Using the MultiValue Features of Caché

Version 2018.1
2018-12-07

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using the MultiValue Features of Caché
Caché Version 2018.1 2018-12-07

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

About This Book

This document provides basic information on installing Caché and getting started with the MultiValue (MV) shell.

This book contains the following sections:

- Introduction and Overview
- Installing Caché
- Starting MultiValue
- Using a Caché Class and MultiValue: A Sample Session
- Available Flavors / Emulations
- Configuration Options
- Frequently Asked Questions about MultiValue and Caché

There is also a detailed Table of Contents.

For general information, see *Using InterSystems Documentation*.

1

Introduction and Overview

Caché provides MultiValue developers with access to its application development and system management features. This set of features provides a native MultiValue environment as part of Caché. It also allows existing MultiValue applications to take advantage of the features of Caché, including:

- Caché objects
- Both MultiValue queries and SQL queries
- Studio, an integrated development environment
- Caché Web-based features, including Caché Server Pages (CSP) and Web services generally
- Browser-based system management
- Security management

2

Installing Caché

This chapter covers the basics of installing Caché for those who are new to InterSystems and its products. To find out if the system you intend to use is one that InterSystems supports for Caché, please consult the online *InterSystems Supported Platforms* document for this release. This lists the details of hardware and software requirements for successful operation.

A full description of the Caché installation process for experienced (and/or more curious) users is available in the Caché Installation Guide. There are separate sections for Windows, and UNIX®/Linux/macOS.

If you are new to Caché, there are some things to keep in mind about installation:

- It is possible to install multiple copies of Caché on a given platform and have them operate independently of one another. In this case, the individual installations are referred to as "instances".
- Each instance of Caché must be installed in a location ("destination folder") separate from all the others.
- Each instance of Caché communicates with the outside world over two TCP/IP ports, (the "superserver port" and the "webserver port"). The installer for your platform will choose available unused ports. Each can be overridden, but in most cases it is not necessary.
- Caché provides security facilities that can be tuned to the level of protection needed by a particular installation. You can defer learning about the details of security by choosing "Minimal" as the level of security for your MV installation.

These instructions assume that you have obtained a CDROM with a version of Caché that will run on your system. If you are new to Caché, you are encouraged to review the introductory material on Caché noted previously. Installation instructions for specific platforms can be found in the Installation Guide for:

- Windows
- Linux and UNIX®
- Macintosh

Note on Files Overwritten During a Caché Upgrade

If you are upgrading your Caché installation, note that the following Multivalue files are overwritten: ERRMSG, TERMDEFS, TERMCMP, NEWACC, DICT.DICT, DICT SYSTEM (except the data section, which is not overwritten), DICT MVPIDTAB (data section is cleared on every startup,

In addition, VOC items might be overwritten during an upgrade.

The VOC for each account is originally created as a copy if the NEWACC file. There is an item RELLEVEL which identifies the release. Upon entering the MV shell, the RELLEVEL item in the VOC is compared to the RELLEVEL in NEWACC, and if they are different, the contents of NEWACC overwrite items with the same name in VOC. Existing VOC items not in NEWACC are not affected, but any changes to supplied commands (for example if you have replaced the VOC QUIT command to do something different when logging out) are replaced.

3

Starting MultiValue

There are two principal paths to accessing the Caché MultiValue shell: via a built-in terminal emulator (on Windows systems only), or via a telnet connection.

3.1 Connecting Via the Terminal on Windows

Once Caché is installed and operational, you can run the MultiValue shell via a Terminal session. To do this:

3.1.1 Starting a Terminal Session

Click on the Caché icon in the System tray. Choose **Terminal** from the menu displayed. A new terminal session starts and displays a window in which to enter commands. Depending on the security level of your Caché installation, you may have to enter your username and password to get to the "USER>" prompt where you can begin entering commands.

3.2 Connecting to Caché Via Telnet

The MultiValue shell can also be run via a telnet connection using your favorite terminal emulator. By default, Caché is installed with its telnet capability disabled; the instructions to enable telnet are given in the appendix.

3.3 Starting the MV Shell

Regardless of how you connect to Caché, in the terminal window at the "USER>" prompt enter the command:

```
MV
```

followed by **ENTER**.

If this is the first invocation of **MV**, it will perform some initialization. For example, **MV** will create the SYSPROG and USER accounts automatically. Once initialization is complete, **MV** displays the Caché version number and date of installation, a message indication how the MultiValue Shell handles two-digit years, and then the colon (:) prompt. For further details, refer to the MV command in the *Caché ObjectScript Reference*.

Invoking the MultiValue Shell causes Caché to search the VOC for items in the following order: an item with the same name as the user, an item with the same name as the account, or an item named "LOGIN". The MultiValue Shell runs the first of these items that it encounters.

You may begin working in the default account just set up. Or, you may create one or more other accounts using the MultiValue CREATE.ACCOUNT command, specifying which MultiValue flavor you wish to use, for example,

```
CREATE.ACCOUNT DEMO <flavor>
```

where <flavor> is the name of the flavor (emulation type) to use as the default for all processes accessing that account. The complete list of available flavors is in the Appendix A.

3.3.1 Exploring MultiValue on Caché

Once the shell is running, you may explore the MultiValue environment. Other documents will help you import your applications and data, as well as expand your capabilities to include the features that Caché provides.

The Q command causes the current terminal process to exit the MV Shell. Once you exit the MV Shell, you are returned to the Caché mode prompt of your terminal or telnet session. To finish a session, you should **QUIT** from your telnet session, or issue the **HALT** command if you are in a Terminal session.

3.4 Importing Data and Programs into Caché

Note: At present, Caché provides support for importing backups made on UniVerse, D3, jBASE, Reality, MVBase, and POWER95 systems. For other MultiValue systems, InterSystems provides example export and import programs that can be modified as needed by the user to transfer the relevant data to Caché.

3.4.1 Using MVIMPORT

There is an implicit assumption that importing applications and/or data is an administrative function that should be done under the SYSPROG account. From the MV Shell you can LOGTO the SYSPROG account, then enter the command:

```
MVIMPORT   <BackupFilePath>
```

where BackupFilePath is the location of the backup file. It must be supplied in machine-specific format. For example, on a Windows system it would be something like:

```
MVIMPORT   C:\MVTEST\Acct_Main\Backup.uvb
```

MVIMPORT will read the backup file, create the necessary dictionary and files entries, and store the programs and data there. The **MVIMPORT** command is described in the "MVIMPORT" chapter of the *Caché MultiValue Commands Reference*. Also see the Operational Differences between MultiValue and Caché document for details on **MVIMPORT**.

3.5 Notes on the MV Shell

This section provides additional information related to interacting with Caché from the MV shell.

3.5.1 Issuing MVBasic Commands in the Shell

The MV shell provides the ability to execute MVBasic commands directly by preceding them with a semicolon. For example,

```
; PRINT  "Hello, World"
```

will result in “Hello, World” being echoed to the terminal window.

For further details, refer to the Caché MultiValue Basic Reference.

3.5.2 Issuing ObjectScript Commands from MV

The MV shell provides a way to issue ObjectScript commands without having to terminate shell operation. The COS command sends the remainder of its line to Caché for execution. For example, at the MV prompt, the command:

```
COS WRITE "Hello, World", !
```

writes the expected string to the output device followed by a newline. Since multiple Caché commands may be placed on a line (separated by single blank spaces), a single invocation of **COS** may execute multiple Caché commands.

The left square bracket “[” command is a synonym for **COS**.

3.5.3 Command History Editor

Within the MV Shell, it is possible to recall previously entered commands, perform simple edit operations on them and submit the edited version as a new command. This is similar to the functionality provided by the UniVerse TCL stacker. All the commands that operate in this fashion begin with a period (.). The available commands are:

Command	Description
.A{nn} string	Append the characters of string to the end of stack number 'nn'
.C{nn}/From/To	Change “From” to “To” in string 'nn'
.DNN{-mm}	Delete stack 'nn (possibly through stack 'mm')
.L{nn}	Display the last 'nn' lines of the stack
.U{nn{-mm}}	Convert stack 'nn' (through stack 'mm') to uppercase
.X{nn{-mm}}	Execute stack 'nn' (through 'mm')
.?	Display a summary of the available commands

For further details, refer to the MultiValue Command Stack Commands chapter of the *Caché MultiValue Commands Reference*.

3.5.4 Starting Directly In the MV Shell

Caché provides the capability for an MV user to start up in the MV shell. This is done by modifying the definition of the user to specify a namespace and initial routine that will be executed when the login is complete.

Note: In order to modify the definition of a user, you must be an administrator. That is, you must be a member of a role that has **`%Admin_Secure`** or **`%All`** privileges. If you installed Caché using the “Minimal” security setting as directed, you should automatically have the proper permission.

If you have the proper authorization, you can modify the user definition via the Management Portal page **[System] > [Security Management] > [Users]**. Select the user you wish to modify and select the "Edit" option. When the **Edit User**, page appears, fill in the **Default Namespace** text box with the namespace you wish the user to be in when the MV shell starts. Enter "^%SYS.MV" into the **Default Tag^Routine** box as the initial Caché command to be executed when login succeeds.

4

Using a Caché Class and MultiValue: A Sample Session

Caché comes with the file MVDEMO.PERSON.xml, which contains the MVDEMO.PERSON class. This section provides a sample session of loading and manipulating this class. Its steps provide examples of the MultiValue features in Caché, and how the native Caché environment interacts with these features.

The steps are:

1. Starting the MV shell.
2. At the MV shell prompt, importing the class definition.
3. Using MVBasic to perform basic object manipulation: instantiation, setting some properties, and saving an instance.
4. Examining the data in the MV shell with a simple MV query.
5. Using the Caché data population utility to instantiate a larger number of objects.
6. Running more complex MV queries against the data.
7. In the Management Portal, running SQL queries against the data.
8. Using Studio to examine the class definition.

In addition to supporting traditional MVBasic file manipulation (such as with OPEN, WRITE, and so on), Caché allows you to treat data as objects. This sample highlights the use of Caché object features.

4.1 Starting the MV Shell

In a Terminal window, make sure you are in the USER namespace; this is indicated by a prompt of USER>. A Caché namespace is similar to a MultiValue account.

To go from another namespace to the USER namespace, use the ZNSPACE (abbreviated **ZN**) command — in this example, from the SAMPLES namespace:

```
SAMPLES>ZN "USER"
USER>
```

Then start the MV Shell by issuing the MV command:

```
USER>MV
```

More on the MV Shell can be found in the Operational Differences documentation.

4.2 Importing the Class Definition

The definition of the MVDEMO.PERSON class is in the file MVDEMO.PERSON.xml. To import the class definition into Caché, use the **Load** method of the %SYSTEM.OBJ class in the MV shell:

```
USER: ; "%SYSTEM.OBJ"->Load(<class definition file>, "c")
```

where

- The leading semicolon (;) in the MV shell specifies that the rest of the line is MVBasic code (not an MV command). The Cache MV shell has been enhanced to allow for lines of MVBasic code to be executed from the MV shell.
- The first argument specifies the quoted name of a XML class definition file, including a full path. The location of this file relative to the Caché installation directory is <cache-installation-dir>\Dev\mv\samples/MVDEMO.PERSON.xml (Windows) or <cache-installation-dir>/dev/mv/samples/MVDEMO.PERSON.xml (UNIX®, Linux, MacOs).
- The second argument specifies the required compiler flags ("c" means compile).

Note: All examples of invoking commands from the MV shell in this chapter include the MV shell prompt ("USER: ").

When you run this command, a series of output statements appear in the Terminal indicating that the file has been imported and compiled.

For a standard Windows installation, the following will do the import:

```
USER: ; demo = "C:\InterSystems\Cache\Dev\mv\samples\MVDEMO.PERSON.xml"
USER: ; stat = "%SYSTEM.OBJ"->Load(demo, "c")
```

When you run this, a series of output statements appear in the Terminal indicating that the file has been imported, compiled, and exported to a MultiValue file.

With the MV shell, you can specify:

- MV shell commands.
- MultiValue Basic (MVBasic) commands. Lines containing MVBasic commands begin with a semicolon.
- ObjectScript calls. Lines containing ObjectScript begin with a leading "COS" or a leading left bracket ("[").

4.3 Creating, Manipulating, and Saving an Object

Once the compiled class is available, you can invoke its **CreateSome** method:

```
USER: ; "MVDEMO.PERSON"->CreateSome()
```

This command invokes the **CreateSome** method of the MVDEMO.PERSON class. This method performs the following actions:

- Creates an instance of MVDEMO.PERSON
- Sets the values for the properties of that instance.

- Saves the instance.
- Creates ten other instances of the class using the Caché data population utility (described in the following section on populating the class with test data).

4.4 Running Some Simple MV Queries against the Class

Since there are now instances of the class, you can use the available Caché tools to run queries against them. Standard MV shell commands are available in the Caché MV shell, so it is easy to run an MV query:

```
LIST MVDEMO.PERSON Name
```

As expected, this query displays the value of Name and ID of each instance of the class. For example, output from this command might look like:

```
LIST MVDEMO.PERSON Name                     02:09:27pm  26 Jul 2006  PAGE    1
ID........    Name.....................

25018         Smith,John
25019         Ott,Barbara T.
25020         Vonnegut,Maria Y.
25021         Sands,Roger J.
25022         Wakefield,Phil U.
25023         Ortiz,Mario O.
25024         Finn,Bob Z.
25025         Jones,Chris K.
25026         Yeats,Phil H.
25027         Goldman,Phil O.
25028         Ng,Valery W.

11 Items listed.
```

Note: Because the populate utility generates data randomly, the results of this command will differ from what appears here.

A query can also refer to collection properties, such as this class' Phone property:

```
LIST MVDEMO.PERSON Name Phone               02:15:04pm  26 Jul 2006  PAGE    1
ID........    Name.....................    Phone.......

25018         Smith,John                  555-123-1234
                                          555-432-4321
25019         Ott,Barbara T.              460-920-4165
                                          661-911-2114
                                          650-886-1018
25020         Vonnegut,Maria Y.           874-843-5492
                                          294-588-4007
25021         Sands,Roger J.              781-540-8221
                                          897-722-5975
25022         Wakefield,Phil U.           635-827-9697
25023         Ortiz,Mario O.              229-472-7806
                                          989-512-1785
25024         Finn,Bob Z.                 214-929-6674
                                          483-685-7455
25025         Jones,Chris K.              874-204-8174
25026         Yeats,Phil H.               345-959-9979
25027         Goldman,Phil O.             423-344-9508
                                          533-923-2919
                                          481-245-8553
25028         Ng,Valery W.                479-793-4879
```

4.5 Populating the Class with Test Data

Because MVDEMO.PERSON is a subclass of the %Library.Populate class (often known simply as %Populate), it has built-in functionality for generating test data. Running the **CreateSome** method creates ten instances of the class using this functionality.

To create one hundred thousand instances of the class, the command is:

```
USER: ; PRINT "MVDEMO.PERSON"->Populate(100000)
```

This code uses a number of special features of the Caché MV shell:

- The leading semicolon (;) specifies that the subsequent code is an MVBasic command.
- The quoted string specifies the name of the class being invoked.
- The arrow syntax "`->`" allows you to refer to a property or method of a class. (If a property is itself an object, this syntax may appear more than once, for example in syntax such as `"MYAPP.EMPLOYEE"->SPOUSE->BIRTHDAY`.
- Since the **Populate** method returns the number of instances it populated, the MVBasic **PRINT** command uses this value as an argument.

Regarding the call itself, the data population utility generates test data for the class properties specified in the %Library.PopulateUtils class. For more information, see the chapter "The Caché Data Population Utility" in the book *Using Caché Objects*.

4.6 Running Other MV Queries

More sophisticated queries are also available. These may be more useful when there are larger amounts of data.

For example, queries can attempt to match parts of strings:

```
LIST MVDEMO.PERSON Name Phone WITH Name LIKE Yeats,Qui...
                                          02:23:27pm  26 Jul 2006  PAGE    1
ID........    Name.....................    Phone.......

26040         Yeats,Quigley S.            201-507-5264
                                          946-663-7004
                                          348-807-3906
41436         Yeats,Quigley A.            859-639-9381

2 Items listed.
```

They can perform multiple matches, with exact and partial matches:

```
LIST MVDEMO.PERSON Name WITH Hair = brunette AND Name LIKE Presl...
                                          02:27:20pm  26 Jul 2006  PAGE    1
ID........    Name.....................

25070         Presley,Mary N.
26245         Presley,Michael M.
26307         Presley,Orson Q.
26455         Presley,Amanda R.
28654         Presley,Chad L.
28727         Presley,Thelma Q.
29922         Presley,Clint B.
30459         Presley,Richard B.
30799         Presley,Filomena M.
31313         Presley,Wilma E.
...
```

They can suppress the display of the ID for each row:

```
LIST MVDEMO.PERSON Name WITH Hair = brunette AND Name LIKE Presl... ID.SUP
                                          2:28:29pm  26 Jul 2006  PAGE    1
Name.....................

Presley,Mary N.
Presley,Michael M.
Presley,Orson Q.
Presley,Amanda R.
Presley,Chad L.
Presley,Thelma Q.
Presley,Clint B.
Presley,Richard B.
Presley,Filomena M.
Presley,Wilma E.
...
```

And they can perform counts:

```
USER:COUNT MVDEMO.PERSON WITH Hair = brunette

 8391 Items counted.
```

4.7 Using the Management Portal to Run SQL Queries

Because Caché provides full support for SQL, you can invoke SQL queries against your data. The Caché Management Portal provides functionality for running ad hoc queries. To run an SQL query:

1. From the Caché cube menu, choose **Management Portal**. If you have performed a minimal-security installation of Caché, this displays the main page for the Management Portal; with Normal or Locked-Down installations, a login screen appears.
2. On the Portal's main page, choose **SQL** from the middle column (**Data Management**). This displays the Portal's **SQL** page.
3. Choose the USER namespace from the **Namespace** list on the left.
4. Choose **Execute SQL Statement** from the left column (**SQL Operations**). This displays the **Execute SQL Query** page.
5. On this page, Caché allows you to create and run standard SQL queries against your data. You can either write a query in the available editing area or click the **Query Builder** button to use the interactive **SQL Query Builder**.

SQL access is also available via any ODBC or JDBC tool.

4.8 Using Studio to Examine the Class Definition

Having previously imported the class definition, you can view and edit it in Studio. Studio is a full-featured development environment. To display the class definition in Studio, the process is:

1. From the Caché cube menu, choose **Studio**. A login screen appears. With a minimal-security installation, simply click **OK** and Studio runs under the UnknownUser account, which has all privileges in this type of installation; with normal and locked-down installations, enter a valid username and password.
2. You should be in the USER namespace. To determine if this is so, check the string that appears in the Studio window's title bar. This is of the form:

   ```
   Studio-<logged-in user>-<Caché instance>/<active namespace>-<project>
   ```

   such as

   ```
   Studio-UnknownUser-CacheMVTest/USER-Default_
   ```

3. If you are not in the USER namespace, select **File-Change Namespace** (or **F4**). This displays the **Caché Connection Manager** screen. Choose USER from the list and click **OK**.
4. Open the MVDEMO.PERSON class. To do this:
   - Select **File-Open** from the menu.
   - Press **Ctrl-O**.
   - Click the Open icon from the menu.

   Any of these actions displays the **Open** dialog for the USER namespace.
5. In the Open dialog, look only for class definitions by specifying Class Definition (*.cls) in the Files of type field.
6. Select MVDEMO from the available choices and the PERSON.cls from the choices that then appear.

For more information on Studio, see the book *Using Studio*.

A

Available Flavors / Emulations

The following is a list of the flavors (or emulations) that are accepted by Caché. The emulation setting for an account can be changed via the CEMU command.

Flavor	Interpretation
CACHE	Our own Caché mode — the default if no emulation is specified
D3	D3 mode — D3 is the latest incarnation of Pick
DEFAULT	Synonym for CACHE
IN2	Intertechnique mode
INFORMATION	Synonym for PRIME
JBASE	jBASE Ideal mode
MVBASE	MV Base, a GA system now owned by Raining Data
PICK	The UniVerse implementation of PICK. Very similar to standard PICK, with minor differences
PIOPEN	PI/Open
PRIME	The basis for UniVerse and UniData, but is subtly different
R83	Original Pick R83 mode to which many applications are programmed
POWER95	Reality R95 - R83 plus a number of enhancements
REALITY	Reality Operating System
UDPICK	UniData running in Pick mode
ULTIMATE	Ultimate Pick and Ult+
UNIDATA	UniData Ideal mode
UNIVERSE	The UniVerse Ideal

Note: While all of these flavors are accepted, InterSystems recommends that you use the Caché version, if possible. Flavor-dependent behavior for commands, functions and so on, has been most completely implemented for UNIVERSE and JBASE.

InterSystems has been focused most tightly on supporting the flavors listed above. If you are porting to Caché from a MultiValue implementation which is not one of those listed, you may find that commands, options, conversion codes, and so on specific to your MultiValue dialect are not supported. In these instances, InterSystems recommends that you replace them with equivalents from one of the supported flavors.

B

Configuration Options

B.1 Controlling Caché Startup Behavior On Windows

Whether or not there is an icon in the system tray, you can always control a Caché instance via the **Start**, **Programs** and **Caché** menu choices.

Alternatively,

B.1.1 Change Caché Start To Manual

From the **Start** menu, select **Programs**, **Administrative Tools**, then **Services**. On some systems, **Administrative Tools** is found in **Start**, **Settings**, **Control Panel**.

From the Service window, select the item called, "Caché Controller for <instance name>", where <instance name> is the name you assigned to the Caché you just installed.

From the **Action** menu, choose **Properties**, and change the Startup Type to Manual. Click **OK**.

B.1.2 Remove The Cube From The System Tray

You can choose to temporarily or permanently remove the Cube from the system tray. For information, see Removing the Cube from the System Tray in the "Using Caché on Windows" chapter of the *Caché System Administrator's Guide*.

B.2 Enabling Telnet

Caché has the capability of communicating with a terminal shell via the telnet protocol. However, when Caché is first installed, the telnet facility is turned off as a security measure. Telnet can be turned on by following these steps:

B.2.1 Start the Management Portal

Caché is administered via a browser-based interface known as the Management Portal. Make sure the Caché instance you installed is running. Then, start the portal.

B.2.1.1 On Windows Systems

Left-click on the Caché Cube in the system. From the menu, choose **Management Portal**.

B.2.2 Configure Telnet

When installed, Caché assumes telnet communications will take place on the default port, 23, specified by the Internet Assigned Number Authority (IANA). If you wish to change this port assignment, choose **Configuration** from the System Administration column. In the following page, from the **Additional Settings** column select **Device Settings**.

In the **[System] > [Configuration] > [Device Settings]** page that displays, choose **Telnet**. The Telnet parameters are shown in the **[System] > [Configuration] > [Device Settings] > [Telnet Settings]** page that displays allows you to modify the settings; for information about the settings, see Telnet Settings in the "Device Settings" chapter of the *Caché Additional Configuration Settings Reference*. Click **Save** to save the new settings. When you have finished, return to the **[Home]** page.

B.2.3 Enable Telnet

When the portal **[Home]** page displays, choose **Security Management** from the System Administration column.

In the new page, choose **Services** from the Security Definitions column. This displays a table of the available Caché services and their respective status. For the table entry named, %Service_Telnet, if the Enabled column has the value "No", click on the %Service_Telnet name.

The **[System] > [Security Management] > [Services] > [Edit Service]** page that displays allows you to modify the properties of the telnet service. The meaningful entries are:

- Service enabled:

  Checking this box will enable the service so you can connect to Caché via telnet.

- Unauthenticated

  Checking this box means that anyone can connect via telnet as long as they know the IP address of the host on which this Caché instance is installed.

- Password

  Check this box if you wish to require operating system authentication (userid and password) of users attempting to connect to this Caché instance.

The remaining settings are for users of more advanced security features. When you have made you selections, click the **Save** button.

C

Frequently Asked Questions about MultiValue and Caché

C.1 General

Questions of a General Nature about Caché and MultiValue

Why is InterSystems adding MultiValue support to Caché?

For almost thirty years, InterSystems Corporation has been the partner of choice of professional application developers around the world. Two themes underlie our commitment to application developers: deliver great technology and have a total commitment to make our partners successful.

Developers in the MultiValue community have succeeded by delivering practical technology and consistent value over the long term. Pragmatism, a focus on rapid application development and a long-term view of business relationships characterize both the MultiValue world and the application partners already served by InterSystems.

InterSystems offers the MultiValue community

- a modern and exciting way to rapidly develop contemporary applications
- a blazingly fast, highly scalable and extremely reliable deployment engine
- a long-term partnership model

What does Caché offer MultiValue developers?

InterSystems implemented a broad set of MultiValue extensions to its Caché high-performance object database. These extensions bring the full range of Caché object and SQL development technologies to MultiValue developers:

- rapid application development through the latest technologies
- preservation and re-use of your existing MultiValue investments
- blending of the best of MultiValue, object, relational and web technologies
- a broad spectrum of deployment options

Is Caché MultiValue an add-on product to Caché? Is there a special version of Caché only for MultiValue users?

MultiValue features are a standard part of Caché. This release is not an add-on product or a special version of Caché for MultiValue users only.

What MultiValue features are included in Caché?

Caché has been enhanced with a wide range of MultiValue capabilities. These include:

- MultiValue Basic (MVBasic)
- MultiValue Storage
- MultiValue Dictionaries
- MultiValue Command Shell
- MultiValue Query Language (CMQL)
- MultiValue PROCs

What new features does Caché deliver for MultiValue developers?

At the heart of Caché is a powerful, multidimensional transaction engine that includes the ability to easily deploy distributed databases. This provides application developers with a unified data architecture uniting the power of objects, high performance SQL and traditional MultiValue data access.

For MultiValue applications, Caché offers:

- A complete native object model in MVBasic, including object inheritance, persistence and polymorphism
- Native, object-based XML and Web Services support
- MultiValue aware Studio integrated development environment
- Caché Server Pages (CSP) using MVBasic
- Blindingly fast SQL (concurrently with MultiValue queries)
- Bullet-proof database
- Complete security model (including auditing and encryption of data-at-rest)
- Simple, web-based system management

On what platforms does Caché run? Is MultiValue functionality supported on all Caché platforms?

MultiValue features are a standard part of Caché, and are available on all Caché platforms.

How fast is Caché?

High-performance database technology has been the cornerstone of InterSystems products since our founding 28 years ago. For object and SQL based applications, Caché is the engine of choice for both performance and scalability. Our continuing investments in the Caché engine will bring new levels of performance for MultiValue applications.

Will I have to abandon my MVBasic development environment?

No. Caché fully supports MultiValue Basic (MVBasic). You can continue to develop applications using MVBasic and MultiValue Queries. The new features are available to re-use and extend your application logic via Java, EJB, JDBC, ActiveX, .NET, C++, ODBC, XML, SOAP and more.

Once I migrate my MultiValue application to Caché, can I use all the other Caché features (objects, SQL, dynamic Web pages, Web Service, etc.) to access my MultiValue data?

Yes. Caché's Unified Data Architecture means that you can access your data as objects, SQL, as MultiValue files or using MultiValue queries.

Does Caché support a query language like English, Access or Recall?

Yes, we do and it is very similar to that on other MultiValue systems. English is the Microdata name for it; Access is the PICK name; Recall is the UniVerse name.

On Caché it is referred to as CMQL. It is described in the Caché MultiValue Query Language (CMQL) Reference.

How do I move my code and my data to Caché?

Caché provides a utility, MVIMPORT, to import files that have been created with account export utilities from MultiValue products.

How long does it take to migrate?

The time to migrate varies by application. InterSystems will work directly with you to facilitate your migration. Our Systems Engineers and World-Wide Response Center provide on-site, phone, web and email support.

C.2 Resources

What Resources Are Available to Users?

Is there training available?

InterSystems provides a broad range of education services, including on-site, classroom and web-based training. Free webinars are available covering every aspect of Caché development and deployment. New webinars specifically targeted at the MultiValue community will be introduced in the coming months.

How does InterSystems provide support?

InterSystems offers only one level of support — 24 hours a day, 365 days a year. Our InterSystems Worldwide Response Center (WRC) (WRC) is concentrated in Cambridge, Massachusetts (for proximity to the development team) but is complemented by local offices, providing local language and time zone support. Our service is second to none.

Is the documentation available separately?

Documentation on all aspects of Caché is available in Adobe Portable Document Format (PDF) from our corporate website. You may download individual books and articles as well as groups of related material.

This same material will be available once Caché is installed on your local machine, and can be accessed via the browser of your choice.

What other materials are available to help me learn Caché?

Once Caché is installed, there are interactive tutorials that teach you how to use some of the more important features of Caché. These are also accessed online. Choose the “Caché Tutorials” options from the documentation main page.

How do I get more information about Caché and MultiValue?

Peruse the information on our corporate website.

C.3 Business

What Are Some Business Considerations I Should Be Aware of?

What's the cost? Is it comparable to what I now pay?

InterSystems has a long history of working with application partners. Caché pricing is based on the model that InterSystems makes money when you make money i.e., when your application is deployed. We have a range of pricing models for Caché. Discuss your business model with us and we’ll work together to find the best model for your business.