Warnings return a warning condition and notification, but the operation is completed. Errors return an error condition and notification, whereby Currency Server continues to operate using the previously cached exchange rate data. One important difference between **Failover** and the other modes is that **Failover** can recover from an error on a given server, trying to collect data from the next server. **Merge** and **Cross-check** will stop at the first error, returning an error. **Failover** only returns an error if the last server returns an error.

Merge mode implicitly contains some failover functionality (if there are no errors), because if the rate for a currency is not available from a server, then it is fetched from the following server(s), if available. On the other hand, an error (not a warning, or one or more missing currencies) on one server causes the entire collection to fail with an error.

When using the **Cross-check** functionality keep in mind that you may be connecting to servers which do not update their data at the same time of the day. If your cross-check **Fluctuation** settings are relatively tight (or if no difference at all is allowed between the data of each server), you may experience warnings or errors for the sole fact that one server has already updated its data, and another has not. For this reason, if you use the **Cross-check** functionality, you may want to set an update schedule that keeps into account the update times of the different servers.

The **Auto-Update** options determines whether the software may automatically connect to the FX feed(s) and collect new exchange rate data.

While data is normally saved and retained across reboot, the **Update on startup** option can be useful in stateless server scenarios, to ensure that the local instance is current as soon as it is initiated.

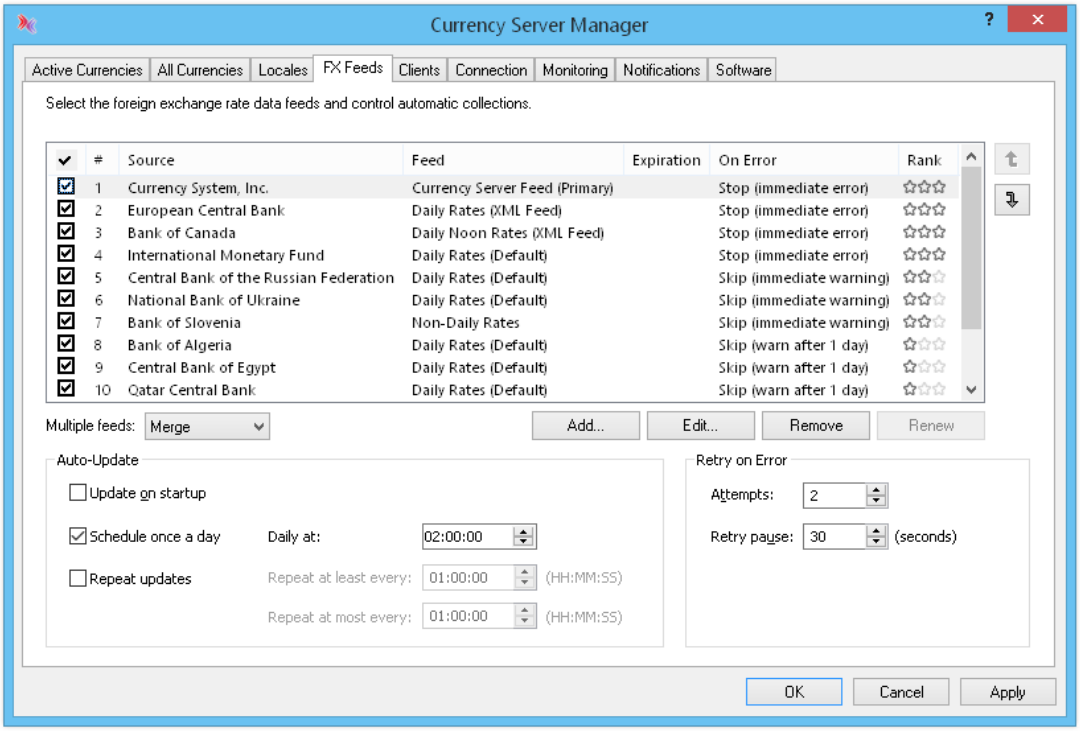
The **Schedule once a day** option makes it possible to set a once-a-day schedule, which may be

useful to integrate with other operations and for scenarios (e.g. accounting) where a predictable schedule is desired. For more complex schedules, see Use with Task Scheduler.

The **Repeat updates** option sets the currency information updates to be automatically repeated at the desired frequency. The **Repeat at least every** and **Repeat at most every** fields respectively indicate the maximum and minimum time between repeat auto-update operations. Within the range allowed by these two extremes, the software tries to only collect new data based on expiration information contained in the data itself (if provided by the data source). If the "at least" and "at most" update fields are set to the same value, for example 24 hours, or one hour, then the software will connect every 24 hours, or every hour, regardless of the data creation and expiration information. The interval further takes into account updates triggered by the **Update on startup** and **Schedule once a day** options, which are not constrained by the "at most" field.

The data may contain both a creation date and time and an expiration date and time. If expiration data is provided, then the software never looks for new data before the declared expiration time (unless the "at least" value or the daily schedule forces it to do so). Even after the data has expired, the software will only connect to the FX feed(s) within the limits set by the "at most" field. If data is collected from multiple servers, then all creation dates, if specified, are considered, and compared against the earliest expiration date found in the different sets of data, i.e. each time one item expires, the software reloads all items. If no expiration data is provided, then the software only looks for new data based on the "at least" value and the daily schedule time (if set).

The **Retry on Error** options allow you to specify how many times Currency Server should attempt to connect to each server before giving up (e.g. in case of connection-related errors). An error notification is only issued if all attempts including the last one fail. **Retry pause** specifies how long to wait before attempting to connect again to a server after a failed attempt.



Related Topics

- For more information about feed-specific options, see The Edit FX Feed Dialog.
- For more information about automatic update options, see Use with Task Scheduler.
- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information on post-update actions, see The Post-Update Action Dialog.
- For more information about custom FX feed filters, see FX Feed Filters.

- For more information about the possible privacy implications of updates, see Privacy Information.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.8 The Edit FX Feed Dialog

The **FX feed** field indicates the specific feed provided by a data source. If you are not familiar with the multitude of available feeds, you may want to consider the **Rank** column. FX Feeds that are recommended based on both popularity and reliability are marked by three stars (feed rank 3) by default.

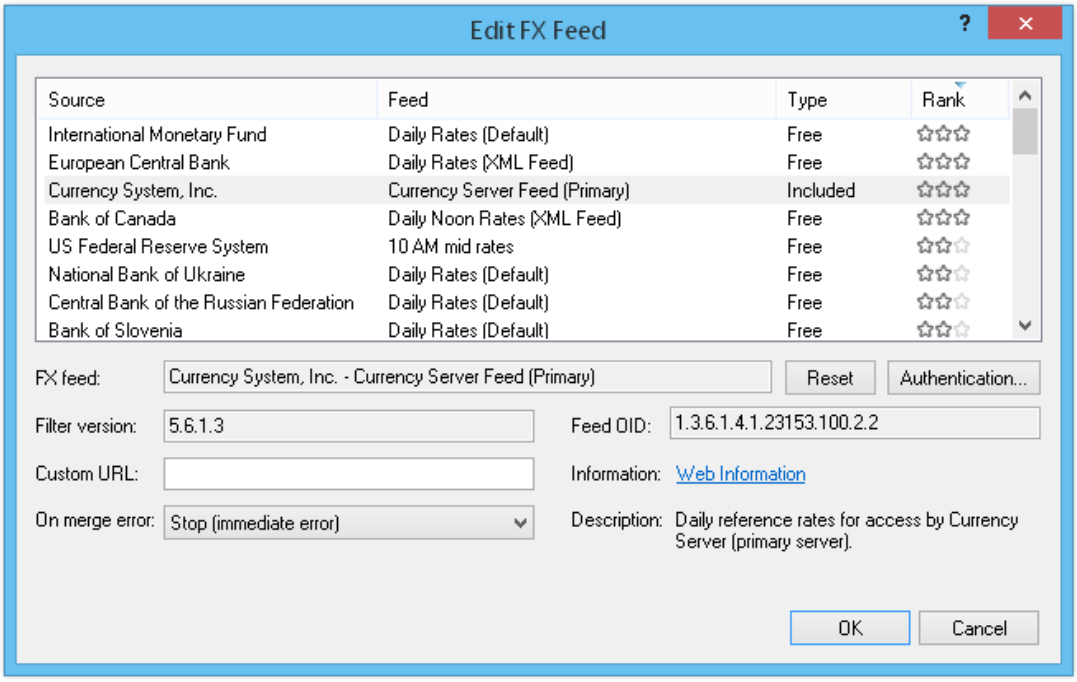
The **Subscription ID** field in the **Authentication** dialog allows you to specify the optional Subscription ID assigned to you by the data source. This information is usually only required by some commercial providers of exchange rate data, who may also refer to it as "Client ID", "User Name", etc. The other fields (user name and password) should only be filled-in if the currency data is otherwise password-protected. This authentication information is not the same as the internet and proxy access authentication fields in the **Connection** tab.

The **Filter version** and **Feed OID** fields directly reflect information set in the underlying FX feed filter.

The **Custom URL** field allows you to manually override the internal URL set in the FX feed filter.

Select the **Information** link to open a web page with additional information about the selected data source. Please refer directly to the data source for information on subscriptions, copyrights, conditions of use, limitations of liability and other terms which may apply.

The **On merge error** option specifies the action to be taken if an error occurs while accessing data from the feed when multiple FX feeds are accessed in **Merge** mode. Additionally, a time window during which feed-specific warning notification events are suppressed can be specified. During this window, a condition is considered resolved if a successful update takes place. Once access problems persist beyond the specified time window, a notification event is issued. **Immediate warning** is recommended for the most reputable sources. **Warn after 1 day** is recommended for sources that are known to experience occasional intra-day content or connectivity issues. **Warn after 1 week** may be preferred for sources that are known to experience content or connectivity issues that may persist for a few days.



Related Topics

- For more information about automatic update options, see Use with Task Scheduler.
- For more information about custom FX feed filters, see FX Feed Filters.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.9 The Clients Tab

This tab contains the settings to manage client accesses to the COM and Web service (.NET and SOAP) interfaces, and to set up a custom task to feed data to other types of clients.

If the **Grant access to COM clients** checkbox is selected, COM clients can access the administrative functions, which include the ability to control exchange rate updates, to issue notification events and to change the software configuration. When Currency Server is deployed on a web hosting platform it may be desirable to prevent certain administrative functions from being invoked through the COM interface (e.g. via ASP calls). If this option is disabled, as it is by default, then the COM methods which provide administrative functionality, such as Admin.UpdateNow() and Admin.Message(), are disabled.

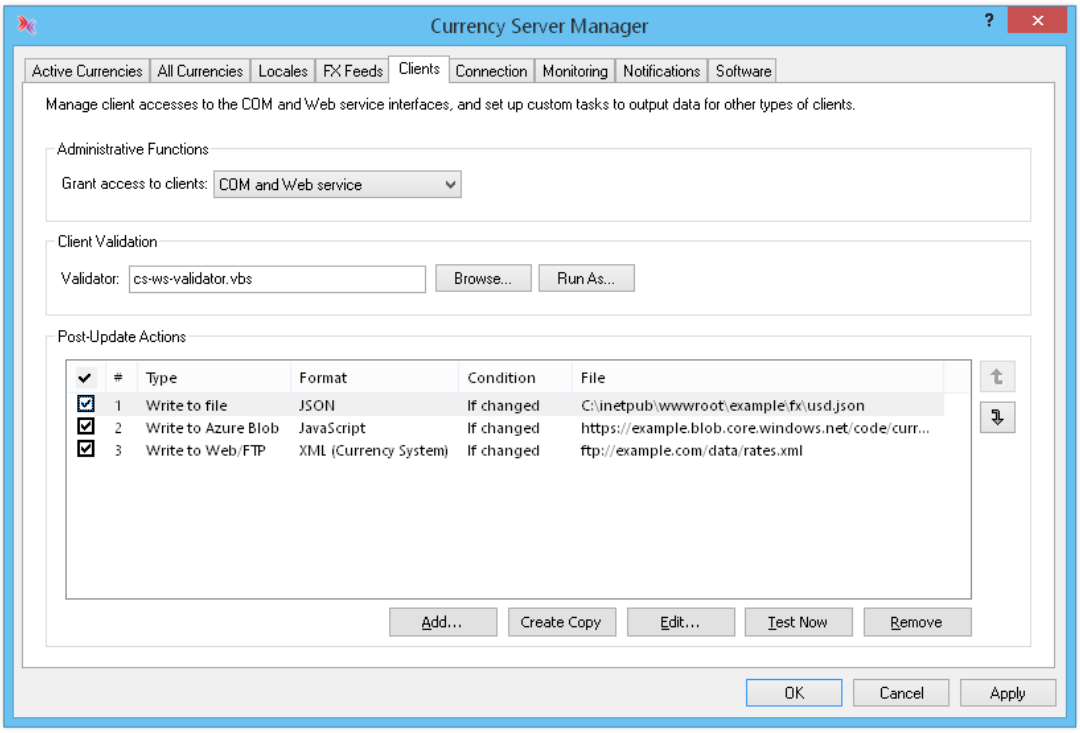
If the **Grant access to Web service clients** checkbox is selected, Web service clients can access the administrative functions, which include the ability to control exchange rate updates, to issue notification events and to change the software configuration. When Currency Server is used to provide a public web service it may be desirable to prevent certain administrative functions from being invoked through the public interface. If this option is disabled, as it is by default, then the Web service (.NET and SOAP) methods which provide administrative functionality, such as AdminUpdateNow() and AdminMessage(), are disabled.

Normally, Currency Server validates the license key provided by the Web service (.NET or SOAP) client before executing any Web service operation. Under **Client Validation**, you can specify a Web service client validation executable file, followed by optional arguments, to perform an additional check on the license key provided by the client (e.g. to provide your own subscription-based service). If the name contains spaces it must be surrounded by double quotes. The license key provided by the Web service client is automatically passed to the executable file as a parameter (after any other arguments, if specified). Client validation modules must be stored in

the "Validators" subdirectory of the Currency Server installation directory. For security reasons, executables placed outside of this directory will not be executed. If Currency Server includes an Unlimited Web Service Clients license, then the license key is not validated by Currency Server (it is only truncated to a maximum of 254 characters, and """, "/", "\" and ":" are converted to underscore characters), but only by the validation executable.

Run As... options are available to optionally set the user context to run the client validation, and for the Post-Update Actions executables. By default, the client validation code runs under the context of the invoking COM client. When invoked by a Web service client, by default the invoking context is that of the IIS Worker Process.

Click the **Add...**, **Edit...** or **Remove** button to open the **Post-Update Action** dialog, which allows you to modified the settings of the selected locale.



Related Topics

- For more information on post-update actions, see The Post-Update Action Dialog.
- For more information about FX feeds and automatic update options, see The FX Feeds Tab.
- For more information about client validation executables, see Client Validation Modules.
- For more information about custom action executables, see Custom Action Modules.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.10 The Post-Update Action Dialog

The **Post-Update Action** dialog lets you specify one or more custom actions to be performed after a currency update. This includes **Write file**, **Upload file** and **Execute**. For example, after new exchange rates have successfully been fetched from one or more FX feeds, Currency Server can write or upload a data file with the processed information to your network or web servers, or invoke an executable action to export to a database (SQL Server, etc.)

The **Authentication...** information (user name, password, domain) should be filled-in based on the access restrictions of the destination. This authentication information is not the same as the

internet and proxy access authentication fields in the **Connection** tab.

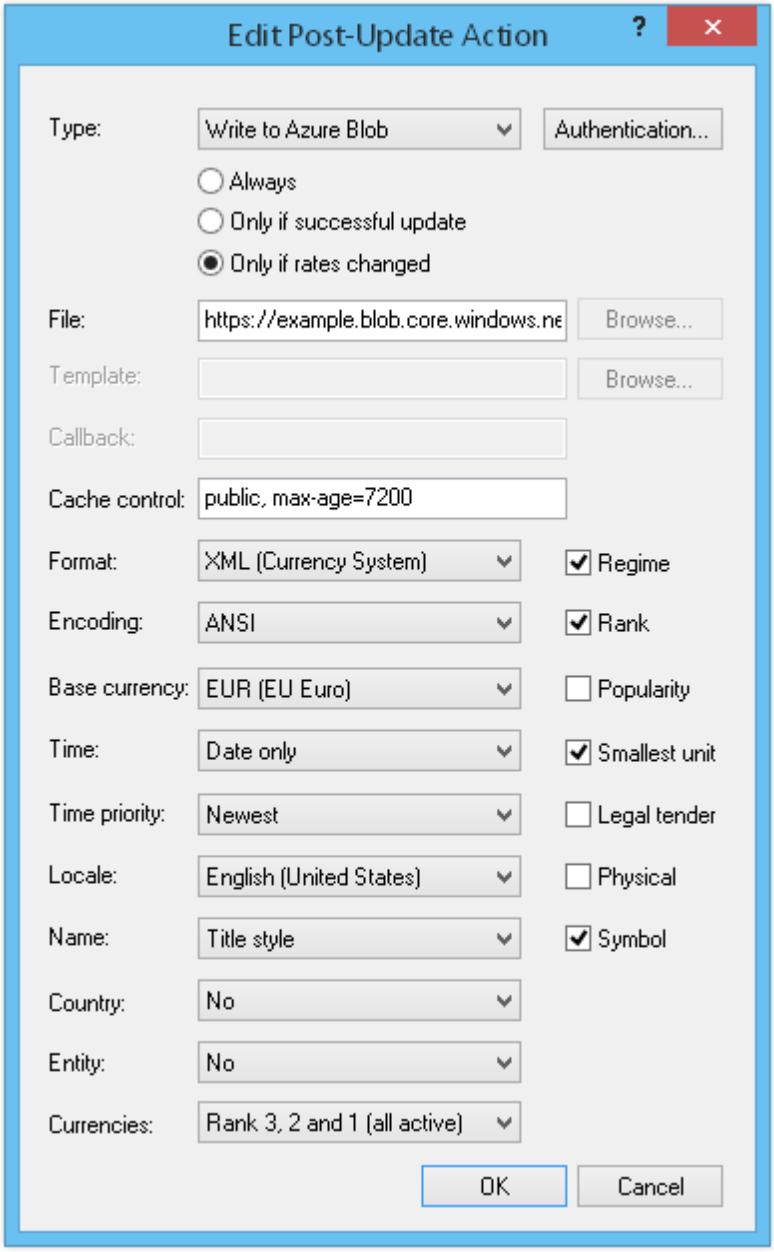
The execution options are **Always**, **Only if successful update** and **Only if rates changed**. If the latter option is set, the custom action is only executed if the update was successful and one or more exchange rates have actually changed since the previous update.

When an update is manually invoked by pressing **Update Now** in the **Active Currencies** tab, the execution of the custom action is deferred until Currency Server Manager is closed and intent to execute the action has been confirmed.

If the **File** name contains spaces it must be surrounded by double quotes.

If the action is **Execute**, the executable file may be followed by optional arguments. Custom action executables must be stored in the "Actions" subdirectory of the Currency Server installation directory. For security reasons, executables placed outside of this directory will not be executed.

For **Write** actions, the destination directory must exist (Currency Server can write files at existing locations, but not create new directories).



Related Topics

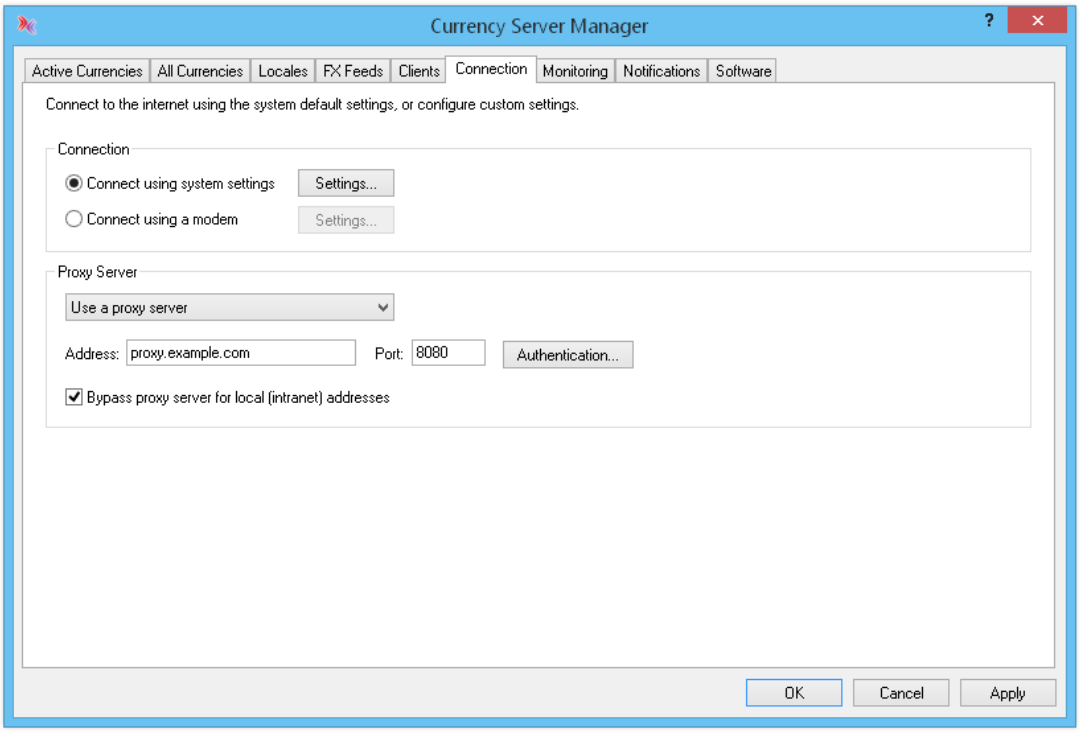
- For more information about adding and editing custom actions, see The Clients Tab.
- For more information about custom action executables, see Custom Action Modules.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.11 The Connection Tab

The options in the **Connection** tab are standard network connection settings, equivalent to the **Internet Options** in the **Control Panel**. By default Currency Server connects to the internet using the system default settings. You can set different options to have Currency Server access the internet or other network through a specific dial-up (modem) connection and/or a proxy server.

If you select **Connect using a modem** you can click **Settings**... to specify a Dial-Up Networking entry. If you leave the **User** and **Password** fields blank, the default settings for that connection will be used. The **Domain** field is usually optional: do not fill in this field unless your internet service provider gave you specific configuration instructions. Please note that if you enabled the **Hang up after dialing** checkbox, Currency Server will only hang up if it also dialed, and not if the line was already established manually, or by some other program.

Currency Server supports standard HTTP, WinSock, SOCKS and TIS FTP proxy servers, with or without authentication. Windows NT Challenge/Response authentication is supported. The proxy **Address** can be written in the usual "proxy.example.com" format, with the default port number for HTTP proxies being 80 (or 8080, e.g. to avoid conflicts with a web server running on the same system).



Related Topics

- For more information about proxy servers and system requirements, System Requirements.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.12 The Monitoring Tab

Use the options in this tab to monitor the exchange rates of the active currencies and the operation of the software for possible anomalies. You can determine how certain conditions should be monitored, and if certain events should lead to a warning or an error condition. The main difference between a warning and an error is that after an error the software discards the entire set of new exchange rates (and keeps working with the previous set of exchange rates), whereas a warning condition only triggers a notification (the new exchange rates are accepted and used).

It is very important that you also configure the notification options to make sure that you receive any notifications issued by the software when a warning or an error occur. We recommend that you also have procedures in place to respond to these notifications.

The Currency Server services can query various application processes from time to time, to verify that they are running and responding properly. Under **Application Monitor**, you can specify how frequently the Currency Server application processes cross-check each other. If the

software does not respond in a timely manner, a "Currency Server application not responding" or "Currency Server service not responding" message is issued. If this value is set to 0, no monitoring is performed.

Under **Fluctuation**, you can determine how Currency Server should monitor and respond to exchange rate fluctuations, both between successive updates from the same server(s) and between different servers at the same time. The **Maximum** field allows you to type the maximum allowed exchange rate fluctuation between updates. **Severity** specifies whether a warning or an error is issued when one or more exchange rates fluctuate beyond the maximum allowed. A notification event is triggered with either setting, however in the case of an error condition the new currency data is discarded (the previous exchange rate information for all currencies is retained) rather than being used.

For example, with a **Maximum** setting of 15% and a **Severity** of Error the entire currency update operation fails if the rate of one or more currencies has changed more than 15% since the previous update. This is unlikely to happen with real data when exchange rates are updated daily, and in general (but not always) it indicates an error in the data itself. Although major currencies are not likely to fluctuate by more than about 2% with respect to each other on a single "normal" day, occasionally a currency can fluctuate by much more. For example, in January 2001 the Turkish lira fluctuated by more than 40% on a single day, and the Icelandic krona fell repeatedly in October 2008, losing about 25% of its value in a single day. In such cases Currency System may issue an email/fax advisory to better explain why an update may have triggered a warning or error condition.

The **Cross-check** field allows you to set the maximum discrepancy allowed between exchange rates provided by different servers in **Cross-Check** mode. If this value is set to 0, the rates provided by different servers must be identical (when using this setting keep in mind that different servers may not update their rates simultaneously, and choose the timing of the check accordingly). **Severity** specifies whether a warning or an error is issued when one or more servers provide exchange rate data conflicting with the data provided by other servers. A notification event is triggered with either setting, however in the case of an error condition the new currency data is discarded (the previous exchange rate information for all currencies from all servers is retained) rather than being used.

Under **Inactivity Warning** you can set the maximum time allowed for the exchange rate data to remain unchanged. This type of monitoring takes into account two levels of severity: the inactivity of individual currencies, and lack of activity for the entire set of currencies (i.e. no currency changes). Extended lack of activity on a single currency may indicate an anomaly in the feed of that currency, or that the currency became pegged to another currency (in which case its **Regime** should be set accordingly). Lack of activity on the entire set, even for only a single day (other than week-ends and bank holidays), may indicate a problem with the scheduling of updates either at the local or at the FX feed level. This type of verification can detect both a non-error configuration issue (e.g. **Auto-update** turned off by mistake), which leaves the software isolated from the FX feed(s), and the more subtle possibility of a data source not updating its own data, thereby leaving old information online, or dropping support for one or more currencies, while updating others. Because one type of monitoring is a subset of the other, and one type of activity includes the other (as soon as any one currency changes, the entire set is considered changed as well), the maximum inactivity period for individual currencies cannot be set to a monitoring value smaller than that set for all currencies.

When all the exchange rates of all active currencies (except for currencies which are subunits of another currency at a constant conversion rate, e.g. EMU national currencies) stay unchanged for a time which exceeds the value set for **Entire set**, a warning is issued. If this value is set to 00:00:00, no warnings are issued (unless the condition occurs when the inactivity warning on individual currencies is enabled).

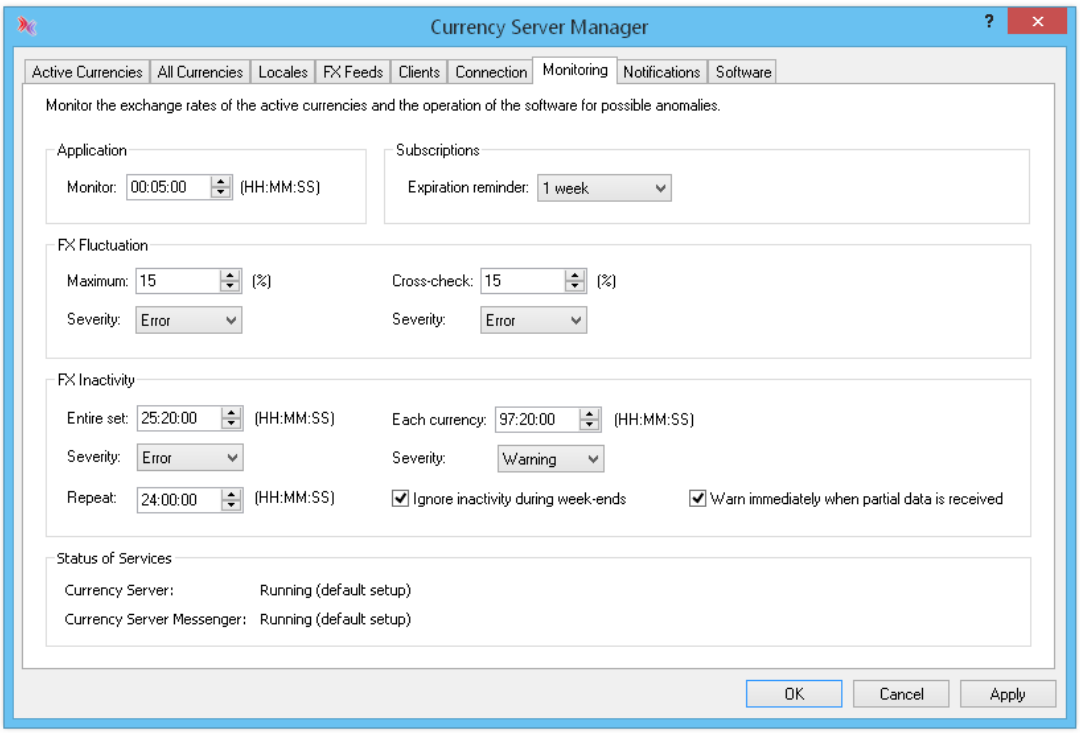
When the exchange rate of any single active currency (except for currencies which are subunits of another currency at a constant conversion rate) stays unchanged for a time which exceeds the value set for **Each currency**, a warning is issued. To switch off the additional monitoring of individual currencies, set this value to 00:00:00, or to the same value as that of **Entire set**.

The **Repeat** field allows you to specify how often inactivity warning notifications should be repeated. Depending on the FX feed, exchange rates are usually updated at least every working day (Monday to Friday except for bank holidays).

If the **Ignore inactivity during week-ends** checkbox is selected, week-ends (i.e. Saturdays and Sundays) are not accounted for in the inactivity warning timer.

Sometimes one or more FX servers may not provide data for all active currencies, in which case Currency Server retains the previous exchange rate information for the missing currencies, and issues a warning notification if the **Warn immediately when partial data is received** checkbox is selected. Additional inactivity warnings are always issued as set under **Each currency**.

The **Status of Services** group displays the status of the two Currency Server services, including inter-service communications. If an error condition is listed, check whether both services are running and review the account requirements for the Currency Server services.



Related Topics

- For more information about notification and logging options, see The Notifications Tab.
- For more information about FX feeds and automatic update options, see The FX Feeds Tab.
- For more information about best practices, see Quality Checklist and Operational Procedures.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.13 The Notifications Tab

Currency Server can issue three types of messages:

- Information (e.g. as provided by an FX feed together with the exchange rate data, or to inform about currency-related changes)
- Warning (e.g. extended lack of exchange rate updates)
- Error (e.g. inability to access one or more servers due to network problems)

Even in the case of warning or error messages the software keeps running, however, within the context of an exchange rate update (or attempted update), a warning indicates that the new exchange rate data has been loaded, whereas an error indicates that the software discarded the rates (valid cached data continues to be used).

It is very important that you configure the notification options to make sure that you receive any

notifications issued by the software. We recommend that you have procedures in place to respond to notifications issued by Currency Server.

The options which can be set in this tab also determine how the Admin.Message() (COM) and AdminMessage() (.NET and SOAP) methods send their messages.

If the **Display on screen** checkbox is selected, the messages are displayed on screen. If Terminal Services (or Remote Desktop Connection) are used to access the computer on which Currency Server is running, and multiple Terminal Services clients are active at the same time, messages are displayed to all Terminal Services users until the first user acknowledges the notification. Conditions resulting in the impossibility to issue an SMTP or Windows Messaging notification are always displayed on screen (and logged in the application log), regardless of the settings.

The **Write to application log** option causes messages to be stored in the application log. Service start and stop events and conditions resulting in the impossibility to issue an SMTP or Windows Messaging notification are always logged at least in the application log, regardless of the settings.

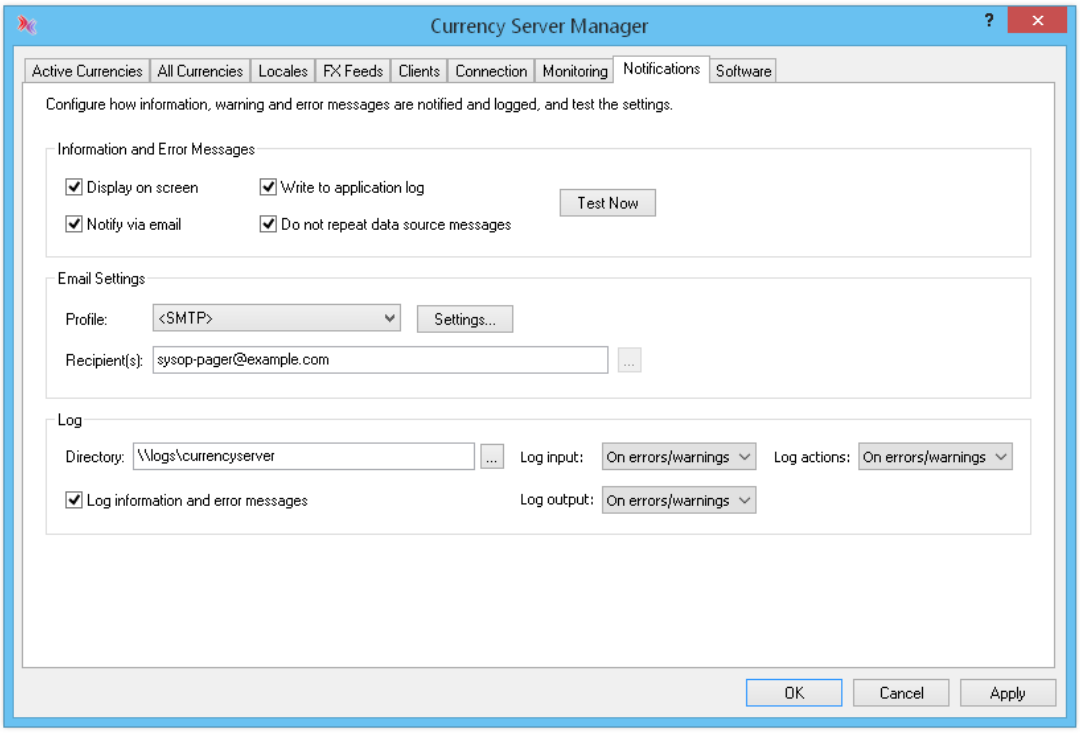
If the **Notify via email** option is enabled, the messages are notified using the specified SMTP or Windows Messaging settings.

Data providers may set a message to be displayed when the exchange rate data is loaded, for example to inform of imminent server address changes. If the **Do not repeat data source messages** checkbox is selected, then identical messages are not displayed each time the software connects to the FX feed(s), but only once.

Under **Email Settings** you can select **SMTP** or a **Windows Messaging** profile (if available on the system). Select **Settings**... to further adjust the available options. Use a space, comma or semicolon to separate multiple addresses. If available, click ... to display the Address Book.

The Windows Messaging functionality can provide capabilities which extend to different transport mechanisms, including fax, pager, etc., however, unlike SMTP, it also requires that the "Currency Server" service be logged on as a user having access to the selected messaging profile. Windows Messaging requires that at least one profile be defined (e.g. via **Mail** in the **Control Panel**, if available on the system, or in a messaging client such as Microsoft Outlook).

Under **Log**, you can choose to save a copy of the information and error messages. It is also possible to save a copy of the input data as it is received and processed from the feeds (**FX Feeds** tab), and the output data which may be written after an update (**Post-Update Action** dialog). If the directory or file does not exist, it is created. Input and output data is logged in timestamped subdirectories. In the case of the input data, both the raw data received by the feed(s) and the processed data as interpreted by the FX feed filter, are saved.



Related Topics

- For more information about services and messaging profiles, see Services and Executable Files.
- For more information about software and data monitoring options, see The Monitoring Tab.
- For more information about best practices, see Quality Checklist and Operational Procedures.
- For additional, specific context help about the various fields press F1 or select **?** and click the desired item in the software.

5.14 The Software Tab

This tab allows you to view software version information, activate the product and add components.

The **Registered to** and **Company** fields can be set or updated when adding or editing a license key.

As the software is licensed per CPU cores, the **CPUs** field indicates the number of CPU cores that have been detected on the system. Each hyper-threading CPU counts as a single CPU for licensing purposes, even if Windows counts them as multiple CPUs, whereas each core in a multiple-core processor count as a CPU. If necessary, you can add CPU Upgrade licenses by clicking **Add...** or **Paste**.

The **Servers** field indicates the type and simultaneous number of servers that the software can connect to. The standard version of the software is normally licensed to connect to any type of server of your choice (one server at a time, i.e. with no aggregation, cross-check or failover between multiple servers). If the software was licensed to you by a specific provider of exchange rate data, then this field may indicate that the software can only connect to servers from that provider. If you have a Multiple FX Feeds license, meaning that the software is enabled work with multiple server (from the same provider, or from different providers), then this field will indicate so.

If the Currency Server Web service (which is an optional installation feature) was installed, the

Web service field provides a link to the .NET Web service help page, from which you can also access the WSDL (Web Services Description Language) file.

The **Add..., Edit...**, **Remove** and **Paste** buttons can be used to manage the registration of license keys, such as:

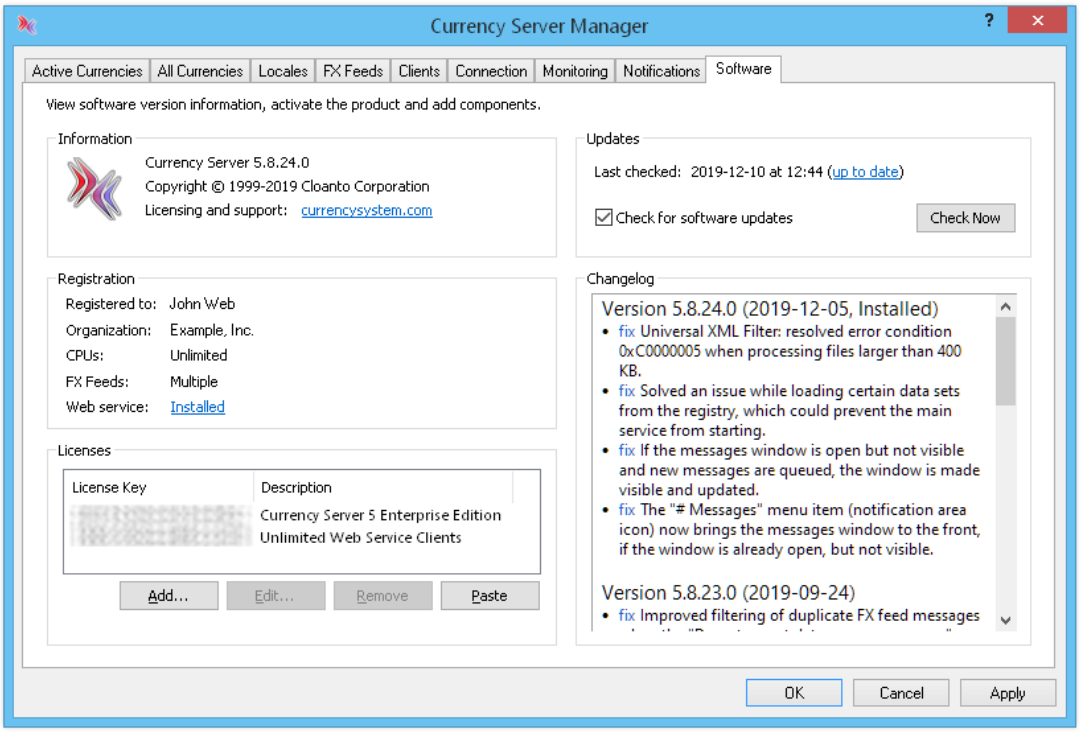
- Evaluation or Evaluation Extension License
- CPU Upgrade License
- Multiple FX Feeds License
- Unlimited Web Service Clients License

You can use **Paste** to add multiple licenses simultaneously.

You don't need to reinstall the software if you are switching from an evaluation license to a commercial license. Just enter the license key(s) after purchase. If instead you need to upgrade from an older version of the software just run the setup procedure of the new version. In either case, your existing software configuration will be preserved.

Currency Server supports both free and subscription-based providers of exchange rate data. Some commercial providers of data may require that the software log on to their system by providing certain access credentials (user name and password) or a User ID (also referred to as "Client ID", "User Name", etc.) Since this information is server-specific, i.e. it can be different for each provider and server to which the software connects, it is set in the **FX Feeds** tab (click **Add...** or **Edit...**, then enter the appropriate information under **Authentication** and **User ID**).

Please note that licensing information (user name, organization name, license keys) is not loaded or saved when the software configuration is loaded or saved with the Admin.Load() and Admin.Save() COM methods and the corresponding AdminLoad() and AdminSave() Web service (.NET and SOAP) methods.



Related Topics

- For more information about software, licensing, service and support options, see Options.
- For more information about privacy, see Privacy Information.
- For additional, specific context help about the various fields press F1 or select **?** and click the

desired item in the software.

Chapter 6

6 Programming Interfaces

This section covers:

6.1 The COM Interface

Overview

The Component Object Model (COM, also referred to as Automation, or ActiveX, depending on the context and the implementation) makes it possible to work with an application's objects from another application or development tool. Applied to Currency Server, this means that any application (e.g. database, spreadsheet, word processor, etc.) or programming language (including scripting languages and ASP) that supports COM has access to the services provided by Currency Server.

The Distributed Component Object Model (DCOM) extends the Component Object Model (COM) to support communication among objects on different computers. The Currency Server COM interface is DCOM-compatible.

Except for the Admin.Message() and Admin.UpdateNow() methods, which are not performance-critical, the Currency Server COM server is implemented as an in-process server DLL, which provides superior performance compared to out-of-process servers.

Installation

The COM interface is always installed. COM client access to administrative functions is disabled by default, and can be enabled through a setting in the **Clients** tab of Currency Server Manager.

ProgIDs

Currency Server supports both version-independent and version-dependent programmatic identifiers (ProgIDs). The root object can be referenced either as "CurrencyServer.Application" or as "CurrencyServer.Application.version" (e.g. "CurrencyServer.Application.5"). When version-dependent identifiers are used, a reference is created to a specific interface, and functionality introduced in newer versions may not be available.

Microsoft Visual C++ Native COM Support

Microsoft Visual C++ version 5.0 or later has native COM support through the "#import" statement. Use the following preprocessor line (assuming the default software installation path on an x64 system) to automatically add Currency Server type information:

```
#import "C:\Program Files (x86)\Cloanto\Currency Server\CURNSRV5.DLL"
```

The above creates classes corresponding to the Currency Server objects within the "CurrencyServer" namespace, with an "I" prefix and a "Ptr" suffix. For example, the "Application" object corresponds to the "CurrencyServer::IApplicationPtr" class.

Currency Server supports both version-independent (e.g. "CurrencyServer::IApplicationPtr") and version-dependent class references (e.g. "CurrencyServer::IApplication5Ptr"). Version-independent classes always refer to the to the most recent interfaces available at compile time, and are bound to the corresponding interface identifier (IID).

Currency Server Objects

The software exposes three levels of objects within the **CurrencyServer** namespace:

Application Object

ActiveCurrencies Object (Collection)

Currency Object

Admin Object

AllCurrencies Object (Collection)

Currency Object

Currencies are referenced by their ISO 4217 code in the various object methods and properties, with preference to the use of three-letter Alpha-3 strings (e.g. "EUR", "USD", "JPY", "GBP", etc.).

Locale strings may contain either a Windows locale identifier (LCID) code (e.g. "1033") or an RFC 4646 identifier (e.g. "en-US" or "EN"). The default locale is "1033" (US English).

Where objects are referenced by a numerical index value, the first index position has a value of 1. All properties are read-only. Once created, objects are static, which guarantees that they do not change while they are used. Optional arguments are enclosed in [square brackets].

Application Object

Methods:

Function **Convert(***FromCurrency As String, ToCurrency As String, Amount As Double, Rounding As Boolean, Format As String, Return As curncsrvReturnRate, [Time As String], [Type As String]***)** As String

Converts an amount from one currency to another currency, with optional rounding and formatting. Both FromCurrency and ToCurrency must be active currencies.

Either FromCurrency or ToCurrency (but not both) may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of conversion results, instead of a single result.

When rounding is enabled, the result is rounded to the nearest Smallest Unit for the current currency. Rounding is important for the results to take into consideration what the smallest unit is for each currency, for example some countries don't use cents or fractions, but just round numbers for monetary amounts. The rounding option only affects the result, not intermediate calculations.

The Return argument indicates whether the result should be returned as a string (curncsrvReturnRateString, or 1) or numerical (curncsrvReturnRateNumber, or 2) value. In order to not lose trailing 0s that may be useful for display purposes (e.g. 3.50 vs. 3.5), it is advised to request a string return value. Rounded numerical results never have trailing 0s.

The Format string, which is used only when returning string results, controls the formatting of the output string, i.e. the formatting of the value (decimal symbol, digit grouping symbol, etc.) and the optional currency code or symbol. The number of digits after the decimal symbol in rounded amounts (Rounding parameter set to True) is currency-dependent, and is set via the **Smallest Unit** field in the **All Currencies** tab of Currency Server Manager. The currency symbols and other default formatting parameters are locale-dependent (e.g. the name of a currency may be written differently in different languages, such as "Franc" vs. "Franken", even if it still refers to the same currency), and can be edited in the **Locales** tab. Please note that the locale code and the currency codes (used in FromCurrency and ToCurrency) are independent: there is usually one locale for each language used on your system or website, while the currency codes indicate the source and destination currencies of the desired conversion. The locale code is used for string formatting purposes only, and usually only slightly affects currency symbols, if at all (it never affects three-letter currency codes, which are the same all over the world). If you are using Currency Server on a website which interacts with the user in only one language, then you don't normally need to use multiple locales (just use the locale corresponding to the language of the website, e.g. US English, which is the default locale for Currency Server).

If the string consists of two or more letters or numbers, then the currency formatting of the locale matching the decimal or hexadecimal numerical code (the Windows Locale ID, or LCID, e.g. "1033" or "0x409" for US English) or the alphanumerical string (e.g. "ENU" for US English, "EN" for any English-language locale) is applied. Additionally, three special characters, i.e. "<", ">" and "&", are used to respectively insert the currency code (e.g. "USD") or the currency symbol (e.g. "$" or "US$"), and to use HTML character entity references instead of binary characters having decimal codes greater than 127 in the output

string (e.g. "£ 12.34" instead of "£ 12.34"). After these currency codes and special characters have been parsed, any additional characters are used to manually force certain decimal and digit grouping symbols. The first remaining character in the Format string (not counting optional currency codes and special characters) specifies the decimal symbol. The optional second character in the string (not counting optional currency codes and special characters) sets the optional digit grouping symbol. If these two characters are not defined in the Format string, then formatting is done using the currency formatting rules set for either the selected locale (if specified in the Format string) or the default locale (as specified in the **Locales** tab of Currency Server Manager).

If no locale is indicated, the Currency Server default locale is used. If a symbol is requested (">" option), but the locale has an undefined or empty symbol string for the currency, then the ISO 4217 Alpha-3 currency code (the same Code property which is also used by the "<" option) is returned.

If the string is set to "+", then no formatting is performed (the decimal point is used as a decimal symbol, with no other currency or digit grouping symbols or separators). The output string is guaranteed to consist of only numbers and a decimal point.

The following table shows different possible outputs for a hypothetical conversion resulting in an amount of 12345.67 in different currencies (ToCurrency parameter), using different format (Format parameter) options, and assuming a default Currency Server locale of US English.

Numerical Result	Format String	Output
USD 12345.67	""	"12,345.67"
USD 12345.67	"."	"12345.67"
USD 12345.67	". "	"12 345.67"
USD 12345.67	"+"	"12345.67"
USD 12345.67	"<enu.,"	"USD 12,345.67"
USD 12345.67	">enu.,"	"$12,345.67"
USD 12345.67	">1033.,"	"$12,345.67"
USD 12345.67	">"	"$12,345.67"
USD 12345.67	">nor"	"$ 12 345,67"
NOK 12345.67	">nor"	"kr 12 345,67"
NOK 12345.67	"<nor"	"NOK 12 345,67"
EUR 12345.67	">ita"	"€ 12.345,67"
EUR 12345.67	">&ita"	"€ 12.345,67"
EUR 12345.67	"<0x410"	"EUR 12.345,67"
EUR 12345.67	"<ita"	"EUR 12.345,67"

You can set FromCurrency and ToCurrency to the same currency unit in order to use this function for rounding and formatting purposes only (no conversion).

Note: if the result has more than 17 digits (e.g. a value of ten million billions without decimals) the double-precision number to string conversion logic used by this function may slightly round the string result. This is a limitation of the Microsoft libraries used by the software, which should not affect practical use. If this is unacceptable, request a numerical result and convert the result to a string using a different procedure.

The Time and Type string parameters are used for forward compatibility and should be omitted or left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **CountryToCurrency(***Country As String, ActiveOnly as Boolean***)** As String

Given a country code (ISO 3166 Alpha-2, Alpha-3 or numerical, e.g. "US", "USA", "840" or "0x348"), returns a "most likely" three-character currency code (ISO 4217 Alpha-3, e.g. "USD"). Since there is no exact relationship between countries and currencies (in some

countries more than one currency may be in practical use, e.g. during the transition from an old currency to a new currency, or because a "stronger" currency is preferred), the result may be slightly approximate (but correct in most cases). If the ActiveOnly flag is set, only currencies from the list of active currencies are considered. If no currency matches the country, the function returns a currency for the default locale set in Currency Server. If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code) is returned.

Function **CurrencyToCountry(***Currency As String, Return As curncsrvReturnCountry***)** As Variant

Returns a "most likely" country code (ISO 3166 Alpha-2, Alpha-3 or numerical, e.g. "US", "USA", or "840") associated to the specified currency (ISO 4217 Alpha-3). Since there is no exact relationship between currencies and countries (some currencies are in official use in more than one country), the result may be slightly approximate (but correct in most cases). Please note that the European Union only has a reserved ISO 3166 Alpha-2 code, but not a formal Alpha-3 or numerical code.

Possible values of Return: curncsrvReturnCountryAlpha2 (1), curncsrvReturnCountryAlpha3 (2), curncsrvReturnCountryNumerical (3), curncsrvReturnCountryOfficialName (4), curncsrvReturnCountryShortName (5).

Function **CurrencyToDomain(***Currency As String***)** As String

Returns a "most likely" top-level domain suffix (IANA TLD, e.g. "uk" or "com") associated to the specified currency code (ISO 4217 Alpha-3). Since there is no exact relationship between currencies and countries (some currencies are in official use in more than one country), and in many cases the "preferred" domain suffix in a country is not its official ccTLD, the result may be slightly approximate, or a generally-acceptable fallback.

Function **CurrencyToLocale(***Currency As String, Return As curncsrvReturnLocale***)** As Variant

Returns a "most likely" locale identifier (RFC 4646 or Windows LCID, e.g. "en-US" vs. "1033") associated to the specified currency code (ISO 4217 Alpha-3). Since not only there is no exact relationship between currencies and countries (some currencies are in official use in more than one country), but locales may span multiple countries, or only parts of a country, the result may be approximate, or a generally-acceptable fallback.

Possible values of Return: curncsrvReturnLocaleRFC4646 (1), curncsrvReturnLocaleLCID (2).

Function **DomainToCurrency(***Domain As String, ActiveOnly as Boolean***)** As String

Given an internet domain string (with or without protocol, username, password, directory and file information, e.g. "https://host.example.co.uk/directory/page.html", ".co.uk" and "uk" all produce the same result), returns a "most likely" currency code (e.g. "GBP"). If the ActiveOnly flag is set, only currencies from the list of active currencies are considered. Since there is no exact relationship between internet domains and countries and/or currencies (e.g. a currency or internet top level domain may be used in multiple countries), the result may be approximate. The function falls back to "EUR" for ".eu" domains, "USD" for ".mil" domains, and to the currency of the default locale set in Currency Server for non-national domains (e.g. ".com", ".int", ".edu", ".info", etc.) and for domains which could not otherwise be resolved to a country or region (e.g. IP addresses). If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code) is returned.

Function **Export(***Format As String, Encoding as String, BaseCurrency As String, Locale As String, Flags As String, ServiceLicenseKey As String, ServiceExpirationTime As String, ServiceRenewalURL As String, [ErrorCode As String], [ErrorMessage As String], [Time As String], [Type As String]***)** As String

This method exposes the output functionality used by post-update actions, generating exchange rates relative to BaseCurrency, and other currency properties. Rather than being written to a file or uploaded, as in the post-update actions, the data is returned as a string.

Possible values of Format: "INI" (default), "CSV", "TSV", "XML", "ECBCompatible", "JavaScript", "JSON", "JSONP". Some export formats may be further customized by specifying the name of a template file or JSONP callback function name, separated by a plus sign (e.g. "JavaScript+currencysystem5.js", "JSON+currencysystem5.json", "JSON +jsonCallback"). Template files must be stored in the "Templates" subdirectory of the Currency Server installation directory. If no template file is indicated, the default template

for that format is used. If no JSONP callback function name is specified (allowed characters: letters, numbers and underscore - if the name contains a period it identifies the string as a template file name), the default "jsonCurrencySystem" name is used.

Possible values of Encoding: "CP-1252" (default), "UTF-8-BOM", "UTF-8-NoBOM", "UTF-16-LittleEndian-BOM", "UTF-16-LittleEndian-NoBOM", "UTF-16-BigEndian-BOM", " UTF-16-BigEndian-NoBOM". Supported legacy values: "ANSI" (same as "CP-1252"), "UTF-8" (same as "UTF-8-BOM"), "UTF-16" and "Unicode" (same as "UTF-16-LittleEndian-BOM"), "BigEndianUnicode" (same as "UTF-16-BigEndian-BOM").

JavaScript, JSON and JSONP data is always output as ANSI, with string characters having a code greater than 127 being output as \u-escaped hexadecimal values.

The Flags string may be composed of the following individual attributes. For the parts that are omitted, or if the string is left empty, default values are used.

Flag	Description
a0	Container = None (default)
a1	Container = GNU Zip
C1	Country = True (ISO 3166 Alpha-2 code)
C2	Country = True (ISO 3166 Alpha-3 code)
C3	Country = True (ISO 3166 numerical)
C4	Country = True (official name)
C5	Country = True (short name)
c	Country = False (default)
D	DateOnly = True
d	DateOnly = False (default)
E1	Entity = True (official name)
E2	Entity = True (short name)
e	Entity = False (default)
G	Line ending = CR+LF (default)
g	Line ending = LF
H	ColumnHeaders = True (default)
h	ColumnHeaders = False
K	Rank = True
k	Rank = False (default)
L	LegalTender = True
l	LegalTender = False (default)
M	Symbol = True
m	Symbol = False (default)
N	CurrencyNames = True (default)
n	CurrencyNames = False
O	Popularity = True
o	Popularity = False (default)
P	Physical = True
p	Physical = False (default)
R	Regime = True
r	Regime = False (default)
S	SmallestUnit = True
s	SmallestUnit = False (default)
T	TitleStyle = True (default)
t	TitleStyle = False
Z	Time priority = Oldest

z	Time priority = Newest (default)
1	Export only level 1 active currencies
2	Export level 1 & 2 active currencies
3	Export all active currencies (default)

The ServiceLicenseKey, ServiceExpirationTime, ServiceRenewalURL, ErrorCode, and ErrorMessage properties, if non-empty, are passed through directly to the output data, without further processing. The corresponding XML tags are <licensekey>, <serviceexpiration>, and <servicerenewal>. If ErrorCode is specified, no currency data is exported, and the error code is included in the output file. If ErrorMessage is specified, no currency data is exported, and the error message is included in the output file.

The Time and Type string parameters are used for forward compatibility and should be omitted or left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **LocaleToCurrency(***Locale As String, ActiveOnly as Boolean***)** As String

Given a Windows locale identifier (LCID, e.g. "1033" or "0x409") or a three-character Windows locale code (e.g. "DEU", i.e. two-character ISO 639-1 Alpha-2 language codes plus an additional character, as specified for use with the Windows "LOCALE_SABBREVLANGNAME" type, which is not necessarily the same as the three-character ISO 639-2 Alpha-3 standard), or a two-letter ISO 639-2 Alpha-2 language code (e.g. "de"), returns a "most likely" currency code (ISO 4217 Alpha-3, e.g. "EUR"). If the ActiveOnly flag is set, only currencies from the list of active currencies are considered. Since there is no exact relationship between locales and currencies (e.g. a two-letter language locale code may match different countries with different currencies, and a three-letter code may match a country which is in a transition between two currencies), the result may be approximate (especially if two-letter language codes are used, whereas LCID codes or three-letter Windows locale codes lead to more focused results). If the input string is empty or could not be resolved, the function returns a currency for the default locale set in Currency Server. If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code) is returned.

Function **Privileges(***Privileges As curncsrvPrivilege***)** As Boolean

Returns a Boolean value indicating whether the specified access privileges to administrative functions are granted to COM or Web service clients.

Possible values of Privileges: curncsrvPrivilegeAdminCOM (1), curncsrvPrivilegeAdminWebService (2).

Function **Rate(***BaseCurrency As String, ToCurrency As String, Rounding As Boolean, Format As String, Return As curncsrvReturnRate, [Time As String], [Type As String]***)** As String

Returns the exchange (conversion) rate of ToCurrency, relative to BaseCurrency, with optional rounding and formatting. This method is equivalent to invoking Convert() with Amount set to 1. Both BaseCurrency and ToCurrency must be active currencies. The resulting value is both the amount of ToCurrency units for one unit of BaseCurrency and the conversion factor required to convert amounts expressed in BaseCurrency units to ToCurrency units (unless a special procedure such as triangulation is required).

Either BaseCurrency or ToCurrency (but not both) may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of rates, instead of a single rate.

The ReturnRate parameter indicates whether the result should be returned as a string or numerical value. In order to not lose trailing 0s that may be useful for display purposes (e.g. 3.50), it is advised to request a string return value. Rounded numerical results never have trailing 0s (e.g. 3.5).

Possible values of ReturnRate: curncsrvReturnRateString (1), curncsrvReturnRateNumber (2).

For example, Rate("USD", "EUR", False, "+", curncsrvReturnRateString) returns the euro amount corresponding to one US dollar, whereas Rate("EUR", "USD", False, "+", curncsrvReturnRateString) returns the dollar amount equivalent to one euro.

Applying rounding to exchange rate values (rather than results of conversions) is not recommended, so Rounding is usually best set to False. For additional information on rounding please refer to the documentation of the Convert() method.

For additional information on the Format parameter please refer to the documentation of the Convert() method.

Note: if the result has more than 17 digits (e.g. a value of ten million billions without decimals) the double-precision number to string conversion logic used by this function may slightly round the string result. This is a limitation of the Microsoft libraries used by the software, which should not affect practical use. If this is unacceptable, request a numerical result and convert the result to a string using a different procedure.

The Time and Type string parameters are used for forward compatibility and should be omitted or left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **Time(***Currencies As String, Information As curncsrvTimeInformation, Priority As curncsrvTimePriority, Return As curncsrvReturnTime, [Time As String], [Type As String]***)** As Variant

Returns time information about the specified currency or currencies, or the current time.

The Currencies parameter may contain a single currency, or a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result. The Currencies may also be left empty, if not applicable, or to indicate a request applied to the entire set of active currencies.

The Information parameter determines the request type: last modification, creation or expiration of the current exchange rate data, or subscription expiration of the FX feed(s) providing the data, or current time.

The Priority parameter indicates which time information ("newest" or "oldest") is to be returned if rates of different currencies (which may have had their rates last set or changed at different times) are combined to produce the desired result.

The ReturnTime parameter specifies whether the time should be returned as UTC (Universal Time, also referred to as "Zulu Time" or GMT), local time (server time), or seconds. UTC and local time strings are in the "YYYY-MM-DDThh:mm:ssTZD" format (e.g. "2008-12-31T14:00:59Z"), where TZD is the time zone designator ("Z", which stands for Universal Time, or "+hh:mm" or "-hh:mm"). The time in seconds is relative to midnight UTC of January 1, 1930.

Possible values of Information: curncsrvTimeRatesLastChange (1), curncsrvTimeRatesDataContent (2), curncsrvTimeRatesExpiration (3), curncsrvTimeServiceExpiration (4), curncsrvTimeCurrent (5), curncsrvTimeRatesEffective (6), curncsrvTimeRatesPublication (7).

Possible values of Priority: curncsrvTimeOldest (1), curncsrvTimeNewest (2).

Possible values of Return: curncsrvReturnTimeUT (1), curncsrvReturnTimeLocal (2), curncsrvReturnTimeSeconds (3).

Failed currency updates, and currency updates without changes to the specified currency or currencies do not change the internal time counters.

For most purposes, the effective time (curncsrvTimeRatesEffective) is the same as the publication time (curncsrvTimeRatesPublication), however for example for FX feeds that publish "next day" rates this information may differ. If no official publication time is known, then curncsrvTimeRatesPublication is set to be the same as curncsrvTimeRatesDataContent. If the data did not contain time information, then curncsrvTimeRatesDataContent is the collection time.

For currencies at a constant exchange or conversion rate (e.g. if ToCurrency is a subunit of BaseCurrency at a constant rate under a regime such as the EMU), the current date and time are used if no other information is available about the effective date of the constant rate. If one of a pair of currencies is the base unit used by the FX feed to provide the rate of the other currency, then the effective date of that rate is returned.

If the information is not available, for example if rates have never been set or changed, or

if no expiration time is available, an empty string is returned. If the result was requested in seconds (numerical return value), the undefined case is returned as 0x7FFFFFFF (decimal 2147483647, C language LONG_MAX definition).

The Time and Type string parameters are used for forward compatibility and should be omitted or left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **Validate(***LicenseKey As String***)** As curncsrvValidateResult

Indicates whether the LicenseKey parameter meets the client license key validation requirements. This procedure is normally performed to validate Web service clients, and is not required for accessing the COM interface.

The LicenseKey parameter is validated by Currency Server and/or by external validation code. The optional external validation procedure (which for example can query a database to verify that a client license is both active and within a daily quota limit) can be set in the **Clients** tab.

In the Enterprise Edition of Currency Server, or if the Unlimited Web Service Clients add-on component is installed, Currency Server itself (opposed to the external validation module) does not check the license key, which does not have to be a valid Currency Server client license key, but can be blank or in any format suitable to be processed by the (optional) external validation procedure.

For security reasons, a license key passed to an external validation executable must have a maximum size of 254 characters, and may not contain the "\", "/", ":" and """ (double quote) characters. Currency Server normalizes the license key so that it complies with these requirements.

Possible values of curncsrvValidateResult: curncsrvValidateSucceeded (-1), curncsrvValidateNoSubscription (0), curncsrvValidateExpiredSubscription (1), curncsrvValidateHitExcess (2), curncsrvValidateToActivate (3), curncsrvValidateLostOrStolen (4), curncsrvValidateBlocked (5).

For additional information, see Client Validation Modules.

Properties:

Property **ActiveCurrencies** As ActiveCurrencies

Returns the ActiveCurrencies collection object. The object contains a snapshot of the currency list (which is not updated during the lifetime of the object).

Property **Admin** As Admin

Returns the Admin object.

Property **AllCurrencies** As ActiveCurrencies

Returns the ActiveCurrencies collection object. The object contains a snapshot of the currency list (which is not updated during the lifetime of the object).

Property **Application** As Object

Returns the Application object.

Property **Parent** As Object

Returns the parent object.

Property **Separator** As String

Returns the separator character used by some of the interface methods. By default it is a semicolon (";").

Property **Version** As String

Returns the software version.

ActiveCurrencies Object (Collection)

Methods:

Function **Find(***Currency As String***)** As Long

Returns the index of the currency (three-letter Alpha-3 short name or numerical code as per ISO 4217) in the collection of active currencies, or 0 if the currency does not exist.

Properties:

Property **Application** As Object

Returns the Application object.

Property **Copyright** As String

Returns the copyright information provided by the FX feed(s), if any. If data was collected from multiple servers, a semicolon is used to separate information provided by the different servers.

Property **Count** As Long

Returns the number of currencies in the collection.

Property **Item(***Index As Long***)** As Currency

Returns an item (Currency object) of the collection. The first object has an index value of 1.

Because this is the default property of the AllCurrencies object, some programming languages (e.g. Visual Basic and VBScript) allow items to be referenced using a short notation such as "Application.ActiveCurrencies(1)", equivalent to "Application.ActiveCurrencies.Item(1)".

Property **Message** As String

Returns the message(s) from the FX feed(s), if any. If data was collected from multiple servers, a semicolon is used to separate information provided by the different servers.

Property **Parent** As Object

Returns the parent object.

Property **Source** As String

Returns the name(s) of the FX feed(s). If data was collected from multiple servers, a semicolon is used to separate information provided by the different servers.

Property **_NewEnum** As Object

Returns the collection enumerator. This property is restricted and not displayed by object browsers. Some programming languages use it to enumerate the collection (e.g. the "for ... each" mechanism in Visual Basic and VBScript).

Admin Object

Methods:

Sub **FXFeed(***FeedID As String, URL As String, UserID As String, UserName As String, Password As String, Action As curncsrvFXFeedAction***)**

Currency Server collects exchange rates from one or more FX feeds. This method sets a new feed, whereby the Action parameter specifies whether the feed should be added to the existing feeds, or whether it should replace existing feeds (effectively deleting all previous feed entries, and leaving only the new feed). It is not possible to delete the last feed (there must always be at least one feed). The Standard Edition of Currency Server only supports

fetching data from a single feed at a time (trying to use the add operation will fail).

The FeedID parameter is the FX feed object identifier (OID), as it appears in the **Edit FX Feed** dialog (being set in the FX feed filter).

The other parameters are as documented for the **FX Feeds** tab.

Possible values of Action: curncsrvFXFeedActionAdd (1), curncsrvFXFeedActionReplace (2).

This administrative function is disabled by default to avoid undesired use.

Sub **FXMode(***Multiple As curncsrvFXModeMultiple, Lock As curncsrvFXModeLock***)**

Specifies the operating mode when multiple FX feeds are used (merge, cross-check or failover), as documented for the **FX Feeds** tab.

The LockCurrencies parameter indicates whether exchange rate updates may not only update the rates, but also add or remove currencies, as documented for the **Active Currencies** tab.

Possible values of Multiple: curncsrvFXModeMultipleMerge (1), curncsrvFXModeMultipleCrossCheck (2), curncsrvFXModeMultipleFailover (3).

Possible values of Lock: curncsrvFXModeLockAll (1), curncsrvFXModeLockEMU (2), curncsrvFXModeFullRefresh (3).

This administrative function is disabled by default to avoid undesired use.

Sub **Load(***Path As String***)**

Loads the specified Currency Server configuration file. The data is applied immediately, overwriting all Currency Server settings, including exchange rate data, but excluding setup information such as installation details (e.g. software installation directory, web service installation path) and software license information (user name, organization name, license keys). If the file name is an empty string a file dialog is opened. If the file name does not end with ".cs-cfg", the suffix is appended by the software.

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default to avoid undesired use.

Sub **Message(***Message As String, Type As curncsrvMessageType***)**

Sends an information, warning or error message as specified in the notification options (the message is displayed and/or emailed and/or stored in the application log).

Possible values of Type: curncsrvMessageInformation (1), curncsrvMessageWarning (2), curncsrvMessageError (3).

Note: the message is always asynchronous, i.e. the call returns immediately.

This administrative function is disabled by default to avoid undesired use.

Sub **Reset(***Flags As curncsrvResetFlags***)**

Resets the entire Currency Server configuration to default values. The data is applied immediately, overwriting all Currency Server settings, including exchange rate data, but excluding setup information such as installation details (e.g. software installation directory, web service installation path) and software license information (user name, organization name, license keys).

The Flags parameter is used for forward compatibility and should be set to curncsrvResetFlagDefault (0).

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default to avoid undesired use.

Sub **Save(***Path As String***)**

Saves the current software state (Currency Server configuration and exchange rate data

values) to the specified file. If the file name is an empty string a file dialog is opened. If the file name does not end with ".cs-cfg", the suffix is appended by the software.

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default to avoid undesired use.

Sub **UpdateNow()**

Establishes a connection to the FX feed(s) and acquires new currency data.

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default to avoid undesired use.

Properties:

Property **Application** As Object

Returns the Application object.

Property **Parent** As Object

Returns the parent object.

AllCurrencies Object (Collection)

Methods:

Function **Find(***Currency As String***)** As Long

Returns the index of the currency (three-letter Alpha-3 short name or numerical code as per ISO 4217) in the collection of all currencies, or 0 if the currency does not exist.

Properties:

Property **Application** As Object

Returns the Application object.

Property **Count** As Long

Returns the number of currencies in the collection.

Property **Item(***Index As Long***)** As Currency

Returns an item (Currency object) of the collection. The first object has an index value of 1.

Because this is the default property of the AllCurrencies object, some programming languages (e.g. Visual Basic and VBScript) allow items to be referenced using a short notation such as "Application.AllCurrencies(1)", equivalent to "Application.AllCurrencies.Item(1)".

Property **_NewEnum** As Object

Returns the collection enumerator. This property is restricted and not displayed by object browsers. Some programming languages use it to enumerate the collection (e.g. the "for ... each" mechanism in Visual Basic and VBScript).

Property **Parent** As Object

Returns the parent object.

Currency Object

Properties:

Property **Application** As Object

Returns the Application object.

Property **BaseUnit** As String

Returns the code (three letters as per ISO 4217 Alpha-3) of the base currency unit relative to which the internal reference rate (Rate property) is expressed.

Property **Code** As String

Returns the code (three letters as per ISO 4217 Alpha-3) of the currency.

Property **Custom(***Locale As String***)** As String

Returns the custom string associated with the currency in the selected locale (e.g. "usd.gif" to indicate the location of a flag image file). Please note that custom strings may use Unicode (UTF-16) text encoding.

If the Locale string is empty, the Currency Server default locale is used.

Property **Entity(***Locale As String, Denomination As curncsrvEntityDenomination, TitleStyle as Boolean***)** As String

Returns the name of the entity associated to the currency, based on the specified locale, denomination (official or short, e.g. "People's Republic of China" vs. "China") and title style. Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the entity names when certain Asian and certain other locales are used. The default entity names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Possible values of Denomination: curncsrvEntityOfficial (1), curncsrvEntityShort (2).

Property **Feeds** As Long

Returns the number of FX feeds (from those listed in the FX Feeds tab) providing exchange rate data for the currency.

Property **Hits** As Long

Returns the number of times this currency has been referenced by a COM object or by the Web service. This property can be reset to 0 by clicking **Reset Hits** in the **Active Currencies** tab of Currency Server Manager.

Property **LegalTender** As Boolean

Returns a Boolean value indicating whether the currency is legal tender. For example, after a currency is replaced by the currency of the monetary union which it previously joined, the original currency may cease to be legal tender. For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.

Property **Name(***Locale As String, TitleStyle as Boolean***)** As String

Returns the extended name of the currency in the given locale (e.g. "Swiss franc", "Schweizer Franken", "Franco svizzero", etc.) and capitalization style (e.g. "US dollar" vs. "US Dollar"). Currency names are usually written in lower case as a rule. In some languages (e.g. German) all nouns, including currency names, are always capitalized (both in body text and in titles). In some languages (e.g. English) titles are capitalized, which style may also be appropriate in drop-down lists, etc. For example, the Currency Server user interface uses title style in most contexts. This property returns the proper capitalization based on the locale and context (title or body text). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the currency names when certain Asian and certain other locales are used. The default currency names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Property **Parent** As Object

Returns the parent object.

Property **Physical** As Boolean

Returns a Boolean value indicating whether the currency is a physical currency (i.e. "real" notes and coins, opposed to units of account or virtual currencies).

Function **Popularity** As Long

Returns the currency popularity value (1 = high, 2 = medium, 3 = low). The value is the inverse of the Currency Rank (1 = low, 2 = medium, 3 = high).

Property **Rate** As String

Returns the internal reference rate of the currency (relative to 1 BaseUnit), as provided by the original FX feed. Trailing zero characters are used to indicate the precision of the original data (e.g. 123.4560 vs. 123.456).

When a currency is manually added for the first time to the list of active currencies, it has an undefined rate (Rate = 0) until the rate is provided by a FX feed or entered manually.

Note: It is not recommended to access the Rate property directly, because it is relative to the BaseUnit property, which is not controlled by your application, but rather it is set when Currency Server gets the data from the FX feed. If you need to get the rate of a currency relative to a given base currency, use the Rate() method of the Application object.

Property **Regime** As String

Returns the relevant identifier if the currency is a member of a regime or monetary union, or an empty string if the currency is not a member of a regime or monetary union. The identifier for currencies which are part of the European Economic and Monetary Union (EMU) is "EMU". The identifier for non-EMU currencies which are pegged to the euro at a constant rate is "Pegged-EUR". This information may be used as a condition to apply exchange commissions only when necessary (e.g. because conversions between currencies which are member of the same regime may not be subject to commissions or credit card surcharges). This property currently only returns "EMU", "Pegged-EUR" or an empty string. For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.

Property **SmallestUnit** As String

Returns the smallest unit for the currency. Trailing zero characters are used to specify the preferred format for rounding (e.g. 0.50 vs. 0.5).

Function **Subunit(***Locale As String, Inflection As curncsrvSubunitInflection, TitleStyle as Boolean***)** As String

Returns the subunit name, based on the specified locale, inflection (singular or plural, e.g. "cent" vs. "cents") and title style (e.g. "cents" vs, "Cents"). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the subunit names when certain Asian and certain other locales are used. The default subunit names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Possible values of Inflection: curncsrvSubunitSingular (1), curncsrvSubunitPlural (2).

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Property **Symbol(***Locale As String***)** As String

Returns the symbol string associated with the currency (e.g. "$" or "US$", as set for the selected locale). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the currency symbols of some Asian and certain other currencies, regardless of the locale. The default currency symbols preset for the US English locale use Unicode characters for some Asian and certain other currencies. If your application does not support Unicode, you may want to edit these currency strings in the properties of the locale(s) you plan to use.

If the Locale string is empty, the Currency Server default locale is used.

Property **Triangulation** As Boolean

Returns True if the currency is a subunit (or "pegged currency") of BaseUnit at a constant rate (Rate property relative to 1 BaseUnit) and requires triangulation (an intermediate conversion to BaseUnit) for conversions to and from the currency. This property is currently only True if Regime = "EMU" or "Pegged-EUR".

Function **Unit(***Locale As String, Inflection As curncsrvUnitInflection, TitleStyle as Boolean***)** As String

Returns the unit name, based on the specified locale, inflection (singular or plural, e.g. "dollar" vs. "dollars") and title style (e.g. "dollars" vs. "Dollars"). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the unit names when certain Asian and certain other locales are used. The default unit names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Possible values of Inflection: curncsrvUnitSingular (1), curncsrvUnitPlural (2).

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Property **Warning** As Boolean

Returns a Boolean value indicating whether the currency has been referenced by a recent warning condition. Possible causes of warnings include excessive fluctuation, extended lack of updates, no data from one or more servers, conflicting data from different servers, FX feed rates conflicting with official EMU (European Economic and Monetary Union) rates, etc. This property can be reset to False by clicking **Reset Warnings** in the **Active Currencies** tab of Currency Server Manager.

Related Topics

- For cross-version compatibility information, see Legacy Support.
- For examples of programmatic access to the Currency Server methods and properties, see Code Examples.
- For more information about the different interfaces which can be used to access and control Currency Server, see Accessing the Software.

6.2 The .NET Web Service Interface

Overview

Building on the SOAP communications protocol, Microsoft created .NET as its platform for XML Web services. The Currency Server Web service is implemented as a .NET server written in managed C++ code, independent of the COM interface.

Installation

The .NET-based Web service, which incorporates the functionality previously provided by a stand-alone SOAP interface, is an optional feature of the Currency Server installer, and can be installed, uninstalled or reinstalled at any time, without interfering with the configuration of Currency Server. .NET-client access to administrative functions is disabled by default, and can be enabled through a setting in the **Clients** tab of Currency Server Manager.

When the Web service is installed, and if the default settings are used, a "CurrencyServer" virtual directory is created in the default website. The web service directory contains a versioned Web service access point file, named "CurrencyServer5.asmx". This directory is also used by the SOAP interface. You can refer to the Software tab of Currency Server Manager to check whether the Web service is installed, in which case you can click the "Web service: Installed" link to open the WSDL file and interact with the service. By default, the address is http://localhost/

CurrencyServer/CurrencyServer5.asmx (append "?WSDL" to open the WSDL file).

License Key Parameter

The LicenseKey parameter is used by all methods, and is validated by Currency Server and/or by external validation code. An optional external validation procedure (which for example can query a database to verify that a client license is both active and within a daily quota limit) can be set in the **Clients** tab.

In the Enterprise Edition of Currency Server, or if the Unlimited Web Service Clients add-on component is installed, the license key may be blank or in any format suitable to be processed by the (optional) external validation procedure. Currency Server does not process the license key, so it does not need to be a valid Currency Server client license key. Only the external validation code processes the key. In the Standard Edition of Currency Server without add-ons, the license key is validated by Currency Server, and must be a valid Currency Server client access key.

For security reasons, a license key passed to an external validation executable must have a maximum size of 254 characters, and may not contain the "\", "/", ":" and """ (double quote) characters. Currency Server normalizes the license key before passing it on to the validation module, if any, so that it complies with these requirements.

Currency Server Objects

The methods exposed by the Web service are equivalent to the methods and properties of the COM interface. The structure of the objects has been flattened and optimized to reduce overhead and in consideration of the stateless nature of the SOAP communications protocol.

Currencies are referenced by their ISO 4217 code in the various object methods and properties, with preference to the use of three-letter Alpha-3 strings (e.g. "EUR", "USD", "JPY", "GBP", etc.).

Locale strings may contain either a Windows locale identifier (LCID) code (e.g. "1033") or an RFC 4646 identifier (e.g. "en-US" or "EN"). The default locale is "1033" (US English).

The following Web service operations are functionally equivalent to the corresponding methods and properties in the CurrencyServer Application, Admin, ActiveCurrencies, AllCurrencies and Currency COM objects.

Methods:

Function **ActiveCurrencies(***LicenseKey As String***)** As String

> Returns a semicolon-separated list of all active currencies (using three-letter codes as per ISO 4217 Alpha-3).

Sub **AdminFXFeed(***LicenseKey As String, FeedID As String, URL As String, UserID As String, UserName As String, Password As String, Action As Enum***)**

> Currency Server collects exchange rates from one or more FX feeds. This method sets a new feed, whereby the Action parameter specifies whether the feed should be added to the existing feeds, or whether it should replace existing feeds (effectively deleting all previous feed entries, and leaving only the new feed). It is not possible to delete the last feed (there must always be at least one feed). The Standard Edition of Currency Server only supports fetching data from a single feed at a time (trying to use the add operation will fail).
>
> The FeedID parameter is the FX feed object identifier (OID), as it appears in the **Edit FX Feed** dialog (being set in the FX feed filter).
>
> The other parameters are as documented for the **FX Feeds** tab.
>
> Possible values of Action: curncsrvFXFeedActionAdd (1), curncsrvFXFeedActionReplace (2).
>
> This administrative function is disabled by default, to avoid improper use when Currency Server is deployed as a public web service.

Sub **AdminFXMode(***LicenseKey As String, Multiple As Enum, LockCurrencies As Enum***)**

> Specifies the operating mode when multiple FX feeds are used (merge, cross-check or failover), as documented for the **FX Feeds** tab.
>
> The LockCurrencies parameter indicates whether exchange rate updates may not only update the rates, but also add or remove currencies, as documented for the **Active**

Currencies tab.

Possible values of Multiple: curncsrvFXModeMultipleMerge (1), curncsrvFXModeMultipleCrossCheck (2), curncsrvFXModeMultipleFailover (3).

Possible values of LockCurrencies: curncsrvFXModeLockAll (1), curncsrvFXModeLockEMU (2), curncsrvFXModeFullRefresh (3).

This administrative function is disabled by default, to avoid improper use when Currency Server is deployed as a public web service.

Sub **AdminLoad(***LicenseKey As String, File As String***)**

Loads the specified Currency Server configuration file. The data is applied immediately, overwriting all Currency Server settings, including exchange rate data, but excluding setup information such as installation details (e.g. software installation directory, web service installation path) and software license information (user name, organization name, license keys). If the file name is an empty string a file dialog is opened. If the file name does not end with ".cs-cfg", the suffix is appended by the software.

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default, to avoid improper use when Currency Server is deployed as a public web service.

Sub **AdminMessage(***LicenseKey As String, Text As String, Type As Enum***)**

Sends an information, warning or error message as specified in the notification options (the message is displayed and/or emailed and/or stored in the application log).

Possible values of Type: curncsrvMessageInformation (1), curncsrvMessageWarning (2), curncsrvMessageError (3).

This administrative function is disabled by default, to avoid improper use when Currency Server is deployed as a public web service.

Sub **AdminReset(***LicenseKey As String, Flags As String***)**

Resets the entire Currency Server configuration to default values. The data is applied immediately, overwriting all Currency Server settings, including exchange rate data, but excluding setup information such as installation details (e.g. software installation directory, web service installation path) and software license information (user name, organization name, license keys).

The Flags parameter is used for forward compatibility and should be left empty.

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default, to avoid improper use when Currency Server is deployed as a public web service.

Sub **AdminSave(***LicenseKey As String, File As String***)**

Saves the current software state (Currency Server configuration and exchange rate data values) to the specified file. If the file name is an empty string a file dialog is opened. If the file name does not end with ".cs-cfg", the suffix is appended by the software.

This method returns an error if Currency Server Manager is open (when in use, Currency Server Manager gives the operator exclusive write access to the currency information).

This administrative function is disabled by default, to avoid improper use when Currency Server is deployed as a public web service.

Sub **AdminUpdateNow(***LicenseKey As String***)**

Requests that Currency Server establish a connection to the FX feed(s) and acquire new currency data.

This administrative function is disabled by default, to avoid improper use when Currency

Server is deployed as a public web service.

Function **AllCurrencies(***LicenseKey As String***)** As String

Returns a semicolon-separated list of all currencies (using three-letter codes as per ISO 4217 Alpha-3). For most practical purposes involving exchange rates or conversions, ActiveCurrencies() should be used instead.

Function **Convert(***LicenseKey As String, FromCurrency As String, ToCurrency As String, Amount As Double, Rounding As Boolean, Format As String, ReturnRate As Enum, Time As String, Type As String***)** As String

Converts an amount from one currency to another currency, with optional rounding and formatting. Both FromCurrency and ToCurrency must be active currencies.

Either FromCurrency or ToCurrency (but not both) may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of conversion results, instead of a single result.

When rounding is enabled, the result is rounded to the nearest Smallest Unit for the current currency. Rounding is important for the results to take into consideration what the smallest unit is for each currency, for example some countries don't use cents or fractions, but just round numbers for monetary amounts. The rounding option only affects the result, not intermediate calculations.

The ReturnRate parameter indicates whether the result should be returned as a string or numerical value. In order to not lose trailing 0s that may be useful for display purposes (e.g. 3.50 vs. 3.5), it is advised to request a string return value. Rounded numerical results never have trailing 0s.

Possible values of ReturnRate: curncsrvReturnRateString (1), curncsrvReturnRateNumber (2).

The Format string, which is used only when returning string results, specifies the decimal symbol (first character in the string) and an optional digit grouping symbol (optional second character in the string). If the string is empty (no characters), formatting is done using the system default symbols (as specified in Control Panel, Regional Options, Currency).

The Format string controls the formatting of the output string, i.e. the formatting of the value (decimal symbol, digit grouping symbol, etc.) and the optional currency code or symbol. The number of digits after the decimal symbol in rounded amounts (Rounding parameter set to True) is currency-dependent, and is set via the **Smallest Unit** field in the **All Currencies** tab of Currency Server Manager. The currency symbols and other default formatting parameters are locale-dependent (e.g. the name of a currency may be written differently in different languages, such as "Franc" vs. "Franken", even if it still refers to the same currency), and can be edited in the **Locales** tab. Please note that the locale code and the currency codes (used in FromCurrency and ToCurrency) are independent: there is usually one locale for each language used on your system or website, while the currency codes indicate the source and destination currencies of the desired conversion. The locale code is used for string formatting purposes only, and usually only slightly affects currency symbols, if at all (it never affects three-letter currency codes, which are the same all over the world). If you are using Currency Server on a website which interacts with the user in only one language, then you don't normally need to use multiple locales (just use the locale corresponding to the language of the website, e.g. US English, which is the default locale for Currency Server).

If the string consists of two or more letters or numbers, then the currency formatting of the locale matching the decimal or hexadecimal numerical code (the Windows Locale ID, or LCID, e.g. "1033" or "0x409" for US English) or the alphanumerical string (e.g. "ENU" for US English, "EN" for any English-language locale) is applied. Additionally, three special characters, i.e. "<", ">" and "&", are used to respectively insert the currency code (e.g. "USD") or the currency symbol (e.g. "$" or "US$"), and to use HTML character entity references instead of binary characters having decimal codes greater than 127 in the output string (e.g. "£ 12.34" instead of "£ 12.34"). After these currency codes and special characters have been parsed, any additional characters are used to manually force certain decimal and digit grouping symbols. The first remaining character in the Format string (not counting optional currency codes and special characters) specifies the decimal symbol. The optional second character in the string (not counting optional currency codes and special characters) sets the optional digit grouping symbol. If these two characters are not defined in the Format string, then formatting is done using the currency formatting rules set for

either the selected locale (if specified in the Format string) or the default locale (as specified in the **Locales** tab of Currency Server Manager).

If no locale is indicated, the Currency Server default locale is used. If a symbol is requested (">" option), but the locale has an undefined or empty symbol string for the currency, then the ISO 4217 Alpha-3 currency code (the same Code property which is also used by the "<" option) is returned.

If the string is set to "+", then no formatting is performed (the decimal point is used as a decimal symbol, with no other currency or digit grouping symbols or separators). The output string is guaranteed to consist of only numbers and a decimal point.

The following table shows different possible outputs for a hypothetical conversion resulting in an amount of 12345.67 in different currencies (ToCurrency parameter), using different format (Format parameter) options, and assuming a default Currency Server locale of US English.

Numerical Result	Format String	Output
USD 12345.67	""	"12,345.67"
USD 12345.67	"."	"12345.67"
USD 12345.67	". "	"12 345.67"
USD 12345.67	"+"	"12345.67"
USD 12345.67	"<enu.,"	"USD 12,345.67"
USD 12345.67	">enu.,"	"$12,345.67"
USD 12345.67	">1033.,"	"$12,345.67"
USD 12345.67	">"	"$12,345.67"
USD 12345.67	">nor"	"$ 12 345,67"
NOK 12345.67	">nor"	"kr 12 345,67"
NOK 12345.67	"<nor"	"NOK 12 345,67"
EUR 12345.67	">ita"	"€ 12.345,67"
EUR 12345.67	">&ita"	"€ 12.345,67"
EUR 12345.67	"<0x410"	"EUR 12.345,67"
EUR 12345.67	"<ita"	"EUR 12.345,67"

You can set FromCurrency and ToCurrency to the same currency unit in order to use this function for rounding and formatting purposes only (no conversion).

Note: if the result has more than 17 digits (e.g. a value of ten million billions without decimals) the double-precision number to string conversion logic used by this function may slightly round the string result. This is a limitation of the Microsoft libraries used by the software, which should not affect practical use. If this is unacceptable, request a numerical result and convert the result to a string using a different procedure.

The Time and Type string parameters are used for forward compatibility and should be left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **Copyright(***LicenseKey As String***)** As String

Returns the copyright information provided by the FX feed(s), if any. If data was collected from multiple servers, a semicolon is used to separate information provided by the different servers.

Function **CountryToCurrency(***LicenseKey As String, Country As String, ActiveOnly as Boolean***)** As String

Given a country code (ISO 3166 Alpha-2, Alpha-3 or numerical, e.g. "US", "USA", "840" or "0x348"), returns a "most likely" three-character currency code (ISO 4217 Alpha-3, e.g. "USD"). Since there is no exact relationship between countries and currencies (in some countries more than one currency may be in practical use, e.g. during the transition from an old currency to a new currency, or because a "stronger" currency is preferred), the

result may be slightly approximate (but correct in most cases). If the ActiveOnly flag is set, only currencies from the list of active currencies are considered. If no currency matches the country, the function returns a currency for the default locale set in Currency Server. If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 three-letter code) is returned.

Function **CurrencyCustom(***LicenseKey As String, Currency As String, Locale As String***)** As String

Returns the custom string associated with the currency in the selected locale (e.g. "usd.gif" to indicate the location of a flag image file). Please note that custom strings may use Unicode (UTF-16) text encoding.

If the Locale string is empty, the Currency Server default locale is used.

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencyEntity(***LicenseKey As String, Currency As String, Locale As String, Denomination As Enum, TitleStyle as Boolean***)** As String

Returns the name of the entity associated to the currency, based on the specified locale, denomination (official or short, e.g. "People's Republic of China" vs. "China") and title style. Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the entity names when certain Asian and certain other locales are used. The default entity names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Possible values of Denomination: curncsrvEntityOfficial (1), curncsrvEntityShort (2).

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencyExists(***LicenseKey As String, Currency As String, ActiveOnly As Boolean***)** As Boolean

Returns a Boolean value indicating whether the currency is included in the list of active currencies or of all currencies.

Function **CurrencyFeeds(***LicenseKey As String, Currency As String***)** As Long

Returns the number of FX feeds (from those listed in the FX Feeds tab) providing exchange rate data for the currency.

Function **CurrencyHits(***LicenseKey As String, Currency As String***)** As Long

Returns the number of times this currency has been referenced by a COM object or by the Web service. This property can be reset to 0 by clicking **Reset Hits** in the **Active Currencies** tab of Currency Server Manager.

Function **CurrencyLegalTender(***LicenseKey As String, Currency As String***)** As Boolean

Returns a Boolean value indicating whether the currency is legal tender. For example, after a currency is replaced by the currency of the monetary union which it previously joined, the original currency may cease to be legal tender. For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.

Function **CurrencyName(***LicenseKey As String, Currency As String, Locale As String, TitleStyle as Boolean***)** As String

Returns the extended name of the currency in the given locale (e.g. "Swiss franc", "Schweizer Franken", "Franco svizzero", etc.) and capitalization style (e.g. "US dollar" vs. "US Dollar"). Currency names are usually written in lower case as a rule. In some languages (e.g. German) all nouns, including currency names, are always capitalized (both in body text and in titles). In some languages (e.g. English) titles are capitalized, which style may also be appropriate in drop-down lists, etc. For example, the Currency Server user interface uses title style in most contexts. This method returns the proper capitalization

based on the locale and context (title or body text). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the currency names when certain Asian and certain other locales are used. The default currency names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencyPhysical(***LicenseKey As String, Currency As String***)** As Boolean

Returns a Boolean value indicating whether the currency is a physical currency (i.e. "real" notes and coins, opposed to units of account or virtual currencies).

Function **CurrencyPopularity(***LicenseKey As String, Currency As String***)** As Long

Returns the currency popularity value (1 = high, 2 = medium, 3 = low). The value is the inverse of the Currency Rank (1 = low, 2 = medium, 3 = high).

Function **CurrencyRegime(***LicenseKey As String, Currency As String***)** As String

Returns the relevant identifier if the currency is a member of a regime or monetary union, or an empty string if the currency is not a member of a regime or monetary union. The identifier for currencies which are part of the European Economic and Monetary Union (EMU) is "EMU". The identifier for non-EMU currencies which are pegged to the euro at a constant rate is "Pegged-EUR". This information may be used as a condition to apply exchange commissions only when necessary (e.g. because conversions between currencies which are member of the same regime may not be subject to commissions or credit card surcharges). This property currently only returns "EMU", "Pegged-EUR" or an empty string. For more information about the currencies of the European Economic and Monetary Union, see The EMU and the Euro.

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencySmallestUnit(***LicenseKey As String, Currency As String***)** As String

Returns the smallest unit for the currency. Trailing zero characters are used to specify the preferred format for rounding (e.g. 0.50 vs. 0.5).

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencySubunit(***LicenseKey As String, Currency As String, Locale As String, Inflection As Enum, TitleStyle as Boolean***)** As String

Returns the subunit name, based on the specified locale, inflection (singular or plural, e.g. "cent" vs. "cents") and title style (e.g. "cents" vs, "Cents"). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the subunit names when certain Asian and certain other locales are used. The default subunit names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Possible values of Inflection: curncsrvSubunitSingular (1), curncsrvSubunitPlural (2).

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencySymbol(***LicenseKey As String, Currency As String, Locale As String***)** As String

Returns the symbol string associated with the currency (e.g. "$" or "US$", as set for the selected locale). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the currency symbols of some Asian and certain other currencies, regardless of the locale (i.e. even in US English).

The default currency symbols preset for the US English locale use Unicode characters for some Asian and certain other currencies. If your application does not support Unicode, you may want to edit these currency strings in the properties of the locale(s) you plan to use.

If the Locale string is empty, the Currency Server default locale is used.

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencyToCountry(***LicenseKey As String, Currency As String, ReturnCountry As Enum***)** As String

Returns a "most likely" country code (ISO 3166 Alpha-2, Alpha-3 or numerical, e.g. "US", "USA", or "840") associated to the specified currency (ISO 4217 Alpha-3). Since there is no exact relationship between currencies and countries (some currencies are in official use in more than one country), the result may be slightly approximate (but correct in most cases). Please note that the European Union only has a reserved ISO 3166 Alpha-2 code, but not a formal Alpha-3 or numerical code.

Possible values of ReturnCountry: curncsrvReturnCountryAlpha2 (1), curncsrvReturnCountryAlpha3 (2), curncsrvReturnCountryNumerical (3), curncsrvReturnCountryOfficialName (4), curncsrvReturnCountryShortName (5).

Function **CurrencyToDomain(***LicenseKey As String, Currency As String***)** As String

Returns a "most likely" top-level domain suffix (IANA TLD, e.g. "uk" or "com") associated to the specified currency code (ISO 4217 Alpha-3). Since there is no exact relationship between currencies and countries (some currencies are in official use in more than one country), and in many cases the "preferred" domain suffix in a country is not its official ccTLD, the result may be slightly approximate, or a generally-acceptable fallback.

Function **CurrencyToLocale(***LicenseKey As String, Currency As String, ReturnLocale As Enum***)** As String

Returns a "most likely" locale identifier (RFC 4646 or Windows LCID, e.g. "en-US" vs. "1033") associated to the specified currency code (ISO 4217 Alpha-3). Since not only there is no exact relationship between currencies and countries (some currencies are in official use in more than one country), but locales may span multiple countries, or only parts of a country, the result may be approximate, or a generally-acceptable fallback.

Possible values of ReturnLocale: curncsrvReturnLocaleRFC4646 (1), curncsrvReturnLocaleLCID (2).

Function **CurrencyTriangulation(***LicenseKey As String, Currency As String***)** As Boolean

Returns True if the currency is a subunit (or "pegged currency") of BaseUnit at a constant rate (Rate property relative to 1 BaseUnit) and requires triangulation (an intermediate conversion to BaseUnit) for conversions to and from the currency. This property is currently only True if Regime = "EMU" or "Pegged-EUR".

Function **CurrencyUnit(***LicenseKey As String, Currency As String, Locale As String, Inflection As Enum, TitleStyle as Boolean***)** As String

Returns the unit name, based on the specified locale, inflection (singular or plural, e.g. "dollar" vs. "dollars") and title style (e.g. "dollars" vs. "Dollars"). Please note that Unicode (UTF-16) text characters with no equivalent character in CP-1252 (Latin 1) 8-bit encoding may be present in the unit names when certain Asian and certain other locales are used. The default unit names preset for the US English locale do not use Unicode characters.

If the Locale string is empty, the Currency Server default locale is used.

Possible values of Inflection: curncsrvUnitSingular (1), curncsrvUnitPlural (2).

The Currency parameter may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result.

Function **CurrencyWarning(***LicenseKey As String, Currency As String***)** As Boolean

Returns a Boolean value indicating whether the currency has been referenced by a recent warning condition. Possible causes of warnings include excessive fluctuation, extended lack

of updates, no data from one or more servers, conflicting data from different servers, FX feed rates conflicting with official EMU (European Economic and Monetary Union) rates, etc. This property can be reset to False by clicking **Reset Warnings** in the **Active Currencies** tab of Currency Server Manager.

Function **DomainToCurrency(***LicenseKey As String, Domain As String, ActiveOnly as Boolean***)** As String

Given an internet domain string (with or without protocol, username, password, directory and file information, e.g. "https://host.example.co.uk/directory/page.html", ".co.uk" and "uk" all produce the same result), returns a "most likely" currency code (e.g. "GBP"). If the ActiveOnly flag is set, only currencies from the list of active currencies are considered. Since there is no exact relationship between internet domains and countries and/or currencies (e.g. a currency or internet top level domain may be used in multiple countries), the result may be approximate. The function falls back to "EUR" for ".eu" domains, "USD" for ".mil" domains, and to the currency of the default locale set in Currency Server for non-national domains (e.g. ".com", ".int", ".edu", ".info", etc.) and for domains which could not otherwise be resolved to a country or region (e.g. IP addresses). If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code) is returned.

Function **Export(***LicenseKey As String, Format As String, Encoding as String, BaseCurrency As String, Locale As String, Flags As String, ServiceLicenseKey As String, ServiceExpirationTime As String, ServiceRenewalURL As String, ErrorCode As String, ErrorMessage As String, Time As String, Type As String***)** As String

This method exposes the output functionality used by post-update actions, generating exchange rates relative to BaseCurrency, and other currency properties. Rather than being written to a file or uploaded, as in the post-update actions, the data is returned as a string.

Possible values of Format: "INI" (default), "CSV", "TSV", "XML", "ECBCompatible", "JavaScript", "JSON", "JSONP". Some export formats may be further customized by specifying the name of a template file or JSONP callback function name, separated by a plus sign (e.g. "JavaScript+currencysystem5.js", "JSON+currencysystem5.json", "JSON +jsonCallback"). Template files must be stored in the "Templates" subdirectory of the Currency Server installation directory. If no template file is indicated, the default template for that format is used. If no JSONP callback function name is specified (allowed characters: letters, numbers and underscore - if the name contains a period it identifies the string as a template file name), the default "jsonCurrencySystem" name is used.

Possible values of Encoding: "CP-1252" (default), "UTF-8-BOM", "UTF-8-NoBOM", "UTF-16-LittleEndian-BOM", "UTF-16-LittleEndian-NoBOM", "UTF-16-BigEndian-BOM", " UTF-16-BigEndian-NoBOM". Supported legacy values: "ANSI" (same as "CP-1252"), "UTF-8" (same as "UTF-8-BOM"), "UTF-16" and "Unicode" (same as "UTF-16-LittleEndian-BOM"), "BigEndianUnicode" (same as "UTF-16-BigEndian-BOM").

JavaScript, JSON and JSONP data is always output as ANSI, with string characters having a code greater than 127 being output as \u-escaped hexadecimal values.

The Flags string may be composed of the following individual attributes. For the parts that are omitted, or if the string is left empty, default values are used.

Flag	Description
a0	Container = None (default)
a1	Container = GNU Zip
C1	Country = True (ISO 3166 Alpha-2 code)
C2	Country = True (ISO 3166 Alpha-3 code)
C3	Country = True (ISO 3166 numerical)
C4	Country = True (official name)
C5	Country = True (short name)
c	Country = False (default)
D	DateOnly = True
d	DateOnly = False (default)

E1	Entity = True (official name)
E2	Entity = True (short name)
e	Entity = False (default)
G	Line ending = CR+LF (default)
g	Line ending = LF
H	ColumnHeaders = True (default)
h	ColumnHeaders = False
K	Rank = True
k	Rank = False (default)
L	LegalTender = True
l	LegalTender = False (default)
M	Symbol = True
m	Symbol = False (default)
N	CurrencyNames = True (default)
n	CurrencyNames = False
O	Popularity = True
o	Popularity = False (default)
P	Physical = True
p	Physical = False (default)
R	Regime = True
r	Regime = False (default)
S	SmallestUnit = True
s	SmallestUnit = False (default)
T	TitleStyle = True (default)
t	TitleStyle = False
Z	Time priority = Oldest
z	Time priority = Newest (default)
1	Export only level 1 active currencies
2	Export level 1 & 2 active currencies
3	Export all active currencies (default)

The ServiceLicenseKey, ServiceExpirationTime, ServiceRenewalURL, ErrorCode, and ErrorMessage properties, if non-empty, are passed through directly to the output data, without further processing. The corresponding XML tags are <licensekey>, <serviceexpiration>, and <servicerenewal>. If ErrorCode is specified, no currency data is exported, and the error code is included in the output file. If ErrorMessage is specified, no currency data is exported, and the error message is included in the output file.

The Time and Type string parameters are used for forward compatibility and should be left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **LocaleToCurrency(***LicenseKey As String, Currency As String, Locale As String, ActiveOnly as Boolean***)** As String

Given a Windows locale identifier (LCID, e.g. "1033" or "0x409") or a three-character Windows locale code (e.g. "DEU", i.e. two-character ISO 639-1 Alpha-2 language codes plus an additional character, as specified for use with the Windows "LOCALE_SABBREVLANGNAME" type, which is not necessarily the same as the three-character ISO 639-2 Alpha-3 standard), or a two-letter ISO 639-2 Alpha-2 language code (e.g. "de"), returns a "most likely" currency code (ISO 4217 Alpha-3, e.g. "EUR"). If the ActiveOnly flag is set, only currencies from the list of active currencies are considered. Since there is no exact relationship between locales and currencies (e.g. a two-letter language locale code may match different countries with different currencies, and a three-letter code may match a country which is in a transition between two currencies), the result

may be approximate (especially if two-letter language codes are used, whereas LCID codes or three-letter Windows locale codes lead to more focused results). If the input string is empty or could not be resolved, the function returns a currency for the default locale set in Currency Server. If this default currency does not satisfy the ActiveOnly requirement, the first active currency (in alphabetical order sorted by ISO 4217 Alpha-3 code) is returned.

Function **Message(***LicenseKey As String***)** As String

Returns the message(s) from the FX feed(s), if any. If data was collected from multiple servers, a semicolon is used to separate information provided by the different servers.

Function **Rate(***LicenseKey As String, BaseCurrency As String, ToCurrency As String, Rounding As Boolean, Format As String, ReturnRate As String, Time As String, Type As String***)** As String

Returns the exchange (conversion) rate of ToCurrency, relative to BaseCurrency, with optional rounding and formatting. This method is equivalent to invoking Convert() with Amount set to 1. Both BaseCurrency and ToCurrency must be active currencies. The resulting value is both the amount of ToCurrency units for one unit of BaseCurrency and the conversion factor required to convert amounts expressed in BaseCurrency units to ToCurrency units (unless a special procedure such as triangulation is required).

Either BaseCurrency or ToCurrency (but not both) may contain a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of rates, instead of a single rate.

The ReturnRate parameter indicates whether the result should be returned as a string or numerical value. In order to not lose trailing 0s that may be useful for display purposes (e.g. 3.50), it is advised to request a string return value. Rounded numerical results never have trailing 0s (e.g. 3.5).

Possible values of ReturnRate: curncsrvReturnRateString (1), curncsrvReturnRateNumber (2).

For example, Rate(*LicenseKey*, "USD", "EUR", False, "+", 1, "", "") returns the euro amount corresponding to one US dollar, whereas Rate(*LicenseKey*, "EUR", "USD", False, "+", 1, "", "") returns the dollar amount equivalent to one euro, and Rate(*LicenseKey*, "EUR", "USD;GBP;JPY", False, "+", 1, "", "") returns the semicolon-separated rates.

Applying rounding to exchange rate values (rather than results of conversions) is not recommended, so Rounding is usually best set to False. For additional information on rounding please refer to the documentation of the Convert() method.

For additional information on the Format parameter please refer to the documentation of the Convert() method.

Note: if the result has more than 17 digits (e.g. a value of ten million billions without decimals) the double-precision number to string conversion logic used by this function may slightly round the string result. This is a limitation of the Microsoft libraries used by the software, which should not affect practical use. If this is unacceptable, request a numerical result and convert the result to a string using a different procedure.

The Time and Type string parameters are used for forward compatibility and should be left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **Source(***LicenseKey As String***)** As String

Returns the name(s) of the FX feed(s). If data was collected from multiple servers, a semicolon is used to separate information provided by the different servers.

Function **Time(***LicenseKey As String, Currencies As String, Information As Enum, Priority As Enum, ReturnTime As Enum, Time As String, Type As String***)** As String

Returns time information about the specified currency or currencies, or the current time.

The Currencies parameter may contain a single currency, or a list of semicolon-separated active currencies, in which case the output will consist of a semicolon-separated list of results, instead of a single result. The Currencies may also be left empty, if not applicable, or to indicate a request applied to the entire set of active currencies.

The Information parameter determines the request type: last modification, creation or expiration of the current exchange rate data, or subscription expiration of the FX feed(s) providing the data, or current time.

The Priority parameter indicates which time information ("newest" or "oldest") is to be returned if rates of different currencies (which may have had their rates last set or changed at different times) are combined to produce the desired result.

The ReturnTime parameter specifies whether the time should be returned as UTC (Universal Time, also referred to as "Zulu Time" or GMT), local time (server time), or seconds. UTC and local time strings are in the "YYYY-MM-DDThh:mm:ssTZD" format (e.g. "2008-12-31T14:00:59Z"), where TZD is the time zone designator ("Z", which stands for Universal Time, or "+hh:mm" or "-hh:mm"). The time in seconds is relative to midnight UTC of January 1, 1930.

Possible values of Information: curncsrvTimeRatesLastChange (1), curncsrvTimeRatesDataContent (2), curncsrvTimeRatesExpiration (3), curncsrvTimeServiceExpiration (4), curncsrvTimeCurrent (5), curncsrvTimeRatesEffective (6), curncsrvTimeRatesPublication (7).

Possible values of Priority: curncsrvTimeOldest (1), curncsrvTimeNewest (2).

Possible values of ReturnTime: curncsrvReturnTimeUT (1), curncsrvReturnTimeLocal (2), curncsrvReturnTimeSeconds (3).

Failed currency updates, and currency updates without changes to the specified currency or currencies do not change the internal time counters.

For most purposes, the effective time (curncsrvTimeRatesEffective) is the same as the publication time (curncsrvTimeRatesPublication), however for example for FX feeds that publish "next day" rates this information may differ. If no official publication time is known, then curncsrvTimeRatesPublication is set to be the same as curncsrvTimeRatesDataContent. If the data did not contain time information, then curncsrvTimeRatesDataContent is the collection time.

For currencies at a constant exchange or conversion rate (e.g. if ToCurrency is a subunit of BaseCurrency at a constant rate under a regime such as the EMU), the current date and time are used if no other information is available about the effective date of the constant rate. If one of a pair of currencies is the base unit used by the FX feed to provide the rate of the other currency, then the effective date of that rate is returned.

If the information is not available, for example if rates have never been set or changed, or if no expiration time is available, an empty string is returned. If the result was requested in seconds (numerical return value), the undefined case is returned as 0x7FFFFFFF (decimal 2147483647, C language LONG_MAX definition).

The Time and Type string parameters are used for forward compatibility and should be left empty. These two empty strings are guaranteed to either be ignored or be interpreted as a request for the latest daily mid rates (or most similar type available from the FX feed).

Function **Version(***LicenseKey As String***)** As String

Returns the software version.

Related Topics

- For cross-version compatibility information, see Legacy Support.
- For examples of programmatic access to the Currency Server methods and properties, see Code Examples.
- For more information about the different interfaces which can be used to access and control Currency Server, see Accessing the Software.

6.3 The SOAP Web Service Interface

Overview

The Simple Object Access Protocol (SOAP) is a cross-platform network protocol which combines proven technologies such as HTTP as a transport mechanism and XML as an encoding scheme, enabling interoperation between diverse platforms and environments (e.g. COM, CORBA, C++, C, Java, JavaScript, Perl, Python, PHP, etc.) SOAP is also the communications protocol used in Microsoft's .NET XML Web services platform. Currency Server adopted Microsoft's .NET technology in version 3.5. Previous versions of Currency Server featured a stand-alone, WSDL 1.1-compliant, SOAP interface.

SOAP functionality is now provided by the .NET-based Web service interface. Before the release of version 3.5 of Currency Server, i.e. when .NET itself had not yet been released by Microsoft, Currency Server featured a stand-alone SOAP interface.

Installation

The SOAP interface is installed as part of the Web service feature of the Currency Server installer. The WSDL (Web Services Description Language) file describing the SOAP services and the operations in the services, as well as the format that clients must follow when creating a SOAP message, can be accessed as "CurrencyServer5.asmx?WSDL", inside the "CurrencyServer" virtual directory which is created in the default website when the SOAP interface is installed (the site and the name of the virtual directory can be modified during or after installation).

If you installed Currency Server on the computer on which you are reading this text, and if you installed the Web service feature using the default site and virtual directory name, you can interact with the Web service by entering "http://localhost/CurrencyServer/CurrencyServer5.asmx" in your browser address bar (or click here). Append "?WSDL" to the same address (or click here) to open the WSDL file. You can also open the WSDL file by clicking the "Web service: Installed" link in the Software tab of Currency Server Manager.

Sample Code: Using SOAP in VBScript

In the following example a VBScript client invokes the ConvertToNum and ConvertToStr operations to convert 1000.00 US dollars to Japanese yen, and then displays a message using the AdminMessage operation.

Before running this sample code you may need to set up the Windows system to run SOAP client applications. This can be done by installing the Microsoft SOAP Toolkit, which is available for free download from the Microsoft website.

Remember to replace "localhost" with the appropriate host name unless you are testing the script on the same system on which Currency Server is running.

```
Dim SoapClient
Set SoapClient = CreateObject("MSSOAP.SoapClient")
Call SoapClient.mssoapinit("http://localhost/CurrencyServer/
CurrencyServer5.asmx?WSDL", "", "", "")
WScript.echo SoapClient.ConvertToNum("12345-12345-12345-12345", "USD", "JPY",
1000.00, True, "", "")
WScript.echo Soapclient.ConvertToStr("12345-12345-12345-12345", "USD", "JPY",
1000.00, True, "", "", "")
Call SoapClient.AdminMessage("12345-12345-12345-12345", "This is a message from
a SOAP client.", 1)
```

Sample Code: Using SOAP in HTTP

The following example shows a transaction in which the client asks the server to convert 1000.00 US dollars to Japanese yen, with formatting suitable for display purposes. The empty formatting string indicates to apply the default locale settings. The empty Date and Type parameters are included for forward compatibility. This is equivalent to a Convert("USD", "JPY", 1000.00, True, "",curncsrvReturnRateString) COM method call. The client provides SOAP request parameters within an HTTP POST request. The server follows the semantics of the HTTP status codes, e.g. a code of 200 in this case indicates successful processing. For additional information and

documentation you may want to refer to the SOAP and HTTP documents available on the W3C (World Wide Web Consortium) website.

Request from client (HTTP headers appear in italics):

```
POST /CurrencyServer/CurrencyServer5.asmx?WSDL HTTP/1.1
User-Agent: SOAP Sdk
Host: www.example.com
SOAPAction: "http://webservices.currencysystem.com/currencyserver/Convert"
Content-Type: text/xml
Content-Length: nnnn

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ConvertToStr xmlns="http://webservices.currencysystem.com/currencyserver/">
      <licenseKey>12345-12345-12345-12345</licenseKey>
      <fromCurrency>USD</fromCurrency>
      <toCurrency>JPY</toCurrency>
      <amount>1000</amount>
      <rounding>true</rounding>
      <format></format>
      <returnRate>curncsrvReturnRateString</returnRate>
      <time></time>
      <type></type>
    </ConvertToStr>
  </soap:Body>
</soap:Envelope>"
```

Response to client (HTTP headers appear in italics):

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ConvertResponse xmlns="http://webservices.currencysystem.com/
currencyserver/">
      <ConvertResult>123,456.00</ConvertResult>
    </ConvertResponse>
  </soap:Body>
</soap:Envelope>
```

Related Topics

- For cross-version compatibility information, see Legacy Support.
- For examples of programmatic access to the Currency Server methods and properties, see Code Examples.
- For more information about the different interfaces which can be used to access and control Currency Server, see Accessing the Software.

6.4 The JavaScript Interface

Overview

With the Currency System Script Library, Currency Server features the ability of dynamically generating JavaScript code (also known as ECMAScript or JScript) that not only contains a programming interface, but also embeds in the code file itself the up-to-date currency properties required for operation. This has several advantages, because as long as the server-side JavaScript code is refreshed periodically by Currency Server as part of regular currency data updates, no further online accesses are required by clients at runtime, resulting in self-contained JavaScript code that is faster, cacheable, simpler, more compatible and less affected by network latencies or errors. This approach offers Ajax benefits without the traditional Ajax communications overhead (e.g. to load external XML or JSON data) and browser requirements.

The code generated by Currency Server was designed to be compatible with JavaScript 1.2 or higher, i.e. with all browsers released since 1997-1998 (Netscape Navigator 4, Internet Explorer 4, and higher).

JavaScript updates can be pushed by Currency Server via a Post-Update Action or by generating the data via the COM or .NET Export() methods.

Customization

Currency Server includes the Currency System Script Library, which is stored as "currencysystem5.js" in the Filters directory. Given the scripted nature of the language, the code is included in source form (as deposited in full at the US Copyright Office). Per the standard End User License Agreement (EULA) this code may not be modified. The Currency System Script Library is available for extended customization under separate licensing terms.

Should you wish to replace the Currency System Script Library with your own code, you can replace the currencysystem5.js file, making sure that it contains a pair of <currencysystem-insert> and </currencysystem-insert> tags, which are used by Currency Server to locate the injection point of up-to-date currency data. Please note that the file itself may be overwritten by Currency Server updates.

Script Library Objects

The methods and properties exposed by the Currency System Script Library include traditional conversion functionality and features dedicated to dynamic web applications such as widgets and convert-as-you-type fields. Compared to the COM or .NET interfaces, the objects tend to expose higher-level functionality, while operating in full isolation from the Currency Server software (there are no administrative functions).

Three-character currency codes (ISO 4217 Alpha-3, e.g. "USD") are generally used where "currencies" and "codes" are referenced.

A "widget" prefix is used for objects which are focused towards converter tools such as widgets, with one source currency ("From" currency row), one or more destination currencies ("To" currency rows), and user-accessible lists (menus, drop downs, etc.) with available currencies.

A "parse(d)" prefix is used in the names of object that relate to the parser functionality, which is helpful in convert-as-you-type fields.

The Currency System Script Library is initialized by instantiating the object and invoking the initialize() function:

```
var currencySystem = new CurrencySystem();
currencySystem.initialize("", "en-US");
...
```

Methods:

Function **compareCurrencies(***Currency1, Currency2, Option***)** As Number

> Given two currency codes (ISO 4217 Alpha-3), this function provides a comparison result for collation purposes, e.g. to arrange the entries in a currency drop-down list.
>
> A return value of -1 indicates that Currency1 comes before Currency2; a value of 0 indicates parity; a value of 1 indicates that Currency1 comes after Currency2.

The comparison preference is set by the Option string parameter: "byCode" indicates that the codes alone are compared; "byName" indicates that the currencies are compared based on their names; "byNameAndCode" indicated that the currencies are compared based on their name first, and then by code (it is not uncommon for a country with high inflation to repeatedly issue new currencies, whereby the same names reappear every few years, in which case the codes are helpful for additional disambiguation).

Function **convert(***FromCurrency, ToCurrency, Amount, Rounding, Format***)** As String

Converts an amount from one currency to another, returning the converted amount with optional rounding and formatting. FromCurrency and ToCurrency may be any currency from the currencies[] array, i.e. not necessarily from the widgetListedCurrencies[] array.

The Rounding parameter is a Boolean value. When rounding is enabled, the result is rounded to the nearest Smallest Unit for the destination currency. In this case, the returned string may contain trailing 0s (e.g. "1.50" instead of "1.5").

As of version 5.0 of the Currency System Script Library the Format parameter is used for forward compatibility and should be left empty.

Function **currencyName(***Currency, Flags***)** As String

Returns the full currency name corresponding to the specified currency code.

The Flags parameter consists of one or more characters which set individual attributes, such as capitalization style (e.g. "US dollar" vs. "US Dollar").

Flag	Description
S	Singular
T	Title Style

As of version 5.0 of the Currency System Script Library the Flags parameter may only be "S" or "ST".

Function **currencySymbol(***Currency***)** As String

Returns the full currency symbol corresponding to the specified currency code.

Function **initialize(***LicenseKey, Locale***)** As Void

This function must be called immediately after object instantiation, in order to make the object fully functional.

If the code is used to create a widget or similar conversion box, widgetPopulate(), optionally followed by widgetSortLists(), should be called immediately after initialize().

As of version 5.0 of the Currency System Script Library the LicenseKey and Locale parameters are used for forward compatibility and should be left empty. The "en-US" locale (LCID 1033) is always used.

Function **parse(***FromString, FromCursorPosition***)** As Boolean

Processes the specified input string and cursor position, enabling convert-as-you-type functionality. Currency codes that precede or follow the numerical amount are removed from the numerical amount and interpreted as "From" and "To" selections (e.g. "USD100EUR" is interpreted as "Convert 100 US dollars to euros"). The numerical amount is normalized (e.g. the comma is converted to a period).

The function returns True if the specified string has been modified, indicating that the user interface field requires a visual update (using the modified fromString as stored in parsedFromString, and the new cursor position as stored in parsedFromCursorPosition).

Additional processing results are stored in the following properties (see the respective documentation below): parsedFromAmount, parsedFromString, parsedFromCursorPosition, parsedFromCurrency and parsedToCurrency.

The fromCursorPosition parameter must be set to -1 if no cursor position processing is needed.

Function **reloadCurrencies(***Script***)** As Boolean

Script clients may programmatically reload the script in order to get updated currency data

(rates, names, dates, etc.) This function applies the updated data.

The function returns True if successful.

In a widget-like implementation, the widgetPopulate() function, and optionally the widgetSortLists() function, must be called after a successful call to reloadCurrencies().

Function **suggestDefaultCurrency()** As String

Returns a "most likely" currency code, suitable for presetting a default currency. The function also takes into account the client system (locale, etc.)

Also see widgetSuggestUnusedCurrency(), which can be used to set additional currencies.

Function **widgetAddRow()** As Void

Appends new items to the widgetToCurrencies[] and widgetToResults[] arrays, also incrementing the value of the widgetToCount property.

This function is typically called when a conversion box is expanded with a new "To/Result" row.

Function **widgetConvert(***Amount, Rounding, Format***)** As Void

Performs a conversion whereby the source and destination currencies are taken from the current values of the widgetFromCurrency and widgetToCurrencies[] properties. Depending on whether widgetToCurrencies[] contains one or more items, this results in a "one to one" or "one to many" conversion. Results are placed in the widgetToResults[] array.

This function is suitable both for a convert-as-you-type scenario, and for a simpler "Convert" button event. For stand-alone conversion functionality, also see convert().

As of version 5.0 of the Currency System Script Library the Format parameter is used for forward compatibility and should be left empty.

Function **widgetCurrencyIsListed(***Currency***)** As Boolean

Returns True or False based on whether the specified currency code is present in the widgetListedCurrencies[] array, i.e. whether the currency is available for selection to the user.

Function **widgetCurrencyIsUsed(***Currency***)** As Boolean

Returns True or False based on whether the specified currency code is referenced by the widgetFromCurrency or widgetToCurrencies[] properties, i.e. whether the From field or the To field(s) in a conversion box reference the currency.

Function **widgetPopulate(***Selection, OldEuro***)** As Void

In a widget-like implementation, this function must be called immediately after initialize(), in order to populate the widgetListedCurrencies[] array with currency codes. The widgetSortLists() function may also be invoked after widgetPopulate().

The Selection parameter specifies which currencies are to be included: 0 = EMU (euro) currencies only, 1 = high popularity currencies, 2 = high and medium popularity currencies, 3 = all currencies. A value of 0 is useful for "Euro Converter" functionality, i.e. to convert between the euro and its subunits (currencies in a transition to the euro). A setting of 3 should be used with care, because it results in long lists of currencies (unless scrolling is enabled in the list elements, they may not even fit entirely on screen).

The OldEuro parameter can be set to 0 or 1, indicating whether EMU (euro) currencies which ceased to be legal tender and were replaced by the euro should be included (if they qualify for the Selection). If both Selection and OldEuro are set to 0, only the euro and currencies that are in a transition phase (imminent replacement by the euro) are included. Normally, OldEuro is set to 1 if Selection is set to 0 (EMU currencies only).

The function also initializes the widgetFromCurrency and widgetToCurrencies[0] properties with default values, enabling basic functionality for a simple From-To converter. If the function is called at runtime (rather than during initialization) to re-populate the widgetListedCurrencies[] array, it makes sure that the widgetFromCurrency and widgetToCurrencies[] properties reference valid currency codes (i.e. codes found in the widgetListedCurrencies[] array).

Function **widgetRemoveRow()** As Boolean

Removes the trailing elements from the widgetToCurrencies[] and widgetToResults[] arrays, and decrements the widgetToCount property.

This function should be called when the user opts to remove a row of "To/Result" fields from the user interface.

The function returns False (no action is taken) if the widgetToCurrencies[] and widgetToResults[] arrays already contain one item only (i.e. widgetToCount is 1).

Function **widgetSortLists(***Option***)** As Void

Rearranges the currencies in the widgetListedCurrencies[] array, based on the specified sort option. Please refer to compareCurrencies() for supported comparison options.

This function can be called after the initialize() and widgetPopulate() functions, to complete object initialization.

Function **widgetSuggestUnusedCurrency()** As String

Returns a currency code which is not already referenced by the widgetFromCurrency or widgetToCurrencies[] properties (i.e. used in the "From" or "To" fields of a conversion box). The currency is selected based on criteria such as popularity.

An empty string is returned if all codes are already used.

Also see suggestDefaultCurrency(), which is suitable for presetting a first currency ("From" field).

Properties:

Property **currencies[]** As Array of Objects

Array of Currency objects holding the complete set of data exported by Currency Server.

Property **currenciesTime** As Date

UTC date and time of the currency data.

Property **initialized** As Number

0 = not initialized, 1 = initialized, -1 = an error occurred during initialization (e.g. Currency Server failed to completely export the currency data).

Property **licenseKey** As String

License key passed to currencySystem.initialize().

Property **parsedFromAmount** As String

Normalized numerical amount set by the parse() function.

Property **parsedFromCurrency** As String

"From" currency code set by the parse() function, or empty string ("") if no currency selection was detected in the input text.

This property only needs to be checked if parse() returned True. In that case, if this property is non-empty, the client code can visually adjust the "From" field and update the widgetFromCurrency property.

Property **parsedFromCursorPosition** As Number

New position of the cursor in the "From" field, as set by the parse() function.

This property only needs to be checked if parse() returned True. A value of -1 indicates that the cursor position is unchanged.

Property **parsedFromString** As String

Normalized and stripped "From" text, after processing by the parse() function.

This property only needs to be checked if parse() returned True, which indicates that the "From" text was modified and requires a visual update. Possible modifications include normalization of numerical amounts (e.g. comma and periods), interpretation and removal of currency codes, and removal of invalid characters.

Property **parsedToCurrency** As String

"To" currency code set by the parse() function, or empty string ("") if no currency selection was detected in the input text.

This property only needs to be checked if parse() returned True. In that case, if this property is non-empty, the client code can visually adjust the appropriate "To" field and update the corresponding widgetToCurrencies[] property.

Property **version** As String

Returns the version of the Currency System Script Library.

Property **widgetListedCurrencies[]** As Array of Strings

Array of codes of the currencies listed as available for user selection in the "From" and "To" fields of the widget. The array is populated by calling the widgetPopulate() function, and contains the entirety or a subset of the codes referenced in the currencies[] array.

Property **widgetFromCurrency** As String

Currency code reflecting the current "From" field selection. This property is initially set by the widgetPopulate() function.

This property must be updated to correctly reflect different user selections, which may be the result of menu interaction, text input processed by parse() and returned via parsedFromCurrency, etc.

Property **widgetToCount** As Number

Number of items (rows) in the widgetToCurrencies[] and widgetToResults[] arrays.

Property **widgetToCurrencies[]** As Array of Strings

Array of one or more currency codes reflecting the current "To" field(s). The first code, i.e. widgetToCurrencies[0], is initially set by the widgetPopulate() function.

This property must be updated to correctly reflect different user selections, which may be the result of menu interaction, text input processed by parse() and returned via parsedToCurrency, etc.

Property **widgetToResults[]** As Array of Strings

Array of one or more numerical results reflecting the numerical amounts in the "To" field(s).

The widgetConvert() function stores the conversion result(s) in this array.

Related Topics

- For cross-version compatibility information, see Legacy Support.
- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information on post-update actions, see The Post-Update Action Dialog.
- For more information about adding and editing custom actions, see The Clients Tab.
- For more information about automatic update options, see The FX Feeds Tab and Use with Task Scheduler.

6.5 Legacy Support

Overview

Cross-version compatibility has always been a priority during the development of new versions of the software. This chapter aims to document the changes since the first interface definitions were published in 1999, so that legacy applications can either continue to run unmodified, or be updated to take advantage of the latest features.

COM Interface

Currency Server 6.*x* is compatible with both early binding and late binding clients designed to

interface with any previous version of the software.

All previous class identifiers (CLSIDs) and interface identifiers (IIDs) are supported by newer software, meaning that early binding clients compiled for previous versions of Currency Server will continue to work with newer versions of Currency Server. Early binding clients compiled for the 5.*x* interface will however not work with previous versions of the software.

Late binding clients referencing the functionality of a previous version of Currency Server by using the version-dependent ProgID will continue to work in newer versions of the software without requiring modifications. Late binding clients which reference the version-independent ProgID always access the latest version of the interface, meaning that version-independent late binding clients may find themselves working with newer objects than they were originally designed to support.

Up to version 4.*x* of the software the ProgID was "CloantoCurrencyServer.Server" (now it is "CurrencyServer.Application" or "CurrencyServer.Application.*version*"). This identifier is still supported for legacy purposes, and provides compatible access to the legacy interface. For additional detailed information on the 4.*x* interface, please refer to the documentation included with that version.

Clients designed to interface and work with any previous version of Currency Server will continue to work with Currency Server 6. No changes are required as part of this transition, because all versions of Currency Server aim to preserve compatibility when objects are extended. This is done either by using different names when major changes are introduced, or by keeping hidden methods and properties in place to support legacy clients.

If you would like to extend existing code to actively use version 5 and 6 features, you need to use the new Currency Server 5 root object, and you may need to edit a few other properties and methods. This is because in Currency Server 5 several changes were made to keep complexity at a minimum and to streamline the differences between different client access interfaces while adding functionality.

The following table lists the main differences between the 4.*x* interface and the new 5.*x* interface (used by Currency Server 5 and 6).

Version 4 Name	Version 5/6 Name	New Parameters
CloantoCurrencyServer.Server	CurrencyServer.Application	
Currencies	ActiveCurrencies	
Information	ActiveCurrencies	
ConvertToNum()	Convert()	Format, Return, [Time], [Type]
ConvertToStr()	Convert()	Return, [Time], [Type]
RateNum()	Rate()	Format, Return, [Time], [Type]
RateStr()	Rate()	Return, [Time], [Type]
Custom, LocalizedCustom	Custom	Locale
Name, LocalizedName	Name	Locale, TitleStyle
Symbol, LocalizedSymbol	Symbol	Locale
IsEuro	Regime	
All time-related objects	Time()	*See Time()*

.NET Web Service Interface

In order to retain cross-version compatibility, beginning with Currency Server 5 the Web service access points are versioned. Currency Server 5 and 6 install a file named "CurrencyServer5.asmx" to the web service directory. Previous file names used in Currency Server 3 and 4 and in the managed Currency System Web service were "CloantoCurrencyServer.asmx" and "CurrencyServer.asmx". Currency Server 5 and 6 not only do not remove any previous versions that may have been installed, but they also install an additional "CurrencyServer4.asmx" file and interface compatible with version 4.*x*. If needed, "CurrencyServer4.asmx" can be copied and renamed to "CloantoCurrencyServer.asmx" or "CurrencyServer.asmx".

If you would like to extend existing code to actively use version 5 features, you need to use the new Currency Server 5 access point (e.g. "CurrencyServer5.asmx"), and you may need to edit a few other properties and methods.

In version 5, the XML namespace was updated from "http://webservices.cloanto.com/currencyserver/" (as in "CurrencyServer4.asmx") to "http://webservices.currencysystem.com/currencyserver/" (as in "CurrencyServer5.asmx"). This URI may be set or updated automatically at runtime or by development environments, in which case it does not need to be configured manually. A hybrid "CurrencyServer4-cs.asmx" sample file is also installed with Currency Server, allowing access to the version 4.*x* interface using the new namespace.

New features of version 5 in general include support for semicolon-separated lists of multiple currencies in many methods. The semicolon (";") also replaces the previously-used vertical bar character ("|", decimal ASCII code 124).

The following table lists the main differences between the 4.*x* interface and the new 5.*x* interface (used by Currency Server 5 and 6).

Version 4	Version 5/6	New Parameters
CurrencyServer.asmx	CurrencyServer5.asmx	
Currencies()	ActiveCurrencies()	
InformationCopyright()	Copyright()	
InformationMessage()	Message()	
InformationSource()	Source()	
ConvertToNum()	Convert()	Format, ReturnRate
ConvertToStr()	Convert()	ReturnRate
RateNum()	Rate()	Format, ReturnRate
RateStr()	Rate()	ReturnRate
CurrencyExists()	CurrencyExists()	ActiveOnly
CurrencyServers()	CurrencyFeeds()	
All time-related methods	Time()	*See Time()*

SOAP Web Service Interface

Before the release of version 3.5 of Currency Server, i.e. when .NET itself had not yet been released by Microsoft, Currency Server featured a stand-alone SOAP interface. This first version of the SOAP server implemented a subset of the current SOAP operations, which are now provided by the .NET-based Web service interface.

Clients designed to use the old interface can be used with the current version of Currency Server without requiring any changes in the way SOAP operations are used. The original WSDL file is preserved when the new version of Currency Server is installed. Currency Server 5 and 6 further install both version 4 and version 5 ASMX and WSDL access points.

Because compatibility with the previous versions is retained by means of the original access points, you do not need to update your code. If however you wish to use the extended set of operations of the new Web service interface, then the URL of the original WSDL file ("CurrencyServer/CurrencyServer.wsdl" in the default website) should be updated to point to "CurrencyServer/CurrencyServer5.asmx?WSDL". In this case, if your SOAP client is running on Windows, you may also want to make sure that you have the latest version of the Microsoft SOAP Toolkit installed, since early versions may not completely support the WSDL data generated by the .NET server.

The names of some operations have changed between versions. Compared to the original pre-3.5 SOAP interface, an initial license key parameter was added to all operations. Functionally, output compatibility is retained with code written for the previous SOAP interface

The following table lists the names of the operations which changed between the old SOAP interface and the new .NET-based Web service. Changes in parameters are marked with an asterisk (*).

Before Version 3.5	Version 3.5	Version 5/6
ConvertToNum	ConvertToNum*	Convert*
ConvertToStr	ConvertToStr*	Convert*
CurrencyName	CurrencyName*	CurrencyName (unchanged)
CurrencySecondsSinceChange	CurrencySecondsSinceLastChange	Time*
InformationCreation	InformationCreationTimeLocal	Time*
InformationCreationUT	InformationCreationTimeUT	Time*
InformationExpiration	InformationExpirationTimeLocal	Time*
InformationExpirationUT	InformationExpirationTimeUT	Time*
Message	AdminMessage	AdminMessage (unchanged)
SecondsSinceChange	SecondsSinceLastChange	Time*
UpdateNow	AdminUpdateNow	AdminUpdateNow (unchanged)

Old VBScript example:

```
Call SoapClient.mssoapinit("http://localhost/CurrencyServer/
CurrencyServer.wsdl", "", "", "")
Call SoapClient.Message("This is a message from a SOAP client.", 1)
```

Version 5 VBScript example (the license key is fictitious):

```
Call SoapClient.mssoapinit("http://localhost/CurrencyServer/
CurrencyServer5.asmx?WSDL", "", "", "")
Call SoapClient.AdminMessage("12345-12345-12345-12345", "This is a message from
a SOAP client.", 1)
```

JavaScript Interface

A first internal-use 4.*x* version (originally marked as 1.0 or 1.1) served as a testbed in preparation for the 5.*x* version, which is the first public version. Version 4.*x* was widely deployed in popular widgets such as Currency Converter 1.0 for Google Desktop. It is maintained internally for legacy widgets, but is otherwise unsupported.

Chapter 7

7 Add-On Components

This section covers:

- Client Validation Modules
- Custom Action Modules
- FX Feed Filters
- The Universal Filter
- Client-Server Configurations

7.1 Client Validation Modules

Currency Server includes a Web service interface, which gives validated .NET and SOAP clients access to currency information and conversion functionality. Client validation modules can be written to validate the identity of the Web service client, e.g. to run a subscription-based service.

This technical section is intended for software developers who wish to add custom Web service client validation functionality to Currency Server.

Web Service Clients

When Web service clients invoke a method they provide a License Key parameter, which is validated by Currency Server and/or by an additional client validation module.

Validation by Currency Server depends on the installed options and licenses. Normally, Currency Server checks if the license key provided by the client is syntactically valid. This verification is highly efficient and has a minimum impact on performance. If the software is running in Unlimited Web Service Clients mode, Currency Server performs no license key validation, and accepts any license key, including null keys. In either mode, and unless the license key was found to be invalid, Currency Server can pass the license key on to an optional client validation module, which can approve or reject the request.

Installation and Activation

Client validation module files must be stored in the "Validators" subdirectory of the Currency Server installation directory.

Client validation module files can be automatically installed by a third-party setup program. The target directory is stored in the system registry in the HKEY_LOCAL_MACHINE\Software\Cloanto\Currency Server*version*\Validators subkey, where *version* is the program version (4.0 or higher). If multiple versions of the program are installed, different subkeys may be present, in which case it is recommended to give preference to the highest version number.

Client validation modules are activated in the **Clients** tab of Currency Server Manager, under **Client Validation**, where it is also possible to set **Run As...** credentials.

Input and Output

When invoked by Currency Server, a client validation executable file is passed the optional parameters set in the **Validation** field (**Clients** tab of Currency Server Manager), followed by the license key as provided by the Web service client and normalized by Currency Server.

Normalization of the license key provided by the Web service client is always done for security reasons (to reduce the possibility of buffer overflow and directory traversal exploits). The license key is truncated to a maximum of 254 characters, and """, "/", "\" and ":" are converted to underscore characters.

The client validation module is not invoked if the license key does not pass validation by Currency Server.

If Currency Server is running in Unlimited Web Service Clients mode, Currency Server performs only normalization (and not the internal validation step), and the client validation module, if set, is always invoked.

By default, the client validation code runs under the context of the invoking COM client. When invoked by a Web service client, by default the invoking context is that of the IIS Worker Process. In these contexts, prompt dialogs are not allowed. Scripts should have an "On Error Resume Next" statement to prevent the appearance of an (invisible) error dialog which would lock the script.

At the end of the process, the validation module must return an appropriate HRESULT exit code, e.g. in main() or WinMain() for C code, or "WScript.Quit *exitcode*" in VBScript, etc. Zero or positive values indicate success (the license key has been successfully validated).

HRESULT Exit Codes

Currency Server uses 32-bit HRESULT result codes, also known as "COM Error Codes", where negative return values (high-order bit set to 1, i.e. SEVERITY_ERROR) are used to indicate an error condition. The Facility field (also in the higher order 16 bits) is set to FACILITY_ITF (indicating an interface-specific error). The 16 lower order bits are used for Currency Server, with the higher order three bits (out of 16) indicating an error category within Currency Server.

For example, in C/C++ filter error codes could be returned as:

```
return MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, currency_server_code);
```

The following table lists the 16-bit Currency Server-specific error codes which have been assigned to describe possible validation failures.

The following well-known codes have been assigned in the "Other Error" range (three high-order bits set to 111, i.e. decimal 7).

Binary	Hex	Description
11100000 00000000	E000	No subscription data found (e.g. incorrect key)
11100000 00000001	E001	Subscription has expired (e.g. needs to be renewed)
11100000 00000010	E002	Hit quota exceeded (e.g. maximum number of daily hits reached)
11100000 00000011	E003	No subscription data found (e.g. valid, but not registered or activated)
11100000 00000100	E004	License key revoked/replaced (e.g. license key reported lost or stolen)
11100000 00000101	E005	License key blocked/replaced (e.g. license key found on "warez" sites)

Additional information about HRESULT and COM Error Codes is available in FX Feed Filters, and in the Microsoft MSDN Library (web link).

Example

The following VBScript example ("client_validator.vbs") shows a simple client validation module which only accepts a license key "1111-2222-3333-4444-5555" (which is fictitious, and will only be accepted by Currency Server operating in Unlimited Web Service Clients mode). User interface interaction, e.g. MessageBox() or similar, is not allowed, because the code is executed in the context of a Web service.

```
Option Explicit
Const VALIDATOR_CATEGORY = &H8004E000
Const VALIDATOR_DETAIL_NOSUBSCRIPTION = 0 ' No subscription data found
Const VALIDATOR_DETAIL_EXPIREDSUBSCR = 1  ' Subscription has expired
Const VALIDATOR_DETAIL_HITEXCESS = 2      ' Hit quota exceeded
Const VALIDATOR_DETAIL_TOACTIVATE = 3     ' License key not yet registered or
activated
Const VALIDATOR_DETAIL_REVOKED = 4        ' License key reported lost or stolen
Const VALIDATOR_DETAIL_BLOCKED = 5        ' License key blocked
Const E_INVALIDARG = &H80070057
Dim WshShell, WshArgs, LicenseKey
' Check if the script has been started by Currency Server
'
Set WshShell = WScript.CreateObject("WScript.Shell")
```

```
Set WshArgs = WScript.Arguments
If WshArgs.Count < 1 Then
        ' This script must be started as a client validation file by Currency
Server
        WScript.Quit E_INVALIDARG
End If
' a license key is passed as first argument
'
LicenseKey = WshArgs(0)
If LicenseKey <> "1111-2222-3333-4444-5555" Then
        '
        ' Nonzero exit code: unsuccessful validation
        '
        WScript.Quit VALIDATOR_CATEGORY + VALIDATOR_DETAIL_NOSUBSCRIPTION
End If
```

Related Topics

- For more information about editing client validation settings, see The Clients Tab.
- For more information about .NET clients, see The .NET Web Service Interface.
- For more information about SOAP clients, see The SOAP Web Service Interface.
- For more information about adding an Unlimited Web Service Clients license, see The Software Tab.
- For more information about security, see Security Sandbox.

7.2 Custom Action Modules

Custom action modules serve two main purposes:

- automatic execution after an exchange rate data update (e.g. to upload the rates to a remote server, or to store the rates in a database, etc.);
- quick menu selection via Currency Server notification area icon (e.g. to invoke updates, to load or save settings, or to manually start other custom actions).

This technical section is intended for software developers and system administrators who wish to create custom action modules.

Installation and Activation

Custom action module files must be stored in the "Actions" subdirectory of the Currency Server installation directory. Executables placed in this directory are listed in the Currency Server notification area icon menu.

Custom action module files can be automatically installed by a third-party setup program. The target directory is stored in the system registry in the HKEY_LOCAL_MACHINE\Software\Cloanto\Currency Server*version*\Actions subkey, where *version* is the program version (4.0 or higher). If multiple versions of the program are installed, different subkeys may be present, in which case it is recommended to give preference to the highest version number.

The automatic execution of custom action modules by Currency Server is activated in the **Clients** tab of Currency Server Manager, under **Post-Update Actions**, where it is also possible to set **Run As...** credentials.

By default, the custom action code runs under the context of the "Currency Server" service, and is not allowed to interact with the desktop. In this context, prompt dialogs are not allowed. Scripts should have an "On Error Resume Next" statement to prevent the appearance of an (invisible) error dialog which would lock the script.

At the end of the process, the custom action module should return an appropriate HRESULT exit code, e.g. in main() or WinMain() for C code, or "WScript.Quit exitcode" in VBScript, etc. Use a

return value of 0 (i.e. NOERROR, or S_OK) to indicate success, or a specific HRESULT error code (e.g. HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND), E_ACCESSDENIED, E_INVALIDARG, etc.) to be logged and/or reported by Currency Server.

Execution Conditions

Currency Server runs the custom action module after an exchange rate data update. Exchange rate updates can be invoked in three ways:

- by Currency Server, via the **Auto-Update** functionality;
- by an external COM or Web Service client (e.g. invoked by Windows Task Scheduler, by a SQL DTS package, by the user via the notification area icon menu, etc.);
- by the user, via the **Update Now** button (in which case the execution of the custom action is deferred until Currency Server Manager is closed and intent to execute the action has been confirmed).

Exchange rate data updates can produce different outcomes, which are detected by Currency Server and which can be used as conditions for the execution of the custom action module:

- Failure (e.g. connection problems, no rates are loaded, previous rates are automatically retained)
- Success, but no changes between the new rates and the previous set of rates
- Success, and rates have changed since previous update

These conditions can be selected via the **Always**, **Only if successful update** and **Only if rates changed** options in the **Clients** tab of Currency Server Manager.

Custom Action Modules vs. Direct Invocation

When combined with the **Only if successful update** or **Only if rates changed** option, post-update custom action modules become a powerful way to simplify and limit operations such as the writing of new exchange rates to a web or database server. The checking for errors or lack of changes is done by Currency Server.

For example, before this functionality was available, a VBS client using Currency Server to write exchange rates to a web server would have had to invoke an update, e.g. via the Admin.UpdateNow() method, and then either always update the rates (even if they hadn't changed), or check (with quite a few lines of code) whether there had been errors or whether the rates hadn't changed at all, and only then upload the rates. With the new functionality, the client would still invoke the Admin.UpdateNow() method, but the upload is then invoked not by the same VBS code, but rather via the Custom Action Module functionality, set in the **Clients** tab of Currency Server Manager.

Related Topics

- For more information about the configuration of custom actions, see The Clients Tab.
- For more information on post-update actions, see The Post-Update Action Dialog.
- For more information about security, see Security Sandbox.

7.3 FX Feed Filters

Currency Server can process virtually any type of exchange rate data by means of both built-in and external "plug-in" filters, which consist of at least one INI file, plus an optional executable file.

This technical section is intended for software developers who wish to create custom FX feed filters, or edit filters provided by Currency System under a source code license.

Additional and more specific information is available for the universal filter.

Overview of Filter Files

All filter files must be stored in the "Filters" subdirectory of the Currency Server installation

directory. In order for a filter to be listed in the **Edit FX Feed** dialog (i.e. when you click **Add...** or **Edit...** in the **FX Feeds** tab in Currency Server Manager), it must be "described" by an INI file in the "Filters" directory. The format of this FX feed filter INI file is described below.

The data access and conversion functionality of a filter which is referenced by an INI file can be implemented using built-in (internal) Currency Server protocol access and conversion functionality, or by using an external executable file, which is also stored in the "Filters" directory. The input/output interface of the FX feed filter executable file is described below. If a custom executable file is used, it must output the processed currency exchange rate data as an INI file complying with the SERIFF specification. The format of the SERIFF output INI file is also described below.

Installation of Files

Filter executable files and accompanying INI files may be packaged into a .cs-filter file, which is a ZIP archive with a modified file suffix. Filter packages are easy to download and open with a double-click, upon which Currency Server provides an automatic installation prompt.

Filter executable files and accompanying INI files can also be automatically installed by a third-party setup program. The target directory is stored in the system registry in the HKEY_LOCAL_MACHINE\Software\Cloanto\Currency Server\version\Filters subkey, where version is the program version (4.0 or higher). If multiple versions of the program are installed, different subkeys may be present, in which case it is recommended to give preference to the highest version number.

INI File Structure

FX feed filter description files and SERIFF data output files both use the INI structure and are 8-bit text files (CP-1252 character set) divided into sections, each containing one or more keys. Each key contains one or more values. The files can be created and modified with a simple editor like Notepad or WordPad (Save as type: Text Document).

Example:

```
[SectionName]
keyname=value
;comment
keyname=value, value, value ;comment
```

Section names are enclosed in square brackets, and must begin at the beginning of a line. Section and key names are case-insensitive, and cannot contain spacing characters. The key name is followed by an equal sign ("=", decimal code 61), optionally surrounded by spacing characters, which are ignored.

Multiple values for a key are separated by a comma followed by at least one spacing character.

Numerical values may be entered in decimal, hexadecimal (digits prefixed by "0x") or octal format (digits prefixed by a "0" character). In Boolean keys "True" and "False", and "Yes" and "No" are equivalent to 1 and 0.

When the Currency Server parser encounters an unrecognized section name, the entire section (with all its keys) is skipped. Within a known section, only unrecognized keys are skipped.

Both Space (decimal code 32) and Horizontal Tab (HT, decimal code 9) are acceptable spacing characters.

Lines are terminated by a CR (decimal code 13) and/or LF (decimal code 10) character.

Comments are introduced by a semicolon character (";", decimal code 59). Comments must begin at the beginning of a line or after a spacing character. Comments terminate at the end of the line.

Additional technical information about the INI file format as it is used in Currency Server is available online.

Sections in FX Feed Filter INI Files

The INI file which describes a Currency Server FX feed filter contains two types of sections:

- [General] (one per file)
- [Feed] (multiple sections allowed)

[General]

This section contains the keys used to describe a data source in general (but not a specific FX feed provided by the data source).

Keys:

CountryCode = *CC*

This is an ISO 3166 Alpha-2 country code (e.g. "US", "EU", "JP") or user-assigned code element (e.g. "XC", "XI") associated with the data source. The value can be used for information and sorting purposes in the Currency Server **FX Feed** tab.

This key is optional.

ID = *OID*

This is the Object Identifier (OID) of the data source, and is used to identify the entity even if it changes name.

This key is required.

Information = *URL*

This is the URL (Uniform Resource Locator) which is opened with the default web browser when the user clicks the "Web Information" link in the Currency Server **Edit FX Feed** dialog. It is recommended to always provide such an information page, describing in more detail the data source and feed, subscription information (if any), etc.

This key is optional. If the key is not provided, no link is displayed in the software user interface.

Name = *String*

This is the name of the data source as it is displayed in the Currency Server **Edit FX Feed** dialog, e.g. "Example Central Bank" or "Example Data Corporation", etc.

This key is required.

Version = *String*

This key indicates the version of the data source filter, and must consist of one to four sequences of digits separated by a dot character (e.g. "4", "1.2.3.4", "5.00.2195", etc.) It is recommended to use the a.b.c.d format.

If multiple INI files with the same data source OID (ID key in the [General] section) exist, only the one with the higher version is considered.

This key can also be read by Software Director for version checking purposes, if the publisher entry includes information for it.

This key is required.

DeletePreviousVersions = *Boolean*

If this key is set to True, filter files which reference the same data source OID (ID key in the [General] section), but having a less recent Version, are deleted. This is only useful if newer versions have different file names, as new filter files always overwrite previous versions.

This key is optional. The default value is False.

RequiredVersion = *String*

This key indicates the minimum required version of the Currency Server software, and must consist of four sequences of digits separated by a dot character (e.g. "1.2.3.4"). The version should be set to at least 4.1.0.0 if XML functionality is used by the filter.

This key is optional.

[Feed]

One or more sections of this type contain the keys used to describe one or more FX feeds provided by the data source. Unless a default feed is specified by using the Type key, the first [Feed] section found in the INI file indicates the default feed.

Keys:

Address = *URL*

This is the URL (Uniform Resource Locator) from which the exchange rate data for this feed is fetched. In addition to RFC 1738 URLs (following the *protocol://hostname:port/directory/filename* syntax), a local path (e.g. "C:\directory\filename") or UNC path (Universal Naming Convention, following the *servername**sharename**directory**filename* syntax) can be indicated. In addition to standard protocol names such as "http", "https" and "ftp", Currency Server supports the "fxp" and "fxpd" protocol identifiers (as per documentation released to the public by Oanda, Inc.)

In addition to supporting characters allowed as per RFC 1738, square brackets may be used in the Address URL to specify certain variable parts to be replaced at run time. The possible items include [subscription-id] (or [1], for legacy support) to insert a subscription service User ID, [userdata] (or [2]) reserved for private use (UserData2 registry value), [series-tag] to insert a "series tag" and [*x*], where *x* is a single, case sensitive, time element code in the same format used by the Windows GetDateFormat() and GetTimeFormat() functions (e.g. "https://www.example.com/data/[yyyy]/[yy]-[MM]-[dd].xml").

The time element is expressed in the time zone set by the PublicationTime key (with one day added, if RequestTimeNextDay is set).

Please note that especially for date and time-dependent functionality the behavior is not set uniquely by the filter executable, but also by the Currency Server software (which supplies the time based on the specific query requirements), and by the FX server (which may even return an error if a future or otherwise invalid time is provided).

A vertical bar character ("|") can be used to indicate multiple URLs (e.g. "https://www.example.com/data1.ini|https://www.example.com/data2.ini"), the content of which is merged before being processed.

This key is optional. Possible scenarios in which this key is not used could include the case where the data source filter executable "knows" how and where to get the data from, or test data being provided by means of the Append key.

Append = *String*

This key can be used to append a static block of SERIFF data to the end of the SERIFF output (e.g. as provided by the FX feed filter executable). Line separators can be encoded by using the "\n" sequence. This can be useful, for example, to append information about currencies which have a known constant conversion rate with respect to another currency already included in the feed, without having to modify the filter binary.

This key is optional. This key may also be placed in the [General] section, in which case it acts as a default fallback if no Append key is specified in one or more [Feed] sections.

DataContentTime = *Time Zone, Local Time*

This key is used to process the date and time data contained in the feed.

Time Zone is a Microsoft Time Zone Index value (e.g. 35 for US Eastern Time, 360 for UTC, etc.)

The optional Local Time argument is a time in the *HH:mm:ss* (e.g. "15:45:00") or *HH:mm* (e.g. "15:45") or *HH* (e.g. "15") form, typically used to set a specific time of the day (e.g. the official time at which a source is known to publish its daily rates) if the data only contains a date, but no time. If the feed contains time information, and the Local Time argument is set, then the latter is added as an offset.

This key is optional. If the key is not used, the time information in the data is assumed to be UTC. The previous name of this key, TimeInfo, remains supported for legacy compatibility.

DataContentType = *String*

This key is used to describe the expected content type of the feed data.

The value follows the two-part format convention used for IANA media types, with the addition of "text/ini" for INI content, and "text/tsv" for tabular data. Supported types include "text/csv", "text/html", "text/json", "text/plain", "text/xml" and "application/octet-stream" (for binary data).

This key is required.

Description = *String*

This is a short description of the data feed, which is displayed in the **Description** field in the Currency Server **Edit FX Feed** dialog based on the selected data feed, e.g. "Daily reference rates provided by Example Data Corporation", etc.

This key is optional. If the key is not provided, no data feed description text is displayed in the software user interface.

DisplayName = *String*

This is an optional dynamic name that can be built including variable parts to generate a more meaningful and specific name than with the static Name key. This name is listed in the Currency Server **FX Feed** tab, where it is useful in differentiating between multiple instances of the same feed, which only differ in the variable part(s).

The items include [series-tag], [custom-url] and [username], respectively mapping to the Series tag, Custom URL and Username fields as set in the **Edit FX Feed** dialog.

This key is optional. If the key is not provided, the static feed name as set in the Name key is used.

Executable = *File Name*

This key indicates the name of the FX feed filter executable file for this data feed. Such an executable is required if the data is not accessible by using a protocol and format which are directly supported by Currency Server (e.g. SERIFF over HTTP). The file must be stored in the "Filters" subdirectory of the Currency Server installation directory. The specifications for data input and output and return codes are included below.

This key is optional. If the key is not provided, the data must be accessible in a format and protocol supported natively by Currency Server. If the built-in XML filter is to be referenced, the key must explicitly be set to "xml.exe".

Fetch = *Mode, Wait, BrowserEngine, BrowserVersion*

This key specifies how the content referenced by the Address key should be fetched.

The Mode value may be set to "Raw" or "Dynamic". In raw mode, the data is collected exactly as provided by the protocol set in the Address key. In dynamic mode, the data is first loaded in the browser container set via the BrowserEngine value, then it is given some additional (optional) time to run any client-side scripts, and finally the resulting rendered content is collected.

The Wait value indicates a delay, in milliseconds, to be added after the data is fetched. In dynamic mode, the wait time starts from when the document completes loading (status changes to "ready"). This value is ignored in raw mode.

BrowserEngine specifies the HTML rendering engine to be used in dynamic mode. The value can be set to "IE" (uses Windows "Trident" engine) or "Chrome" (requires Chrome to be installed). This value is ignored in raw mode.

If BrowserEngine is set to "IE", BrowserVersion is a Microsoft WebBrowser control value (FEATURE_BROWSER_EMULATION) which defines the default Internet Explorer emulation mode. For example, a value of "11001" indicates to render the page in IE11 edge mode, regardless of the declared !DOCTYPE directive (lacking a !DOCTYPE directive, the page is loaded in Quirks mode), whereas a value of "11000" indicates to render the page in IE11 edge mode if it contains a standards-based !DOCTYPE directive. A minimum value of "9000" (IE9 mode) is recommended. If BrowserEngine is set to "Chrome", BrowserVersion sets the minimum required version of Chrome. This value is ignored in raw mode.

This key is optional. If the key is not provided, the data is fetched in raw mode, with no

delay.

ID = *OID*

This is the Object Identifier (OID) of the data feed, and is used to identify the feed independently of modifications to the feed URL or name.

This key is required, and must be an immediate OID subnode of the ID indicated in the [General] section.

Map = *File Name*

This key indicates the name of the mapping file, which is used to process the data by way of simple replacement rules. The file must be stored in the same directory as the filter INI file (in the "Filters" subdirectory of the Currency Server installation directory). A file can be shared across multiple filters, or unique for a given filter.

This key is optional.

Name = *String*

This is the name of the data feed as it is listed in the Currency Server **FX Feed** tab and in the **Edit FX Feed** dialog.

This key is required. Also see the optional DisplayName key, for cases where it may be useful to display a more detailed resulting (dynamic) name

PublicationTime = *Time Zone, Local Time*

This key describes the time zone and the estimated daily publication time of the feed. When reading the data, it is used to set the time elements in the query. After reading the data, it becomes a property of the exchange rate set if the data itself contained no publication time. In a future version of the software, this property may be used to schedule feed-specific collection time ranges.

Time Zone is a Microsoft Time Zone Index value (e.g. 35 for US Eastern Time, 360 for UTC, etc.), indicating the time zone used for the Local Time data.

Local Time is a time in the *HH:mm:ss* (e.g. "15:45:00") or *HH:mm* (e.g. "15:45") or *HH* (e.g. "15") format.

This key is optional. If this key is not used, and if the data contained no time information, the publication time is set to the time at which the data was fetched.

Rank = 1 | 2 | 3

Indicates the default feed rank value of the feed (3 = high popularity, 2 = medium, 1 = low). Currency Server displays this default value if no value was explicitly set by the user.

A data source may publish multiple feeds that have different feed rank values. For example, its default XML-based feed may have a value of 3 (very popular feed), while its legacy feed may have a value of 1 (low popularity). When displaying the rank value of a data source (opposed to one of its feeds), the popularity of the data source is determined by the feed with the highest popularity value.

This key is optional. The default value is 1.

ReplacedBy = *OID*

This key is used to replace the FX feed with a different one (from the same data source or from a different data source). When a new filter or software version is installed, Currency Server automatically applies any ReplacedBy statements.

For example, if a data source used to publish two feeds, one at 10 AM and one at 2 PM, and one of the two feeds is later discontinued, the best option to avoid an error in relation to data collection from the discontinued feed could be to replace the discontinued feed with the only remaining active one.

This key is optional.

Replaces = *URL*

This key is used to replace one or more old URLs with the new URL referenced by the

Address key. When a new filter or software version is installed, Currency Server checks to see if any URLs have been updated, and applies the updates automatically.

This key is optional. Multiple keys are supported, to indicate several URLs which are to be replaced with the URL referenced in the Address key.

RequestTimeNextDay = *Boolean*

This is a Boolean value indicating whether one day should be added to the time element used in the Address key. This is useful for feeds which publish "next day" rates, in which a request for the current day's rates would result in the rates published the previous day, rather than the latest (current) rates.

This key is optional. The default value is False.

Resource = *URL*

This key references a secondary resource property. For example, some feeds use one URL (referenced by the Address key) for the exchange rate data query, and another URL (referenced by the Resource key) for a static table containing the list of available currencies.

When accessing the Currency Server Web Service (e.g. for monitoring and testing), the Resource key may be set to "CurrencyServer5", "CurrencyServer4" or "CurrencyServer4-cs" to respectively test version 5, version 4 and a hybrid (version 4 API with version 5 namespace) web service.

This key is optional.

TriangulationRate = *Unit1, Unit2*

This key is used to add a "helper" exchange rate between Unit1 and Unit2 to the command line of the FX feed filter executable file referenced by the Executable key. This rate can be used for internal triangulation purposes by basic filters, e.g. those that process all data in EUR, while the data provider uses a different base currency.

If this triangulation mechanism is used, care should be taken to ensure that the rate between Unit1 and Unit2 is as recent as may be required.

This key is optional.

Type = Free | Subscription | SoftwareLicense | Private | Default

This flag is used to categorize the type of data feed based on its availability and use. "Free" indicates that the data feed is free to access (most feeds from public sources such as central banks are of this type). "Subscription" means that the fee is subscription-based. "SoftwareLicense" indicates that the feed is only available to licensed users of the Currency Server software. The "Private" flag is available for internal use and for other custom projects which do not fit into one of the other categories. "Default" may be combined with any one of the previous flags to indicate the default feed.

This key is required.

UserAgent = *String*

This key makes it possible to set the HTTP User-Agent string that is presented to the server when fetching content.

This key is optional. If the key is not provided, the default "Web Loader" user agent is used.

The following is an example of a filter INI file:

```
[General]
ID = 1.3.6.1.4.1.23153.100.14
Name = "Example Data Corporation"
Executable = "exampledata.exe"
Information = "https://www.example.com/csfilterinforedirect.html"
Append = "\n\n[EUR]\nEUR=1.00\n"
Version = 1.0.0.0
```

```
[Feed]
ID = 1.3.6.1.4.1.23153.100.14.1
Name = "Primary Server"
Popularity = 1
Type = Free, Default
Description = "Daily reference rates provided by Example Data Corporation
(primary server)"
Address = "https://fxserver1.example.com/rates/daily.csv"
Replaces = "http://fxserver1.example.com/oldurl.csv"
Replaces = "http://oldserver.example.com/daily.csv"

[Feed]
ID = 1.3.6.1.4.1.23153.100.14.2
Name = "Secondary Server"
Popularity = 3
Type = Free
Description = "Daily reference rates provided by Example Data Corporation
(secondary server)"
Address = "https://fxserver2.example.com/rates/daily.csv"
```

Sections in Mapping INI Files

Mapping data files are used to apply simple replacement rules, e.g. to map non-standard and other non-XML currency codes and text strings to ISO 4217 currency codes as listed in the **All Currencies** tab of Currency Server Manager. Mapping data files must use the .cs-map file name suffix, and be referenced by the Map key in one or more FX feed filter files. A mapping data file may be shared by multiple filters. It contains four types of sections:

- [General] (one per file)
- [Codes] (one per file)
- [Names] (one per file)
- [BaseCurrencies] (one per file)
- [Multipliers] (one per file)

[General]

This section contains version information.

Keys:

Version = *String*

> This key indicates the version of the currency mapping INI file, and must consist of four sequences of digits separated by a dot character (e.g. "4.1.0.0").
>
> This key can be read by Software Director for version checking purposes.
>
> This key is required.

MatchSubstrings = *Boolean*

> If this key is set to True, substring (partial) matches are allowed against currency name strings in the [Names] section. When using this option, care should be taken to avoid the possibility of multiple matches (e.g. a generic "dollar" may match both "US dollar" and "New Zealand dollar").
>
> This key is optional. The default value is False.

[Codes]

This section is used to assign one or more non-standard currency codes to the correct ISO 4217 currency codes as used by Currency Server. It is used to normalize data from sources which tend to use currency codes, but which also include non-standard codes. If present, the [Codes] section is applied to the output of any filter (including XML filters).

Keys:

CurrencyCode = *NonStandardCode1, NonStandardCode2, NonStandardCode3, etc.*

This key, named after the ISO 4217 currency code to which it refers, contains one or more non-standard codes which should be mapped to it.

XXX = *IgnoreCode1, IgnoreCode2, IgnoreCode3, etc.*

The special ISO 4217 code XXX is used to assign codes to "No Currency", i.e. to ignore one or more non-standard codes which may be found in the data.

[Names]

This section is used to assign one or more currency name strings to ISO 4217 currency codes as used by Currency Server. It is used to normalize data from sources which refer to currencies by name or symbols rather than using currency codes. For example, XML filters that use the XPathName key refer to the table set in the [Names] section of the mapping file.

Keys:

CurrencyCode = *NonStandardName1, NonStandardName2, NonStandardName3, etc.*

This key, named after the ISO 4217 currency code to which it refers, contains one or more currency names or symbols which should be mapped to it (e.g. USD = "U.S. Dollar", "US Dollar", "$").

The Tab character (\t) may be used to match one or more spacing characters. Unicode characters may be expressed as *\xHHHH* (hexadecimal) sequences or, if the file is saved as a Unicode text file, embedded directly in the name strings.

XXX = *IgnoreName1, IgnoreName2, IgnoreName3, etc.*

The special ISO 4217 code XXX is used to assign currency name strings to "No Currency", i.e. to ignore one or more names which may be found in the data.

[BaseCurrencies]

This section is used to specify the base currency unit relative to which each rate is expressed. For XML filters, use the BaseCurrency and XPathBaseCurrency keys instead of this section.

Keys:

CurrencyCode = *BaseCurrency*

This key, named after the ISO 4217 currency code to which it refers, specifies the base currency unit relative to which the rate is expressed.

This key is optional. By default, each rate is considered to be expressed relative to one CurrencyCode unit (the CurrencyCode key is the same as the BaseCurrency value).

[Multipliers]

This section is used to address exchange rate unit multiplier values. For example, if the exchange rate for the Japanese yen is provided for 100 yen rather than for 1 yen, a multiplier value of 100 could be applied to the given rate in order to generate the exchange rate for 1 yen. If present, the [Multipliers] section is applied to the output of any filter (including XML filters, even if the XPathMultiplier key is already used).

Keys:

CurrencyCode = *Multiplier*

This key, named after the ISO 4217 currency code to which it refers, contains a single multiplier that should be applied to the value.

This key is optional. The default rate multiplier value is 1 (no change).

Built-In Formats

The following access protocols and file formats are supported natively (internally) by Currency Server, and can be referenced by a data source INI filter file without the need for a dedicated FX

feed filter executable file:

- XML over standard protocols (e.g. HTTP, HTTPS, FTP, local file system, etc.)
- SERIFF (Simple Exchange Rate Information File Format as published by Cloanto) over standard protocols
- PRN (Implementation of PRN File Format to Describe Currency Exchange Rates as published by Cloanto) over standard protocols
- Currency Server Web Service (enumeration of currencies and loading of exchange rates as per SOAP Web Service Interface)
- FXP (protocol and format as published by Oanda, Inc.)
- FXPD (protocol and format as published by Oanda, Inc.)

XML: The universal filter can be configured by means of XPath expressions to interface with different FX feeds.

SERIFF: The SERIFF implementation of Currency Server is described in more detail below. Proxy servers are supported.

PRN: Currency Server supports currency data in spreadsheet format, also referred to as PRN, which is used by some online services. For a description of the PRN format as it is supported by Currency Server please refer to this document. SERIFF and PRN can be chained in a way to support the best of both worlds (e.g. using the Next key in a SERIFF file to reference PRN data). Proxy servers are supported.

The **Currency Server Web Service** format may be of interest mostly for monitoring purposes (to test services provided by Currency Server).

FXP: Currency Server can query servers which are compatible with version 1.1 of the Foreign Exchange Protocol Specification as released to the public by Oanda, Inc. on October 1, 1997.

This protocol provides exchange rate information in terms of daily median bid and ask prices. The software calculates the average of the two values to extract the mid rate. Unlike SERIFF and PRN, which can be implemented using different transport mechanisms (e.g. HTTP, HTTPS, FTP, etc.), FXP is based on its own mechanism, built on top of TCP/IP. If the port number is omitted in the Address key of the [Feed] section of the filter INI file, the port defaults to 5011 (this port number, while it appears to not be officially registered, is mentioned in the FXP specification). If a firewall or other filter is installed on the network make sure to open the port number as needed. HTTP and SOCKS proxy servers are not supported by Currency Server for this protocol.

FXPD: This format is an adaptation of FXP, in which the HTTP protocol is used to send a POST request containing an FXP query. This method offers better support for proxy servers than FXP, and, if implemented on port 80, does usually not require changes to firewall configurations. Proxy servers are supported.

Additional formats are supported by means of FX feed filter executable files. SERIFF remains built-in because it is the common format used to interchange data between the software and the filters. PRN remains built-in because it shares some functionality with the SERIFF parser. FXP and FXPD remain built-in because they are linked to a non-standard access protocol, and it was more efficient to implement support by sharing lower level access functionality used in other software modules.

FX Feed Filter Executable Files

The FX feed filter executable file must be stored in the "Filters" subdirectory of the Currency Server installation directory, and is referenced by the Executable key in the [General] section of the filter INI file.

The Examples section on the Currency System website includes sample source code of a data conversion filter.

Input and Output

When invoked by Currency Server, a filter executable file is passed one or two parameters:

- the name of a temporary file (created and destroyed by Currency Server) used for both data input and data output;

- the URL of the FX feed, if provided by the Address key of the [Feed] section.

If an Address key was provided in the [Feed] section, Currency Server downloads the data from the URL and places it in the temporary file.

If no Address key was provided in the [Feed] section, the filter executable is expected to fetch the data directly.

At the end of the process, the filter file must write data in SERIFF format as described below to the temporary data file. Currency Server then reads the file, and then deletes it.

Error Conditions

A filter may return an error in one of three ways:

- by providing an appropriate HRESULT exit code, e.g. in main() or WinMain() for C code, or "WScript.Quit *exitcode*" in VBScript, etc.;
- by placing an Error key in the [General] section of the SERIFF data, in which case the error message string is notified and the rest of the data is ignored;
- by deleting the original file.

The first option (HRESULT exit code), which is used by the filters distributed with Currency Server, is the recommended method, and it is the only one which allows the software to issue a notification message which includes the error code and a plain text description (for known codes).

HRESULT Exit Codes

Currency Server uses 32-bit HRESULT result codes, also known as "COM Error Codes", where negative return values (high-order bit set to 1, i.e. SEVERITY_ERROR) are used to indicate an error condition. The Facility field (also in the higher order 16 bits) is set to FACILITY_ITF (indicating an interface-specific error). The 16 lower order bits are used for Currency Server, with the higher order three bits (out of 16) indicating an error category within Currency Server.

For example, in C/C++ filter error codes could be returned as:

```
return MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, currency_server_code);
```

The three high-order bits of the 16-bit Currency Server-specific error are assigned to indicate eight different error categories (16-bit binary values):

- 000*xxxxx xxxxxxxx* (decimal 0 << 13): Reserved
- 001*xxxxx xxxxxxxx* (decimal 1 << 13): Error in Input Parameters (command line)
- 010*xxxxx xxxxxxxx* (decimal 2 << 13): Memory Allocation Error
- 011*xxxxx xxxxxxxx* (decimal 3 << 13): File Read Error
- 100*xxxxx xxxxxxxx* (decimal 4 << 13): File Write Error
- 101*xxxxx xxxxxxxx* (decimal 5 << 13): Protocol-Specific Error Number (e.g. HTTP error 404)
- 110*xxxxx xxxxxxxx* (decimal 6 << 13): Data Content Error (0-7 assigned to well-known codes, from 8 to 999 reserved for future well-known codes)
- 111*xxxxx xxxxxxxx* (decimal 7 << 13): Other Error (0-5 assigned to well-known codes, from 6 to 999 reserved for future well-known codes)

Even when the exact error code is not defined, each error category has a plain text description which reaches the user as part of the notification. Be sure to make your return codes as meaningful and unique as possible in order to be able to track the exact position within the code where the error condition occurred.

Data content errors (three high-order bits set to 110) are used to indicate that the data could be accessed properly, however the data itself was not in the format that was expected by the filter executable. Within this category, codes from 0 to 999 (decimal) are reserved for assignment by Currency System. You can use either the well-known codes which have already been assigned, or use codes higher than 999 decimal (i.e. in the binary range 11000011.11101000 to 11011111.11111111).

The following well-known codes have been assigned in the data content error range (three high-order bits of Currency Server-specific 16-bits set to 110).

Hex Value	Description
0x8004C000	HTML instead of non-HTML data received (e.g. proxy or authentication message)
0x8004C001	Expected known data not found
0x8004C002	Expected data terminator not found
0x8004C003	Required base unit not found
0x8004C004	Known error content detected
0x8004C005	Rates not published
0x8004C006	No currencies found
0x8004C007	One or more currencies had a rate of zero

Sections in SERIFF Output INI File

After successful connection and parsing of the data, the FX feed filter executable is expected to write the output in SERIFF format to the file referenced in the parameter passed to the executable file by Currency Server.

Additional technical information about the SERIFF specification is available online. The following information is an implementation-specific subset.

The output file must contain at least two sections:

- [General] (one section per file)
- [Service] (one section per file, required only for subscription-based FX feed services)
- [BaseCurrency] (one section required, multiple sections allowed)

[General]

This section contains the keys used to describe the data set in general.

Keys:

Application = *String*

This string should be set to "Cloanto(R) Currency Server(TM)", or to its CP-1252 character set equivalent, "Cloanto® Currency Server™".

This key is required.

Copyright = *String*

This key indicates the copyright information for the data, if any (e.g. "© 2004 Example Data Corporation", or "Please refer to our website for information concerning copyrights and terms of use.", etc.)

This key is optional.

DataContentTime = *String*

This is the creation date and time associated with the exchange rate data (Universal Time), derived from elements contained in the feed (if present). The date string is in the "YYYY-MM-DDThh:mm:ssZ" format (e.g. "2006-02-13T14:00:59Z"), where "Z" is the time zone designator which stands for Universal Time (also referred to as "Zulu Time").

This key is optional. If the key is not provided, Currency Server adds a date stamp based on the collection time.

EffectiveTime = *String*

This is the effective date and time associated with the exchange rate data (Universal Time). The date string is in the "YYYY-MM-DDThh:mm:ssZ" format (e.g. "2006-02-14T14:00:59Z"), where "Z" is the time zone designator which stands for Universal Time (also referred to as "Zulu Time").

This key is optional. If the key is not provided, the effective time is considered to be the

same as the publication time.

ExpirationTime = *String*

This is the expiration date and time associated with the exchange rate data (Universal Time). The date string is in the "YYYY-MM-DDThh:mm:ssZ" format (e.g. "2006-02-14T14:00:59Z"), where "Z" is the time zone designator which stands for Universal Time (also referred to as "Zulu Time").

The expiration information is used by the **Auto-Update** function in order to schedule automatic updates.

This key is optional. If the key is not provided, the update frequency of the **Auto-Update** function is controlled by the "update at least every..." value which can be changed by the user.

Message = *String*

This key indicates an information message which is to be notified to the user. By default, identical messages (e.g. if the same Message key string is received multiple consecutive times) are notified only once.

This key is optional. This key should be use sparingly, as it triggers a notification event.

PublicationTime = *String*

This is the official publication date and time associated with the exchange rate data (Universal Time). The date string is in the "YYYY-MM-DDThh:mm:ssZ" format (e.g. "2006-02-13T14:00:59Z"), where "Z" is the time zone designator which stands for Universal Time (also referred to as "Zulu Time").

This key is optional.

Source = *String*

This key indicates the name of the data source (e.g. "Example Data Corporation"). It should normally be set to the same value as the Name key in the [General] section of the data source filter INI file.

This key is optional.

[Service]

This section contains service expiration and renewal information, so that it is possible for the client application to inform the user before the current feed subscription expires, and to include a quick renewal link. If the service is not provided as a subscription, then this section may be omitted.

ExpirationTime = *String*

This key indicates the date and time of the end of the current service subscription.

This key is optional.

Renewal = *URL*

This key indicates the URL, preferably with account-specific information, which can be used by the service customer to renew the current subscription.

This key is optional.

[BaseCurrency]

This section, named after the ISO 4217 currency code of the base unit relative to which rates are expressed, contains the actual exchange rate data. For full compliance with EMU triangulation requirements, the internal base unit for exchange rates should be the euro (EUR code).

CurrencyCode = *Mid Rate*

or

CurrencyCode = *Bid Rate, Ask Rate*

This key indicates the rate of the currency (relative to 1 BaseCurrency unit), defined either as the mid rate, or as a bid and ask pair. If a value of a bid/ask pair is undefined, it should be set to 0. Trailing zero characters may be used to indicate the precision of the data (e.g. 123.4560 vs. 123.456).

There should be one key, named after the corresponding ISO 4217 currency code, for each currency, except for the BaseCurrency currency (which would always have a rate of 1).

The following is an example of SERIFF output INI file generated by a filter:

```
[General]
Application = "Cloanto(R) Currency Server(TM)"
PublicationTime = 2006-11-02T17:46:03Z
Source = "Example Data Corporation"
Copyright = "This data is public and free for any use."
Message = "Hello World"

[Service]
ExpirationTime = 2007-12-31T23:59:59Z
Renewal = "https://www.example.com/renew/?account=12345"

[EUR]
AUD = 1.7373
BGN = 1.9466
CAD = 1.5922
CHF = 1.5494
CNY = 9.40923
CYP = 0.58754
CZK = 31.983
DKK = 7.4312
EEK = 15.6466
GBP = 0.7044
HKD = 8.9065
HUF = 263.38
ILS = 5.0256
INR = 52.408
ISK = 87.58
JPY = 136.97
KRW = 1347.45
LKR = 110.34
LTL = 3.4532
LVL = 0.6504
MTL = 0.428
MXN = 11.901
MYR = 4.3201
NOK = 8.2065
NZD = 1.9526
PLN = 4.374
ROL = 37247
RUB = 34.402
SEK = 9.178
SGD = 2.0058
SIT = 234.66
SKK = 41.835
TRL = 1631000
USD = 1.142
XDR = 0.8177
ZAR = 8.4333
```

Related Topics

- For more information about the universal FX feed filter, see The Universal Filter.
- For more information about FX feeds and automatic update options, see The FX Feeds Tab.
- For more information about automatic currency data updates and notifications, see Automatic Updates.
- For more information about proxy servers and system requirements, see System Requirements.
- For more information about security, see Security Sandbox.

Web Links

- For more information about the INI file format, see Implementation of INI File Format.
- For more information about the SERIFF specification, see SERIFF (Simple Exchange Rate Information File Format).
- For more information about currency data in PRN files, see Implementation of PRN File Format to Describe Currency Exchange Rates.
- For more information about Object Identifier (OID) assignments, see Currency System Object Identifiers (OIDs).

7.4 The Universal Filter

As explained in the FX Feed Filters section, Currency Server can process virtually any type of exchange rate data by means of both built-in and external "plug-in" filters, which consist of at least one INI file describing the data, plus an optional executable file to process the data. In order to process exchange rate data in XML format, Currency Server includes a built-in filter featuring an XML parser, an XLST processor and an XPath query language interpreter. This makes it possible to reference XML data by processing and describing the relevant parts of the XML document in the filter's INI file without having to create a new filter binary executable.

Overview of XML Filter Files

The built-in XML filter executable file is named "xml.exe". Like all filter executable files, it is stored in the "Filters" subdirectory of the Currency Server installation directory.

As is the case for all other feeds, XML data feeds accessed via the built-in XML filter are "described" by one or more INI files in the "Filters" directory. The general format (but not some XML-specific keys) of such filter INI files is the same as the one documented for generic FX feed filters. The following aspects are also shared among all filters (including the built-in XML filter files):

- Installation of files
- INI file structure
- All keys in the [General] section of the INI file
- Non-XML-specific keys in the [Feed] section of the INI file
- Input and output parameters
- Error conditions
- Exit codes

Please refer to the documentation of the generic FX feed filters for information on the above topics. Certain additional XML-specific keys which are used in the [Feed] section of the INI file are described below.

XML-Specific Keys in the [Feed] Section of FX Feed Filter INI Files

One or more [Feed] sections may contain both the keys as documented for generic FX feed filter INI files, and the following XML-specific keys. The XPath strings are standard expressions which can be tested with common online services and local XML editing and debugging tools. For example, an XPath expression "A[1]/B[1]/C/text()" would select the text value of all C elements that are children of the first B element within the first A element (when there are multiple matching elements, square brackets can be used to select a specific element).

XML-Specific Keys:

BaseCurrency = *Unit*

This key specifies the base currency unit relative to which rates in the XML data are expressed, unless specified otherwise on a per-currency basis via XPathBaseCurrency.

This key is required.

DefaultRate = Direct | Inverse

This key specifies whether the currency rates in the XML data are represented as direct rates (local currency unit per base currency unit) or as inverse rates (base unit per local unit). The default rate type may additionally be changed (inverted) on a per-currency base using the XPathInvertRate key (or, indirectly, the XPathBaseCurrency key).

This key is optional. The default value is Direct.

Namespaces = *Namespace1:URI1, Namespace2:URI2, etc.*

This key can be used to define one or more XML namespaces. The namespaces can be used in XPath expressions to prefix tag names and to reach specific XML nodes.

This key is optional.

RateDecimalSymbol = *String*

This key specifies the character used as a decimal separator in the exchange rate values (e.g. "." as in "123.45").

This key is optional. The default value is ".".

RateGroupingSymbol = *String*

This key specifies the character used as a digit grouping symbol in the exchange rate values (e.g. "," as in "1,234.56"). The grouping symbol characters, if found, are stripped from the rate value.

This key is optional. By default, no digit grouping symbol is set (no characters are stripped from the rate value).

XLST = *File Name*

This key specifies an XSL Transformation that is to be applied to the XML data before the XPath query is run. The XLST file must be stored in the same directory as the filter INI file (in the "Filters" subdirectory of the Currency Server installation directory).

This key is optional.

XPathAskRate = *XPath Expression*

This key is used to reference the ask values of the currency exchange rates. The individual values can in turn be inverted or multiplied on a case by case basis as specified by means of the XPathInvertRate and XPathMultiplier keys.

Unless XPathRate is used, this key is required and must be used in combination with XPathBidRate.

XPathBaseCurrency = *XPath Expression*

This key addresses a set of currency codes (as per ISO 4217), indicating the base currency unit relative to which the associate rate is expressed. The key serves a function similar to XPathInvertRate, in that it is generally used to support feeds where some rates are "inverted", i.e. expressed in terms of local currency units per one foreign currency unit. However, XPathBaseCurrency allows exceptions to be set in terms of currency codes instead of Boolean values.

If the currency code is different from the BaseCurrency value, then an inversion is applied, and the specified currency code overrides the code that would otherwise have been retrieved with XPathCode. If the currency code is the same as BaseCurrency value, then the status specified by DefaultRate applies.

This key is optional. The default value is set by the BaseCurrency key, to which the status

specified by DefaultRate applies.

XPathBaseCurrency and XPathInvertRate are mutually exclusive and cannot be used in the same description.

XPathBidRate = *XPath Expression*

This key is used to reference the bid values of the currency exchange rates. The individual values can in turn be inverted or multiplied on a case by case basis as specified by means of the XPathInvertRate and XPathMultiplier keys.

Unless XPathRate is used, this key is required and must be used in combination with XPathAskRate.

XPathCode = *XPath Expression*

This key is used to reference the currency codes (as per ISO 4217).

The number of values referenced by this key must exactly match the number of values referenced by all other XPath keys (with the sole exception of the data referenced by the XPathTime key, which may reference either the entire data set, or each key individually). This is usually achieved by using XPath predicates. For example, the "[@code and @rate]" predicate in the following keys are used to make sure that only "currency" tags with both "code" and "rate" attributes (introduced by the "@" syntax) are selected:

```
XPathCode = /exchangerates[1]/dailyrates[1]/currency[@code and @rate]/@code
XPathRate = /exchangerates[1]/dailyrates[1]/currency[@code and @rate]/@rate
```

Either this key or XPathName are required.

XPathError = *XPath Expression, TriggerError*

If present, this human-readable error message string is logged and notified to the administrator as part of an error condition. If TriggerError is True, the presence of the error string itself is considered an error condition. If TriggerError is False, the string is ignored unless an error condition is triggered by other factors (e.g. no currency data, etc.)

XPathInvertRate = *XPath Expression*

This key addresses a set of Boolean values (True, False, or 1, 0). A simple Boolean value can be used to specify, for each currency exchange rate, whether the DefaultRate setting (Direct or Inverse) must be applied as is, or inverted. The "inverted" condition is equivalent to setting the value of XPathBaseCurrency to the same value as XPathCode.

This key is optional. The default value is False, meaning that the status specified by DefaultRate applies unmodified.

XPathInvertRate and XPathBaseCurrency are mutually exclusive and cannot be used in the same description.

XPathMultiplier = *XPath Expression*

This key is used to address exchange rate unit multiplier values. The XPathMultiplier values are multiplied by the corresponding XPathRate, XPathAskRate and XPathBidRate values to generate the final exchange rate values.

This key is optional. The default rate multiplier value is 1 (no change).

XPathName = *XPath Expression*

This key filters the XML data to generate a list of currency names. The names are then converted into currency codes as set in the [Names] section of the mapping data.

Either this key or XPathCode are required.

XPathRate = *XPath Expression*

This key is used to reference the currency exchange rate values (mid rates). The individual values can in turn be inverted or multiplied on a case by case basis as specified by means of the XPathInvertRate and XPathMultiplier keys.

This key is required, unless XPathBidRate and XPathAskRate are used.

XPathServiceExpirationDays = *XPath Expression*

This key indicates the expiration date of the current feed service subscription, in active days remaining.

This key is optional.

XPathServiceExpirationTime = *XPath Expression, TimeFormat*

This key indicates the expiration date of the current feed service subscription.

This key describes the format used to express time information in the XML data, using the same format as the Windows GetDateFormat() and GetTimeFormat() functions, or using single quotes do mark characters to be processed literally (e.g. "yyyy-MM-dd'T'HH:mm:ss'Z'"). Text after the date in the specified format, if any, is ignored.

The TimeFormat string value describes the format used to express time information in the XML data, using the same format as the Windows GetDateFormat() and GetTimeFormat() functions, or using single quotes do mark characters to be processed literally (e.g. "yyyy-MM-dd'T'HH:mm:ss'Z'"). Text after the date in the specified format, if any, is ignored.

This key is optional.

XPathServiceRenewal = *XPath Expression*

This key indicates the URL of the web page which can be used to renew the current service.

This key is optional.

XPathTime = *XPath Expression, TimeFormat*

This key addresses one or more time strings. The strings must be in the format specified by the TimeFormat key. If more than one date is addressed, the least recent one is associated with the currency exchange rate data. As a single time value can be used to reference the entire data set, this is the only XPath expression which does not need to result in a number of items matching the number of items referenced by the other XPath expressions.

The TimeFormat string value describes the format used to express time information in the XML data, using the same format as the Windows GetDateFormat() and GetTimeFormat() functions, or using single quotes do mark characters to be processed literally (e.g. "yyyy-MM-dd'T'HH:mm:ss'Z'"). Text after the date in the specified format, if any, is ignored.

This key is optional.

The following is an example of an XML FX feed filter INI file:

```
[General]
ID = 1.3.6.1.4.1.23153.100.14
Name = "Fantasy Data Corporation"
RequiredVersion = 5.8.10.0
Information = "https://www.example.com/fantasydata/csxmlfilterinforedirect.html"
Append = "\n\n[EUR]\nEUR=1.00\n"
Version = 1.0.0.0

[Feed]
ID = 1.3.6.1.4.1.23153.100.14.1
Name = "XML Web Service"
Popularity = 1
Type = Subscription
Description = "Daily reference rates provided by Fantasy Data Corporation (XML
Web service). Enter the web service license key in the User ID field."
Address = "https://fx.fantasydata.example.com/rates.fx?licensekey=[1]
&format=xml"
BaseUnit = EUR
DataContentTime = 360
Namespaces = "fantasydata:http://fx.fantasydata.example.com"
XPathTime = "/fantasydata:rates[1]/fantasydata:time[1]/text()", "yyyy-MM-
ddTHH:mm:ssZ"
```

```
XPathCode = "/fantasydata:rates[1]/fantasydata:currency[fantasydata:code and
fantasydata:rate]/fantasydata:code[1]/text()"
XPathRate = "/fantasydata:rates[1]/fantasydata:currency[fantasydata:code and
fantasydata:rate]/fantasydata:rate[1]/text()"
Executable = "xml.exe"
```

XML-Specific HRESULT Exit Codes

The built-in XML filter uses the following well-known exit codes in the input parameters (three high-order bits of Currency Server-specific 16-bits set to 001) and data content error (three high-order bits set to 110) range.

Hex Value	Description
0x8004200A	A namespace definition did not include the ':' separator
0x8004200B	The namespace could not be created
0x8004200D	A mandatory XPath expression was omitted (e.g. XPathRate, XPathCode, XPathName)
0x8004200E	An XPath expression (e.g. XPathCode, XPathName, XPathMultiplier, XPathInvertRate) failed to reference the same number of items referenced by the XPathRate string
0x8004D001	The XML parser failed to parse the XML data
0x8004D002	The XPathTime string failed to reference any data
0x8004D004	A piece of data referenced by XPathTime could not be reached
0x8004D005	A piece of data referenced by XPathTime could not be parsed using the provided TimeFormat string
0x8004D006	An XPath expression failed to reference any data (e.g. XPathCode, XPathName, XPathMultiplier, XPathRate, XPathInvertRate)
0x8004D007	The service expiration string could not be parsed
0x8004D008	A rate string contains illegal characters

See the chapter on FX Feed Filters for additional general information on Currency Server HRESULT result codes.

Related Topics

- For more information about FX feed filters, see FX Feed Filters.

Web Links

- For more information about the XPath specification, see XML Path Language (XPath).

7.5 Client-Server Configurations

In addition to server software like Currency Server, Currency System offers client software such as customized currency calculators.

This technical section covers the deployment and operations of client-server configurations, and is intended for administrators of systems where Currency Server is used to feed exchange rate data to calculator clients by Currency System.

For additional information about the calculator software which is available from Currency System, please refer to the following websites:

- WorldCalc (examples of currency-enabled customizable calculators)
- Euro Calculator (examples of customizable calculators with a focus on EMU)
- Calculator Builder (SDK and license to create redistributable custom calculators)

Location of Data Files

One of the most important planning steps in the client-server configuration is the decision involving the physical location and the access URL of the currency data files which are accessed

by the calculator clients.

Depending on the configuration, there are usually two or three compact (less than 1 KB) INI files:

- Exchange Rate Data Header File (optional, rarely modified, contains static information such as currency names, and points to Exchange Rate Data File via Next key)
- Exchange Rate Data File (required, usually accessed at most once a day to get new rates, contains exchange rate data written by Currency Server)
- EMU Configuration File (required, accessed at most once a week to check for changes in the status of EMU currencies)

In theory, as INI files can be chained without limitations, another separate file could be used for separate management of banner advertising data (if the calculator client includes a banner).

You should decide whether to use the ".txt" or ".ini" suffix for the INI files served to calculator clients, and act accordingly:

- When the first versions of the client software were released in 1999, it was safe to use the ".ini" suffix for the names of INI currency data files fetched from the internet. Personal firewalls and pop-up advertising blockers are now increasingly used by consumers. These products can limit access to files based both on file suffix and relation between user action and access to remote files by the software. Some products use a combination of factors to determine if access to a file should be blocked or not, e.g. a rapid succession of accesses to test1.ini and test2.ini file would be blocked, whereas test1.ini followed by test2.txt would not be blocked. Because INI files are text files, and because files ending in ".txt" are not blocked in the same way as ".ini" files, you should consider using the ".txt" suffix for all INI files served to calculator clients.
- MIME Media Types instruct web clients how to handle files received from a server. The HTTP protocol specification requires that the web server reports the MIME type for content. By default, servers like Internet Information Services 6.0 serve only files with extensions registered in their MIME type list. If you plan to use the ".ini" suffix for INI files, and if your web server does not already have a MIME type entry for such files, we recommend that it be added (set the MIME type for extension ".ini" to "application/octet-stream").
- If you plan to use the ".ini" suffix for INI files, and if a security filter such as Microsoft Urlscan is running on the web server to control client requests, it must be configured to allow the data files to be fetched (e.g. by adding .ini to the [AllowExtensions] section and removing it from [DenyExtensions] in %SystemRoot%\System32\Inetsrv\Urlscan\URLScan.ini). If it is not possible to change the security settings, the names of the data files should be set to end with ".txt" instead of ".ini".

Other requirements include:

- The files must be accessible by the calculators from the internet over HTTP (recommended) or HTTPS. An existing web server usually provides the ideal solutions for hosting the files.
- At least one of the files (i.e. the Exchange Rate Data File) should be written automatically at least once a day during weekdays. This can be done via a **Post-Update Action**. If Currency Server is running on the web server, it can write the file(s) locally. If the web server is on another machine, a protocol like WebDAV or FTP can be used to upload the file(s).
- Any address (URL) that is set must be stable in the long term, because it becomes hard-coded inside the calculator client. This can be achieved by placing the files in a dedicated directory, e.g. "rates", at the root of the web server. This can even be a virtual directory, mapped to a different location which is easily accessible for management and updates (e.g. via FTP).
- It may be desirable to leave open the option for a future relocation of the data files, e.g. for maintenance reasons. This can be achieved by using a dedicated host name for the URLs used by the client, e.g. by setting up an appropriate DNS record and referencing the web server as rates.example.com instead of www.example.com.

The following naming conventions are recommended for the data files:

- The file containing the exchange rate data should be named after the base unit (which, for full compliance with EMU triangulation requirements, ought to be EUR), e.g. "eur.txt".
- An optional additional file, chained with a Next statement to the file containing the exchange rate data, can be used to store localized data such as currency names. We recommend to name the file using the same base name as the exchange rate data file, plus a suffix (separated with "-") indicating the two-letter language codes as per ISO 639-1 standard ("eur-en.txt", "eur-fr.txt", "eur-es.txt", "eur-de.txt", "eur-it.txt", etc.) This syntax allows calculator

clients to support the automatic loading of the proper data, based on the run time system language.

- The file containing EMU status updates, which is accessed at most once a week, should be named "emu.txt" (or "emu-en.txt", "emu-fr.txt", "emu-es.txt", "emu-de.txt", "emu-it.txt", etc.).

Configuration of the Calculator Client

Custom calculators with currency conversion functionality can be created either with Calculator Builder, or as a service provided by Currency System.

The redistributable calculator package includes a "custom.ini" INI file, which contains the URLs of the currency data files. These keys, located in the [Services] section, are Rates and EMU.

For example:

```
[Publisher]
Name         = "Our Company Name"
HomePage     = "https://www.example.com"
ServicesRoot = "https://rates.example.com/rates/"
Registerable = False
License      = "12345-12345-12345-12345"

[Services]
Level  = 3
Rates  = "en", "eur-en.txt"
EMU    = "en", "emu-en.txt"
Banner = "en", "banner-en.txt"

[Calculator]
Language        = "en"
ApplicationName = "en", "Currency Calculator"
ApplicationIcon = "icon.ico"
DefaultSkin     = "en", "CurrencyCalculator"
DefaultUnits    = "GBP", "EUR"
StartupMessage  = "Optional one-time message."
...
```

Additional calculator-specific information is included in the Calculator Builder documentation.

Configuration of the Server

The following steps assume the following typical configuration:

- Single (or automatically mirrored) web server
- Three INI files (Exchange Rate Data Header File, Exchange Rate Data File, EMU Configuration File)
- Exchange Rate Data Header File containing currency names in English, plus other static currency properties, as documented below
- Exchange Rate Data File updated by Currency Server via CurrencyUploader.exe
- EMU Configuration File updated by the administrator based on procedures documented below

To configure Currency Server to feed data to calculator clients:

1. On the computer running Currency Server, perform the steps described in Initial Configuration.
2. Define a local directory for log files. In this sample configuration, the directory will be assumed to be "C:\logs\currency_server".
3. In the **Notifications** tab of Currency Server Manager, enable **Log to file** and set the **Log File** to "C:\logs\currency_server\notifications.log" (without the quotes).
4. In the **Monitoring** tab, set the **Fluctuation** to a **Maximum** of 15% (or smaller), with a **Severity** of **Error**. This means that an administrator will be notified and will have to manually approve unusual fluctuations before they are uploaded to the server which feeds data to the calculator clients.

5. Identify the web URL and format of the exchange rate data. In this sample configuration, the URL will be assumed to be https://rates.example.com/rates/eur.txt, and the format will be a SERIFF INI file as written by CurrencyUploader.exe.
6. On the DNS server(s) for "example.com", add an IP Address (A) record for the host "rates" (you can normally use the same settings as those of the record for "www").
7. On the web server, create a directory (or virtual directory) "rates". Add an appropriate "index.html" or other default document file to the directory, or deny browsing, in case users try to access the directory manually.
8. Make sure that Currency Server can write to the "eur.txt" file. In this sample configuration, we will assume that the file is uploaded via FTP to "ftp://ftp.example.com/www/htdocs/rates/eur.txt", with a user name "myuid" and a password "mypw".
9. Install CurrencyUploader.exe on the computer running Currency Server (which is available from the Examples section on the Currency Server site) to the "Actions" subdirectory of the Currency Server installation directory. On some systems you may also have to install Microsoft's Web Publishing Wizard.
10. In the **Clients** tab of Currency Server Manager, set **Post-Update Actions** to "CurrencyUploader.exe ftp://ftp.example.com/www/htdocs/rates/eur.txt /u:myuid /p:mypw /l:C:\logs\currency_server" (without the quotes). Set **Execute** to **Only if rates changed.**
11. In the **FX Feeds tab** of Currency Server Manager, disable the **Auto-Update** option.
12. Close Currency Server Manager (which has exclusive write access to the currency data).
13. Set up a Windows Task Scheduler task to run **UpdateNow.vbs** every hour from Monday to Friday (or as appropriate). If desired, and if your FX feed(s) is known to update its rates only once a day (e.g. as central banks usually do), you can limit the updates to a range of hours following the known update time(s). It is still recommended to perform the upload attempts for 4-5 hours after the known release time of the data, both to take into account possible network connection problems, and in consideration of possible additional updates following the first update.
14. From the Currency Server icon in the notification area, run **UpdateNow.vbs**. As long as the rates changed since the previous update (if not, you can manually edit one exchange rate in the **Active Currencies** tab before invoking the update), Currency Server will fetch the new rates and then invoke **CurrencyUploader.exe**.
15. Open https://rates.example.com/rates/eur.txt with a web browser to verify that the data was uploaded properly.
16. From the Currency Server icon in the notification area, save the configuration settings for possible future use.

If data needs to be uploaded to multiple locations, multiple CurrencyUploader.exe instructions can be placed in a batch file, which can be invoked by Currency Server as a custom action instead of CurrencyUploader.exe itself.

The Exchange Rate Data Header File

In this sample configuration Currency Server writes the exchange rate data to a file named "eur.txt" in the "rates" directory of the web server. This is the Exchange Rate Data File, which however is not the file which is directly referenced in the calculator clients, and which instead is a header file named "eur-en.txt" (note the suffix used to indicate English language texts). Having the currency data in two files simplifies administration, because "eur-en.txt" contains localized data which normally needs to be set up only once (or when currencies are added, or currency names are edited), whereas "eur.txt" contains only the data which is updated and uploaded automatically by Currency Server.

For each currency which appears in the **Active Currencies** tab of Currency Server Manager (that is, after you completed the configuration, deciding which currencies to post exchange rates for), you need to list the currency names and the smallest units, as they appear in the corresponding Currency Server properties. This is because the calculator clients rely on this external data, which is not built-in (in Currency Server it is both built-in and updated).

Your "eur-en.txt" file will then look like the following (note the Next key pointing to the actual Exchange Rate Data File):

```
[General]
Source = "Rates provided by Our Company..."
Next = "https://rates.example.com/rates/eur.txt"

[Names]
```

```
AUD = "Australian Dollar"
BGN = "Bulgarian Lev"
CAD = "Canadian Dollar"
CHF = "Swiss Franc"
CNY = "Chinese Yuan Renminbi"
CYP = "Cypriot Pound"
CZK = "Czech Koruna"
DKK = "Danish Krone"
EEK = "Estonian Kroon"
GBP = "British Pound"
HKD = "Hong Kong SAR Dollar"
HUF = "Hungarian Forint"
ILS = "Israeli New Shekel"
INR = "Indian Rupee"
ISK = "Icelandic Krona"
JPY = "Japanese Yen"
KRW = "South Korean Won"
LKR = "Sri Lanka Rupee"
LTL = "Lithuanian Litas"
LVL = "Latvian Lat"
MTL = "Maltese Lira"
MXN = "Mexican Peso"
MYR = "Malaysian Ringgit"
NOK = "Norwegian Krone"
NZD = "New Zealand Dollar"
PLN = "Polish Zloty"
ROL = "Romanian Leu"
RUB = "Russian Ruble"
SEK = "Swedish Krona"
SGD = "Singaporean Dollar"
SIT = "Slovenian Tolar"
SKK = "Slovak Koruna"
TRL = "Turkish Lira"
USD = "US Dollar"
ZAR = "South African Rand"

[Smallest]
AUD = 0.01
BGN = 0.01
CAD = 0.01
CHF = 0.01
CNY = 0.01
CYP = 0.01
CZK = 0.01
DKK = 0.01
EEK = 0.01
GBP = 0.01
HKD = 0.01
HUF = 0.01
ILS = 0.01
INR = 0.01
ISK = 0.01
JPY = 1
KRW = 1
LKR = 0.01
LTL = 0.01
LVL = 0.01
MTL = 0.01
```

```
MXN = 0.01
MYR = 0.01
NOK = 0.01
NZD = 0.01
PLN = 0.01
ROL = 0.01
RUB = 0.01
SEK = 0.01
SGD = 0.01
SIT = 0.01
SKK = 0.01
TRL = 1
USD = 0.01
ZAR = 0.01
```

In this example, English title style (capital initials) was been used for the currency names, because this is the format used for drop-down menu and similar texts as they are displayed in the calculator user interface. Different style preferences may be appropriate for other languages and regions.

If for example you need to post the data in French, or German, you could have similar files named "eur-fr.txt" and "eur-de.txt", with texts in French and German, respectively.

The [General] section can also include additional keys, e.g. to display a text message (e.g. Message = "Hello World", or Message.*yourversionnumber* = "A new version of the software is available") or to control an optional banner in the calculator client.

The Exchange Rate Data File

The following is an example of Exchange Rate Data file written by CurrencyUpload.exe to "eur.txt" (which is included by reference in "eur-en.txt"):

```
[General]
Application = "Cloanto(R) Currency Server(TM)"
PublicationTime = 2006-07-30T17:46:03Z

[EUR]
AUD = 1.7373
BGN = 1.9466
CAD = 1.5922
CHF = 1.5494
CNY = 9.40923
CYP = 0.58754
CZK = 31.983
DKK = 7.4312
EEK = 15.6466
GBP = 0.7044
HKD = 8.9065
HUF = 263.38
ILS = 5.0256
INR = 52.408
ISK = 87.58
JPY = 136.97
KRW = 1347.45
LKR = 110.34
LTL = 3.4532
LVL = 0.6504
MTL = 0.428
MXN = 11.901
MYR = 4.3201
NOK = 8.2065
NZD = 1.9526
```

```
PLN = 4.374
ROL = 37247
RUB = 34.402
SEK = 9.178
SGD = 2.0058
SIT = 234.66
SKK = 41.835
TRL = 1631000
USD = 1.142
ZAR = 8.4333
```

The EMU Configuration File

When the calculator client is released for distribution in the latest version released by Currency System, it normally has built-in information about the status of the currencies of the EMU (European Economic and Monetary Union). Also with special consideration to the fact that the calculator may be distributed as an EMU-related tool (for example, in countries where the euro is next to replacing the local currency), it is both important that the EMU data be updated as appropriate, and also, like the exchange rate feed, this adds value to the calculator client.

The EMU Configuration File is an INI file which contains:

- a list of active EMU currencies, which are listed in the [EUR] section together with their official conversion rate;
- a list of EMU currencies which ceased to be legal tender (having been replaced by the euro), listed in the [EUR-Replaced] section together with their official conversion rate;
- a list of smallest units for each currency, indicating the smallest unit for rounding purposes (e.g. 0.01 = one cent, 0.50 = 50 cents, etc.), which usually is either the smallest coin for the given currency or the smallest unit for accounting purposes, if different.

When the user clicks **Reset EMU** in the calculator client, all of the above data is taken into account to reset the information of EMU currencies. Additionally, if information for both legal tender and non-legal tender currencies is available, the user can choose whether to include currencies which ceased to be legal tender.

Once a week, and only during exchange rate data updates, the calculator client also loads the EMU Configuration File and verifies whether there have been changes in the file. If the information has changed the user is informed and reminded that the **Reset EMU** button can be selected to apply the changes.

As of 2003, a sample emu-en.txt file is:

```
[General]
PublicationTime = 2002-02-28T23:00:00Z

[EUR]
EUR = 1.00000

[EUR-Replaced]
DEM = 1.95583 ; 2002-01-01 00:00
NLG = 2.20371 ; 2002-01-29 00:00
IEP = 0.787564; 2002-02-10 00:00
FRF = 6.55957 ; 2002-02-18 00:00
ATS = 13.7603 ; 2002-03-01 00:00
BEF = 40.3399 ; 2002-03-01 00:00
ESP = 166.386 ; 2002-03-01 00:00
FIM = 5.94573 ; 2002-03-01 00:00
GRD = 340.750 ; 2002-03-01 00:00
ITL = 1936.27 ; 2002-03-01 00:00
LUF = 40.3399 ; 2002-03-01 00:00
PTE = 200.482 ; 2002-03-01 00:00

[Names]
```

```
ATS = "Austrian Schilling"
BEF = "Belgian Franc"
DEM = "German Mark"
ESP = "Spanish Peseta"
FIM = "Finnish Mark"
FRF = "French Franc"
GRD = "Greek Drachma"
IEP = "Irish Pound"
ITL = "Italian Lira"
LUF = "Luxembourg Franc"
NLG = "Netherlands Guilder"
PTE = "Portuguese Escudo"
EUR = "EU Euro"

[Smallest]
ATS = 0.01
BEF = 0.50
DEM = 0.01
ESP = 1
FIM = 0.01
FRF = 0.01
GRD = 1
IEP = 0.01
ITL = 1
LUF = 0.50
NLG = 0.01
PTE = 0.10
EUR = 0.01
```

In the above example the only currency which is part of the EMU is the euro itself. Both the [General] section and the comments (data after ";") are optional.

As new currencies join the EMU, they are listed in the [EUR] section. Once they complete the transition phase, they are moved from [EUR] to [EUR-Replaced], from where they are removed only in the unlikely event of the currency abandoning the EMU.

Maintenance Procedures

Maintenance of a client-server configuration where both Currency Server and calculator clients are deployed is covered in general by the Currency Server Operational Procedures. Additionally, the following special steps have to be performed:

- When a currency joins the EMU, its official rate and smallest unit have to be included in the [EUR] and [Smallest] sections of the emu-en.txt file, respectively. For the smallest unit, remember to use trailing zeros to indicate precision (e.g. if for example the smallest unit is set to 0.50, an amount of 3.51 would be rounded to 3.50, whereas if the smallest unit is set to 0.5, then the rounded amount would be 3.5).
- When a currency has completed the euro transition phase, its rate has to be moved from [EUR] to [EUR-Replaced].
- If a currency abandons the EMU (which is considered a very unlikely event), both its rate and its smallest unit have to be removed from the emu-en.txt file.
- After each of the above changes, update the emu-en.txt file on the web server and verify the functionality with a calculator client (you can delete the EMULoadSuccess and EMULoadFailure values in the calculator registry settings to force the emu-en.txt file to be fetched at the next exchange rate data update).

Related Topics

- For more information about configuration, see Initial Configuration.
- For more information about best practices, see Quality Checklist and Operational Procedures.

Web Links

- For more information about the INI file format, see Cloanto Implementation of INI File Format.
- For more information about the SERIFF specification, see SERIFF (Simple Exchange Rate Information File Format).

Chapter 8

8 Additional Resources

This section covers:

- Web Resources
- Advanced Registry Settings
- Privacy Information

8.1 Web Resources

You may find the following resources on the Currency System website of interest:

- Currency Server Homepage
- Frequently Asked Questions
- Providers of Exchange Rate Data
- Code Samples and Source Code
- Troubleshooting

Related Topics

- For more information about currency-related topics, see Working with Currencies and Recommended Reading.

8.2 Advanced Registry Settings

Overview

Currency Server stores its settings and exchange rate information at the following location in the system registry:

HKEY_LOCAL_MACHINE\SOFTWARE\Cloanto\Currency Server\[VersionNumber]

The data which is stored in the registry may include a few additional, advanced options with respect to those which are normally available through Currency Server Manager. Depending on feedback, in a future version of the software these settings may or may not be included in Currency Server Manager, but in any case they are guaranteed to be supported in the future.

Warning

The following sections contain information about editing the registry. You should be very careful not to modify parts of the registry which are unrelated to the topics described here.

If you do not know how to access or edit the registry, please consult the Windows documentation and Help topics. For information about how to edit the registry, you can view the "Changing Keys and Values" Help topic in Registry Editor (Regedit.exe) or the "Add and Delete Information in the Registry" and "Edit Registry Data" Help topics in Regedt32.exe. Before you edit the registry, make sure you understand how to restore it if a problem occurs. For information about how to do this, view the "Restoring the Registry" Help topic in Regedit.exe or the "Restoring a Registry Key" Help topic in Regedt32.exe.

Using Registry Editor incorrectly can cause serious problems that may require you to reinstall your operating system. It cannot be guaranteed that problems resulting from the incorrect use of Registry Editor can be solved. You edit the registry at your own risk. You should consider backing up the registry and/or creating a System Restore Point before making changes to the registry.

ConflictingPegged (DWORD)

This setting specifies how to deal with currencies which are pegged to another currency at a constant rate set in the software when a different value is provided during an update. The possible settings are 0 (Use internal rates without warning), 1 (Use internal rates and warn) and 2

(Accept rates without warning). Default value: 0 (internal rates are used without warning).

DebugLogFile (String)

Based on technical support requirements, the software may be made available in a special version which includes a tracing option to log certain steps to a file. To enable logging set the string to a complete file path (e.g. "C:\CS_Log.txt").

IgnoreStaticUpdates (DWORD)

Currency Server contains hard-coded data (current at compile time) about the codes, names and other static properties of all currencies of the world, including European Economic and Monetary Union currencies (e.g. which currencies have joined the EMU, or which currencies have been replaced by the euro) and their constant conversion rates. This data is independent from the exchange rate data, which is usually updated at least daily. From time to time, and only when already online, the software may also download up-to-date information provided by Currency System as part of its Currency World Monitor service to update the static data. If this value is set to 1, online updates are ignored. Default value: 0 (the software uses both the built-in data and the online updates).

LogFXData (DWORD)

This setting specifies whether Currency Server should keep a detailed log of the currency exchange rate data that is fetched from the FX servers. The possible settings are 0 (never log), 1 (log only if errors or warnings) and 2 (always log). Default value: 1 (log FX data only in relation to update errors or warnings).

NotificationAreaIcon (DWORD)

This setting controls the presence of the Currency Server icon in the notification area. A new logon is required for changes to this setting to be applied. Default value: 1 (icon is displayed).

Separator (String)

This is the separator character exposed in the Application.Separator property and used to separate multiple currency codes or other fields when they are merged into one string. It defaults to a semicolon (";").

UserData2 (String)

This value contains the private run-time replacement string which is referenced by the special text "[2]" which may be contained in the Address key of an FX feed filter INI file.

Web Links

- Editing the Windows Registry on the Currency System website.

8.3 Privacy Information

The following is a copy of the privacy information which is displayed as part of the Currency Server setup procedure.

Internet Clients and Servers

When Currency Server retrieves data from remote servers it may act as an "internet client", and it may cause accesses to be logged by such "internet servers", as is commonly done when web browsers and other internet clients are used. Access logs typically include the IP address of the internet client, the time of the access, the address of the data which was accessed, and information provided by the client software, which typically includes the client software and operating system name and version number.

Internet Access

Currency Server never establishes an internet client connection unless a currency exchange rate

data update was either scheduled (automatic updates) or explicitly requested (manual updates). By default, automatic updates are disabled.

Territory

When connecting to a global computer communication network such as the internet, data may cross and be processed and stored in several countries and jurisdictions.

Cookies

Currency Server makes use of standard internet access functionality provided by the operating system, which supports "cookies". Currency Server does not require cookies to operate, however when Currency Server acts as an internet client and the internet server requests that a cookie be written or read, Currency Server provides the requested functionality.

Other Information Transmitted by Currency Server

Currency Server was designed to access data (e.g. currency data) which may not be available to the general public for universal, anonymous access. Such data may be provided as part of a subscription service provided to a specific individual or other legal entity, requiring certain credentials to be transmitted as part of the request to access said data, which may result in the identification of said individual or other legal entity.

The automatic transmission of such credentials using either plain text or encrypted (e.g. SSL or Windows NT Challenge/Response) transmission protocols is supported by the software. Specifically, Currency Server supports the transmission of any combination of the following fields, previously entered by the user, if requested by the server or otherwise defined in the data access protocol (names are quoted as they appear in the software user interface and documentation):

1. "User name" and "Password";
2. "User ID";
3. One or more "License Keys".

Providers of data may use this information, entered by the user and transmitted by Currency Server, alone or in conjunction with other internet client information, to restrict access and/or for license compliance purposes.

Access to Currency Data Provided by Currency System

The information contained in the introduction and in the sections titled "Internet Clients and Servers", "Internet Access", "Territory", "Cookies" and "Other Information Transmitted by Currency Server" also applies to connections to servers operated by Currency System.

More specifically:

1. By default, the Currency Server software accesses non-public currency data provided by Currency System;
2. Currency System is also the publisher of the Currency Server software, which is protected by copyright and other intellectual property laws and international treaties, and is licensed in accordance with the terms of an EULA;
3. Currency System may process or store information in the United States, in the European Union and in any other country;
4. The default authentication mechanism employed by Currency System to grant access to the currency data consists of the transmission of one or more License Keys, previously entered in the software;
5. The License Key(s) may identify the licensee(s) of both the data feed service and the software;
6. Currency System may use this information, entered by the user and transmitted by Currency Server, alone or in conjunction with other internet client information, to restrict access and/or for license compliance purposes.

When Currency Server is offered by a third-party redistribution partner the above defaults and authentication mechanism details may not apply, i.e. it may be possible that no data is ever sent to Currency System.

Additional Information

The Currency Server documentation contains additional information about the software configuration options and the currency data provided by Currency System.

Both Currency System and other data providers may provide additional information about their privacy policies on their respective websites.

Should you have any privacy-related questions, please refer to the contact information provided at currencysystem.com/privacy/.

Related Topics

- For more information about software user and license information, see The Software Tab.
- For more information about server user information and access credentials, see The FX Feeds Tab.
- For more information about network access credentials, see The Connection Tab.