Miva Merchant 9

Module API Reference Guide

version 4.5

MM9-103-4.5 6/3/19

Table of Contents

CHAPTER 1

Introduction

About Miva Merchant API

The Miva Merchant API comprises tools for third party developers to create extensions to the Miva Merchant shopping cart. It includes a robust collection of functions to leverage into modules that can be provided to Miva Merchant end users.

Miva Merchant is an inherently modular system from top to bottom. Most of the important functionality of an online store can be augmented or replaced by user-installable modules. These modules interact with the core Miva Merchant software using the Module API.

Much of the default functionality of Miva Merchant is itself packaged into modules, and available as part of the Limited Source Kit (LSK). (See *Requirements* on page 8 for instructions on obtaining the LSK.) The contents of the LSK provide concrete examples of the use and implementation of the functions described in this document.

Intended Audience

This guide is intended for application developers with web application development experience. A working knowledge of HTML and CSS is required. Familiarity with web scripting languages in general and Miva Script in particular is assumed. MySQL development skills and secure web scripting practices are highly recommended.

Purpose of This Guide

This guide, in conjunction with the other API reference guides listed in *Recommended Reading* on page 10, provides a comprehensive reference to the elements that make up the Miva Merchant API. It describes the purpose of each available function, its syntax, parameters and return values. It is a guide for developers to use as a resource for designing and building third party modules to plug into the Miva Merchant shopping cart software.

This guide is not a user's manual for the Miva Merchant software. Refer to the *Miva Merchant User Reference Guide* and *The Official Guide to Miva Merchant* for information on the Miva Merchant administration interface.

Requirements

To get started with the Miva Merchant API, you will need a functioning Miva Merchant installation (minimum version 5.00 PR5 or above) and a MivaSQL or MySQL database installation.

Recommendations regarding which database to use are beyond the scope of this guide. In general, MivaSQL is appropriate for smaller stores with limited customization needs. MySQL is more powerful, faster and easier to customize. MivaSQL is easier to maintain, however it is not PCI compliant and is therefore discouraged for high-volume stores.

Building Miva Merchant modules requires the Miva Merchant Empresa script interpreter and the Miva Merchant Script Compiler. You may optionally install and run Miva Merchant Mia instead of (or in addition to) Empresa in a Windows localhost environment. This allows you to develop your Miva Merchant modules offline without a live web server.

Empresa, Mia, the Script Compiler and the Limited Source Kit can be downloaded from the Miva Merchant website free of charge here:

http://www.miva.com/support/downloads

Development Accounts

Development Accounts are made available to active developers for the specific purpose of developing modules and add-ons to Miva Merchant. To create a developer account:

1. Go to https://docs.miva.com/developers
2. Click on **Free Developer Store**
3. Enter your login and contact information
4. Click **Create Store**

Document Conventions

The following typographic conventions are used in this guide:

Bold – Used to emphasize new terms or product features.

Bold sans serif – Used for syntax descriptions and filenames.

Bold italic sans serif – Used for user-defined input.

Italic – Used for external document references.

Blue san serif underline – Used for clickable URL links.

Blue italic – Used for clickable cross-reference links to other sections within the document.

`Monospace` – Used for code examples and screen output.

`*Italic monospace*` – Used for user-defined values in code examples.

Terminology

Admin – The administration interface, **admin.mvc**, and the files it calls.

API – A collection of functions written in Miva Script that interface with the Miva Merchant software.

Feature – A set of functionality that a module provides – for example, batch reports, payment or shipping.

Miva Merchant – The storefront application, **merchant.mvc**, and all the files it calls (such as those governing the user interface and various features).

Miva Merchant Empresa – The server-side engine for Miva Script. A script interpreter that runs on Windows and *nix web servers.

Miva Merchant Limited Source Kit (LSK) – Source code comprised of a collection of script files containing functions that can be implemented by third party developers.

Miva Merchant Mia – A portable, standalone version of the Empresa engine that runs on a Windows localhost server. No other server software is required unless POP and SMTP functions are required. Provides a development environment for software developers to run and test their work on a Windows desktop.

Miva Merchant Script Compiler – Compiler for Miva Script that creates executable **.mvc** files to be interpreted by Empresa or Mia.

Miva Script – Proprietary server side scripting language used for API implementation.

Module – An implementation of one or more module API features.

Store Morph Technology (SMT) – Miva Merchant Page Templates, Items, Entities and the Miva Merchant Template Language collectively make up the **Store Morph Technology**.

User Interface (UI) – The portion of the application that displays the screens the shopper sees and with which he or she interacts.

XML Provisioning – A format for importing/exporting data in Miva Merchant that allows advanced store setup through an XML file.

Recommended Reading

The following documents provide useful information for developers. All can be found on the Miva Merchant **Developer Docs** page.

http://docs.miva.com/developers

- *Miva Merchant API Reference Guide*s – A collection of API reference guides (including this one). See the Documentation web page for the latest versions.
- *Miva Merchant Database Reference Manual* – A comprehensive reference to all the tables in the Miva Merchant API, including table relationship diagrams.
- *MivaScript Function Reference* (a list of MivaScript functions and usage)
- *MivaScript online documentation* (www.mivascript.com)
- *Miva API Function Reference* (a list of API functions and usage)
- *Miva Template Language* (online documentation)
- *Miva Code Samples* (online documentation)
- *XML Provisioning Tags* (online documentation)
- *MMBatchList Guide*
- *MMButton Guide*
- *MMMenuButton Guide*

Additionally, there are two end user documents available.

- *Miva Merchant User Reference Guide* (www.miva.com/referenceguide)
- *The Official Guide to Miva Merchant 9* by Pamela Hazelton can be downloaded for free at the following link:

  http://apps.miva.com/product/MEDIASI-MM9EBOOK.html

Training Videos

There are a number of training videos available on the Miva website.

Template Language videos:

http://docs.miva.com/dev-videos/template-language

Developer Training Series:

http://docs.miva.com/dts/developer-training-series

Module Developer Series:

http://docs.miva.com/dev-videos/module-developer-series

Technical Support

Access to technical support, patches and downloads, the community forums, Miva Merchant documentation, video tutorials and other resources can be found at the following link:

http://www.miva.com/support

The Miva Merchant Knowledgebase can be accessed at the following link:

https://support.miva.com/supportsuite/index.php?/default/Knowledgebase/List

The Miva Merchant Community Forums can be accessed at the following link:

http://www.miva.com/forums/index.php

CHAPTER 2

API Overview

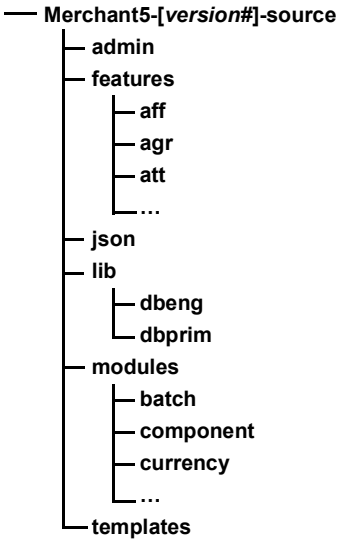
Miva Merchant provides a library of functions for third party developers to implement to produce plug-in modules for the Miva Merchant software. These functions allow developers to customize the Administrative User Interface, the Wizard User Interface, the Database API, the Module API and Miva Merchant's built-in Features (e.g. the Inventory System, the Template System, the Shopping Interface, etc.)

The functions are contained in the Miva Merchant Limited Source Kit (LSK). The following chapters describe these functions, grouped by the primary feature type. For reference, all available functions are listed alphabetically in *Appendix B: API Functions* on page 327 along with their module and location.

Miva Merchant Limited Source Kit

The Miva Merchant LSK contains the source code for a library of functions that developers can implement to create third party modules to extend and customize Miva Merchant's capabilities. The LSK also includes core modules that make up the Miva Merchant UI. Developers may refer to these files for examples of implementations of the API.

The LSK is organized with the following directory structure:



- The **admin** directory contains functions for Admin and UI.
- The **features** directory holds the bulk of the functions to be leveraged by third party developers.
- The **json** directory contains data structures and arrays.
- The **lib** directory contains functions relating to global variables, encryption/decryption and the database API.
- The **modules** directory contains the core components for the Miva Merchant storefront.
- The **templates** directory holds templates for various Miva Merchant UIs.

Implementing a Module API Feature

Miva Merchant modules are typically written in a combination of Miva Script and HTML and have a specific structure — a comment block at the top followed by the function **Module_Description**, which assigns the local module variables. The rest of the module contains the functions that make up the module. The file itself has a "**.mv**" extension.

The ".mv" File

The **Module_Description** function is required in every module. It assigns the code and name for the module. It also declares which module features are to be implemented.

Important: All functions for the features you implement must be included in the module. Otherwise, runtime errors will occur.

The functions contained in a feature are described in the *API Reference Guide*s. They can also be found in the API document on the Miva Merchant website here:

http://www.miva.com/docs/api

The following code example shows a complete module, **mmlsk-ex-system.mv**, which can be found in the LSK under **modules/system**. Note that all of the functions included in the features **system** and **vis_system** are included in the module.

```
<MvCOMMENT>
|
| Miva Merchant v5.x
|
| This file and the source codes contained herein are the property of
| Miva Merchant, Inc.  Use of this file is restricted to the specific terms and
| conditions in the License Agreement associated with this file.  Distribution
| of this file or portions of this file for uses not covered by the License
| Agreement is not allowed without a written agreement signed by an officer of
| Miva Merchant, Inc.
|
| Copyright 1998-2011 Miva Merchant, Inc.  All rights reserved.
| http://www.mivamerchant.com
|
| Prefix         : MER-SYS-EXP-
| Next Error Code: 1
|
</MvCOMMENT>
```

```
<MvFUNCTION NAME = "Module_Description" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvASSIGN NAME = "l.module:code"      VALUE = "mmlsk-ex-system">
    <MvASSIGN NAME = "l.module:name"      VALUE = "System Extensions Example">
    <MvASSIGN NAME = "l.module:provider"  VALUE = "Miva Merchant">
    <MvASSIGN NAME = "l.module:version"   VALUE = "5.5000">
    <MvASSIGN NAME = "l.module:api_ver"   VALUE = "5.00">
    <MvASSIGN NAME = "l.module:features"  VALUE = "system, vis_system">
</MvFUNCTION>


<MvFUNCTION NAME = "Module_System_Tabs" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvFUNCTIONRETURN VALUE = "EXMP:System Extension Example">
</MvFUNCTION>


<MvFUNCTION NAME = "Module_System_Content" PARAMETERS = "module var, tab,
 load_fields" STANDARDOUTPUTLEVEL = "text, html, compresswhitespace">
    <MvIF EXPR = "{ l.tab EQ 'EXMP' }">
              <MvEVAL EXPR = "{ 'Hello from System Extension Content Settings.'
              }">
    </MvIF>
    <MvFUNCTIONRETURN VALUE = 1>
</MvFUNCTION>


<MvFUNCTION NAME = "Module_System_Validate" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvEVAL EXPR = "{ 'Hello from System Extension Validate Settings.<br>' }">
    <MvFUNCTIONRETURN VALUE = 1>
</MvFUNCTION>


<MvFUNCTION NAME = "Module_System_Update" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvEVAL EXPR = "{ 'Hello from System Extension Update Settings.<br>' }">
    <MvFUNCTIONRETURN VALUE = 1>
</MvFUNCTION>


<MvFUNCTION NAME = "SystemModule_Screen" PARAMETERS = "module var, screen"
 STANDARDOUTPUTLEVEL = "">
    <MvEVAL EXPR = "{ 'Hello from System Extension Runtime Screen.<br>' }">
    <MvFUNCTIONRETURN VALUE = "1">
</MvFUNCTION>


<MvFUNCTION NAME = "SystemModule_Action" PARAMETERS = "module var, action"
 STANDARDOUTPUTLEVEL = "">
    <MvEVAL EXPR = "{ 'Hello from System Extension Runtime Action.<br>' }">
    <MvFUNCTIONRETURN VALUE = "1">
```

```
</MvFUNCTION>


<MvFUNCTION NAME = "SystemModule_UIException" PARAMETERS = "module var, exception"
 STANDARDOUTPUTLEVEL = "">
    <MvEVAL EXPR = "{ 'Hello from System Extension Runtime UI Exception.<br>' }">
    <MvFUNCTIONRETURN VALUE = "1">
</MvFUNCTION>
```

Creating a New Function

When you write a new function, you must declare it via the Miva Script **<MvFUNCTION>** tag:

```
<MvFUNCTION NAME = "Module_Install_Store"
 PARAMETERS = "module var">
    …
    code
    …
</MvFUNCTION>
```

Calling a Function From the API

There are two methods for calling functions. The **<MvDO>** tag allows you to call a function in an external file declared with **<MvFUNCTION>**.

Example

```
<MvDO FILE = "{ g.Module_Library_DB }"
 NAME = "l.return_code"
 VALUE = "{ Product_Insert( l.product ) }">
```

Alternatively, you can call an external function via the **<MvASSIGN>** tag.

Example

```
<MvASSIGN NAME = "l.return_code"
 VALUE = "{ [ g.Module_Library_DB ]
 .Product_Insert( l.product ) }">
```

In the preceding examples, the **<MvDO>** and **<MvASSIGN>** tags achieve exactly the same goal — they call the **Product_Insert** function from **lib/dbeng/products.mv** and store the return value in a local variable called **return_code**. The global variable **g.Module_Library_DB** contains the path to **lib/dbeng/products.mv**.

Global Variables

The **lib/config.mv** file configures global variables for use by the Miva Merchant API. These variables provide access to the various **features/** files. Scripts that call into the API must execute **lib/config.mv** (via **<MvDO>** or **<MvASSIGN>**) prior to calling API functions.

Important: Variables are initialized in **config.mv** with relative paths, for example, **g.Library_DB**. When developers use these variables in their modules, they should reference them by the absolute path, e.g., **g.Module_Library_DB**, as in our example above.

Uploading a Module to Miva Merchant

Once you have finished editing your module, save it to the code name specified by the **l.module.code** variable and compile it with the Miva Merchant Script Compiler via the following command:

mvc *my_module*.mv

If there are no errors, a "***my_module*.mvc**" file will be created. Upload it to your store via **Add Modules** in the Miva Merchant admin. The "**.mvc**" file will be placed in the appropriate directory (e.g., **www/mm5/5.00/modules/system**) and the module will be activated.

Building a New Module — the Easy Way

The easiest way to create a new module is to take an existing module, strip out the unnecessary content, then add new functionality. Choose a module from the LSK that employs the same core feature(s) as the module you are building.

In the following example, we will create a new currency format that displays in your store in red, blinking text with the currency symbol, "N$".

All modules require the **Module_Description** function at the top. The following code excerpt is from the LSK module **mmlsk-usmoney.mv** under **modules/currency**:

```
<MvFUNCTION NAME = "Module_Description" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvASSIGN NAME = "l.module:code"     VALUE = "mmlsk-usmoney">
    <MvASSIGN NAME = "l.module:name"     VALUE = "US Currency Formatting">
    <MvASSIGN NAME = "l.module:provider" VALUE = "Miva Merchant">
    <MvASSIGN NAME = "l.module:version"  VALUE = "5.8000">
    <MvASSIGN NAME = "l.module:api_ver"  VALUE = "5.00">
    <MvASSIGN NAME = "l.module:features" VALUE = "currency">
</MvFUNCTION>
```

We will create a new module with the name, "New Currency Formatting" (code name: "newmoney"). It will require the feature **currency**.

Starting from the module **mmlsk-usmoney.mv**, keep **Module_Description** but assign new values to **l.module:code** and **l.module:name**. Assign the required features to **l.module:features**.

```
<MvFUNCTION NAME = "Module_Description" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvASSIGN NAME = "l.module:code"     VALUE = "newmoney">
    <MvASSIGN NAME = "l.module:name"     VALUE = "New Currency Formatting">
    <MvASSIGN NAME = "l.module:provider" VALUE = "Miva Merchant">
    <MvASSIGN NAME = "l.module:version"  VALUE = "5.8000">
    <MvASSIGN NAME = "l.module:api_ver"  VALUE = "5.00">
    <MvASSIGN NAME = "l.module:features" VALUE = "currency">
</MvFUNCTION>
```

The **currency** feature contains three functions: **CurrencyModule_AddFormatPlainText**, **CurrencyModule_AddFormatPlainTextShort** and **CurrencyModule_AddFormatting**. All three functions must be in your module.

If your module requires other features, they can be included in a comma-separated list. For example:

```
    <MvASSIGN NAME = "l.module:features" VALUE = "currency, util, vis_util">
```

In this case, your module would require the three **currency** functions listed above plus the eight functions that comprise the **util** and **vis_util** features.

We can re-purpose the **CurrencyModule_AddFormatting** function from **mmlsk-usmoney.mv** to add red, blinking text to the currency in our store by stripping out the code in **mmlsk-usmoney.mv** and substituting the appropriate HTML code. The dollar sign ($) is used as a concatenation operator.

```
<MvFUNCTION NAME = "CurrencyModule_AddFormatting" PARAMETERS = "module var, value"
 STANDARDOUTPUTLEVEL = "">
    <MvASSIGN NAME = "l.retval"VALUE = "{ '<font color=\"red\"><blink>N$ ' $
     l.value $ '</blink></font>' }">
    <MvFUNCTIONRETURN VALUE = "{ l.retval }">
</MvFUNCTION>
```

Note: Attribute values are surrounded by double quotes ("") in Miva Script. To specify an HTML attribute with double quotes (e.g., **<font color="red">**) inside an expression, precede the quote mark (") with a backslash (\) as shown above.

The other two **currency** functions, **CurrencyModule_AddFormatPlainText** and **CurrencyModule_AddFormatPlainTextShort**, can be duplicated as is from **mmlsk-usmoney.mv**.

```
<MvFUNCTION NAME = "CurrencyModule_AddFormatPlainText" PARAMETERS = "module var,
 value" STANDARDOUTPUTLEVEL = "">
    <MvFUNCTIONRETURN VALUE = "{ CurrencyModule_AddFormatting( l.module, l.value )
     }">
</MvFUNCTION>
<MvFUNCTION NAME = "CurrencyModule_AddFormatPlainTextShort" PARAMETERS = "module
 var, value" STANDARDOUTPUTLEVEL = "">
    <MvFUNCTIONRETURN VALUE = "{ CurrencyModule_AddFormatting( l.module, l.value )
     }">
</MvFUNCTION>
```

Save the file as **newmoney.mv** and compile it. Then upload the resultant **newmoney.mvc** file to your store via the Miva Merchant admin, as described in *Uploading a Module to Miva Merchant* on page 18.

To implement the new currency format, go to **Edit Store: Settings** in the admin. Select "New Currency Formatting" from the **Currency Formatting** drop-down list. When you view the store in your browser after applying this change, you will see the red blinking text with a currency symbol "N$" everywhere currency is visible on the store pages.

CHAPTER 3

Module API

The **modules** directory contains examples of the core Miva Merchant software. It is included in the LSK for third party developers to use as a guide for creating new modules. The Module API comprises a library of functions for this purpose. This chapter describes the Module API functions.

Note: Functions that are new to Module API version 5.60 and above and functions that have been deprecated include the supported API version.

Refer to *Implementing a Module API Feature* on page 15 for examples of function calls into the Module API.

All Modules

All modules require the following function.

Module_Description

Miva Merchant, the Admin and the UI all call the **Module_Description** function at several different points. It provides information that identifies the module and describes some of its characteristics, such as a list of features it implements. One uses this function to load this information into a structure with the name "module". Use the "var" option on the parameter to pass variables to the calling program by reference. The actual value returned by **MvFuncReturn** does not contain the main information.

Syntax

Module_Description (module var)

Parameters

module The **Module** structure used to pass variables to the calling program

Return Value

NULL

Example

```
<MvFUNCTION NAME = "Module_Description" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvASSIGN NAME = "l.module:code"       VALUE = "mmlsk-stdacct">
    <MvASSIGN NAME = "l.module:name"       VALUE = "Standard Batch Report">
    <MvASSIGN NAME = "l.module:provider"   VALUE = "Miva Merchant">
    <MvASSIGN NAME = "l.module:version"    VALUE = "5.8000">
    <MvASSIGN NAME = "l.module:api_ver"    VALUE = "5.00">
    <MvASSIGN NAME = "l.module:features"   VALUE = "batchreport">
</MvFUNCTION>
```

In the preceding example, a module calls the **Module_Description** function and passes variables (**"l.module:code"**, **"l.module.name"**, etc.) into the **Module** structure.

Batch Report Feature (batchreport)

Miva Merchant supports the creation of batches of orders. Batches can be run daily, weekly, or at will. The **Batch Report** feature (**batchreport**) creates a summary report for the specified batch order. The feature has complete control over what information is used and how it is displayed.

The **batchreport** feature contains the following functions:

- **BatchReportModule_Order_Reports**
- **BatchReportModule_Report**
- **BatchReportModule_Run_OrderList**
- **BatchReportModule_Run_ShipmentList**
- **BatchReportModule_Shipment_Reports**

BatchReportModule_Order_Reports

This function returns a list of order reports to Miva Merchant generated by this module. The module puts information about each supported report into the **reports** parameter.

Supported API Version

5.70 and higher

Syntax

BatchReportModule_Order_Reports(module var, reports var)

Parameters

module	The **Module** record of the current module
reports	An array with each element in the array having the following members: **code** – A string that uniquely identifies the report to the module (it can be anything) **name** – A descriptive name that will be displayed to the user

Return Value

The number of reports that the module supports (how many entries it put into the **reports** array parameter)

BatchReportModule_Report

The **BatchReportModule_Report** generates an order batch report. The module determines the format and content of the output. Typically, a module would load a list of Order records using **OrderList_Load_Batch()**. Examples of implementations of **BatchReportModule_Report** can be found in the **modules/batch** directory.

Syntax

BatchReportModule_Report(module var, batch_id)

Parameters

module	The **Module** record of the current module
batch_id	The unique ID of the batch to be output

Return Value

1 on success

0 on error

BatchReportModule_Run_OrderList

This function is called to generate and output a report on a particular list of orders. The orders can be loaded from an order batch or directly selected by the user. The module generates whatever content is appropriate for the report using the specified orders.

Supported API Version

5.70 and higher

Syntax

BatchReportModule_Run_OrderList(module var, report_code, orders var, order_count)

Parameters

module	The **Module** record of the current module.
report_code	The report code to generate. This parameter comes from the **code** member of the **reports** array returned by the module in **BatchReportModule_Order_Reports**.
orders	An array of **Order** records, one for each order to be included in the report.
order_count	The number of **Order** records in the **orders** array.

Return Value

1 on success

0 on error

BatchReportModule_Run_ShipmentList

This function is called to generate and output a report on a particular list of shipments. The shipments can be loaded from a shipment batch or directly selected by the user. The module generates whatever content is appropriate for the report using the specified shipments.

Supported API Version

5.70 and higher

Syntax

BatchReportModule_Run_ShipmentList(module var, report_code, shipments var, shipment_count)

Parameters

module	The **Module** record of the current module.
report_code	The report code to generate. This parameter comes from the **code** member of the **reports** array returned by the module in **BatchReportModule_Shipment_Reports**.
shipments	An array of shipment records, one for each shipment to be included in the report.
shipment_count	The number of shipment records in the **shipments** array.

Return Value

1 on success

0 on error

BatchReportModule_Shipment_Reports

Returns a list of shipment reports to Miva Merchant generated by this module. The module puts information about each supported report into the **reports** parameter.

Supported API Version

5.70 and higher

Syntax

BatchReportModule_Shipment_Reports(module var, reports var)

Parameters

module	The **Module** record of the current module
reports	An array with each element in the array having the following members: **code** – A string that uniquely identifies the report to the module (it can be anything) **name** – A descriptive name that will be displayed to the user

Return Value

The number of reports that the module supports (how many entries it put into the **reports** array parameter)

Box Packing Feature (boxpacking)

Most modern shipping calculation APIs require the dimensions of the package(s) being shipped to be known ahead of time. The boxpacking feature allows modules to be built to implement store-specific rules for determining how many and what size boxes will be used to ship a **Basket**, **OrderShipment**, or other collection of shippable items.

The underlying shipping system maintains a list of available boxes with their dimensions (width, length, height), and the **boxpacking** module API provides a mechanism for the box packing module to capture additional configuration information that is required for whatever algorithm the module implements. For example, the pack by quantity module provides an additional box-level field that indicates the maximum quantity of items that can be put in each box.The **boxpacking** feature contains the following functions:

- **BoxPackingModule_Box_Delete**
- **BoxPackingModule_Box_Field**
- **BoxPackingModule_Box_Fields**
- **BoxPackingModule_Box_Insert**
- **BoxPackingModule_Box_Invalid**
- **BoxPackingModule_Box_Prompt**
- **BoxPackingModule_Box_Provision**
- **BoxPackingModule_Box_Update**
- **BoxPackingModule_Box_Validate**
- **BoxPackingModule_Pack_Items**

BoxPackingModule_Box_Delete

This function is called when a box is deleted to allow the module to clean up any data that references the box.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Delete(module var, box_id)

Parameters

module	The **Module** record of the current module
box_id	The unique ID of the box that was deleted

Return Value

1 on success

0 on error

BoxPackingModule_Box_Field

This function draws HTML input element(s) required for configuration of a single field from the list returned by **BoxPackingModule_Box_Fields**. The module should output the required HTML directly.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Field(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **BoxPackingModule_Box_Fields**.

Return Value

Ignored

BoxPackingModule_Box_Fields

This function returns a comma separated list of the field identifiers for module-specific fields that should be present for a given box.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Fields(module var, box var)

Parameters

module	The **Module** record of the current module
box	The **Box** record being edited or NULL if a box is being added

Return Value

A comma separated list of field identifiers

BoxPackingModule_Box_Insert

This function is called when a new box is created so that the module can store any module controlled box fields.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Insert(module var, box var)

Parameters

module	The **Module** record of the current module
box	A **Box** record containing the newly created box

Return Value

1 on success

0 on error

BoxPackingModule_Box_Invalid

When **BoxPackingModule_Box_Validate** returns **0**, this function is called for each field to determine which field(s) should be displayed in the invalid state.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Invalid(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The identifier of the field being queried

Return Value

1 if the specified field is invalid

0 if the specified field is valid

BoxPackingModule_Box_Prompt

This function is called for each field returned by **BoxPackingModule_Box_Fields** to obtain the prompt (descriptive text displayed to the left of the field).

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Prompt(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The identifier of the field being queried

Return Value

Textual prompt. HTML is permitted.

BoxPackingModule_Box_Provision

This function is called when a box is created using the **Box_Add** provisioning tag and the **<BoxPackingSettings>** tag is present. The module is expected to implement provisioning for whatever settings are normally maintained on a box-by-box basis.

Important: All **boxpacking** modules must implement this function.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Provision(module var, box var, provide_xml var)

Parameters

module	The **Module** record of the current module
box	The newly created **Box** record
provide_xml	Provisioning-parsed XML from the **BoxPackingSettings** tag

Return Value

None. This function must not return a value. Provisioning errors should be logged using **PRV_LogMessage** or **PRV_LogError**.

BoxPackingModule_Box_Update

This function is called when a box is modified so that the module can store any changes to the module controlled box fields.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Update(module var, box var)

Parameters

module	The **Module** record of the current module
box	The **Box** record being updated

Return Value

1 on success

0 on error

BoxPackingModule_Box_Validate

This function is called to validate module-provided box fields when creating or updating a box record from the administrative interface.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Box_Validate(module var, box var)

Parameters

module	The **Module** record of the current module
box	The **Box** record being updated, or NULL if a new box is being created

Return Value

1 if all module-provided fields are valid

2 if any module-provided field is invalid

Note: Validation failures can be reported via **BoxPackingModule_Box_Invalid** or by calling the **FieldError** function.

BoxPackingModule_Pack_Items

This function is called to build a packaging solution using currently enabled boxes for a specified set of items.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

BoxPackingModule_Pack_Items(module var, items var, item_count, packages var)

Parameters

module The **Module** record of the current module

items An input array of items to pack. Each element in the array has the following members:

- **product** – A Product record for the item (if one exists)
- **basketitem** – The BasketItem of the item (if one exists)
- – OR –
- **orderitem** – The OrderItem of the item (if one exists)
- **width** – The width of the item in store dimension units
- **length** – The length of the item in store dimension units
- **height** – The height of the item in store dimension units
- **weight** – The weight of the item in store dimension units

item_count The number of elements in the **items** array

packages An output array that receives the resulting packaging solution. Each element in the array should receive the following members:

- **width** – The width of the package in store dimension units
- **length** – The length of the package in store dimension units
- **height** – The height of the package in store dimension units
- **items[]** – An array containing the items from the input **items** array to be packed in this package
- **item_count** – The number of items in the output **items[]** array
- **weight** – The total weight of the package

Note: Multiple quantities of the same item are expanded and will be represented by multiple items in the **items** array.

Return Value

The number of entries the module put into the **packages** array

Clean Up Store Feature (cleanup_store)

This feature gives modules the ability to clean up residual data that gets left behind when store contents are deleted.

The **cleanup_store** feature contains the following function:

- **Module_Cleanup_Store**

Module_Cleanup_Store

The **Module_Cleanup_Store** function is called through the admin interface on the **Delete Baskets** page when a module contains the feature **cleanup_store**. This function can do any clean up functionality your module requires from one unified location in the Miva Merchant admin. It is also available through provisioning.

Example

```
<Provision>
    <Store code="xxx">
        <Module_Cleanup [module_code="xxx"] />
    </Store>
</Provision>
```

Supported API Version

All versions (new in PR8 Update 6)

Syntax

Module_Cleanup_Store(module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Client Side Feature (clientside)

This feature gives modules the ability to deliver static CSS or JavaScript content through **clientside.mvc**. Once a module implements this feature, URLs to **clientside** should take the following form:

http://www.somesite.com/mm5/clientside.mvc?
Module_Code=<module_that_implements_clientside>&Filename=xxx

The **Filename** parameter is then available to the module as **g.Filename**.

The **clientside** feature contains the following function:

- **Module_Clientside**

Module_Clientside

This function is called when a **clientside.mvc** URL contains the **Module_Code** parameter. The filename requested is available in **g.Filename**. The module should call the function **Module_Content_Type** in **g.Module_Clientside** to set the content type, then directly output the appropriate static content.

Syntax

Module_Clientside(module var)

Parameters

module The **Module** record of the current module

Return Value

None. This function must not return a value.

Component Feature (component)

Component modules underlie the items used in Store Morph Technology providing information to the UI on how to display data (or telling Miva Merchant how to alter the data before displaying it).

You install a component to a store by assigning it to an item within the store. This can be achieved on the **Edit Item** configuration page. Alternatively, you can assign a component to the store by assigning it as an extension of a component already installed to the store. This can be done via the **Items** tab on the **Edit Page** configuration screen.

The **component** feature contains the following functions:

- **ComponentModule_Content**
- **ComponentModule_Defaults**
- **ComponentModule_Initialize**
- **ComponentModule_Page_Assign**
- **ComponentModule_Page_Unassign**
- **ComponentModule_Prerender**
- **ComponentModule_Render_End**
- **ComponentModule_Render_Head**
- **ComponentModule_Render_Start**
- **ComponentModule_Tabs**
- **ComponentModule_Update**
- **ComponentModule_Validate**

ComponentModule_Content

ComponentModule_Content runs when the Admin displays the **Edit Page** configuration screen for a page using an **Item** based on the component. It tells the Admin what to display on the **Edit Page** screen when the user selects the tab belonging to this module. Examples of implementations of **ComponentModule_Content** can be found in the **modules/component** directory.

Syntax

ComponentModule_Content (module var, item, tab, load_fields, field_prefix, fields var, settings var)

Parameters

module	The **Module** record of the current module
item	The **item** record of the current item
tab	The code of the currently displayed tab
load_fields	A Boolean value indicating whether the current screen's data fields must be populated from a data source
field_prefix	A component-specific structure name (as a string) in which all variables on a screen are stored
fields	A structure containing all variables that were submitted from the component's content screen
settings	A structure containing configuration options for the current component on the current page

Return Value

1 on success

0 on error

ComponentModule_Defaults

Admin calls this function when you assign a corresponding item to a page. Examples of implementations of **ComponentModule_Defaults** can be found in the **modules/component** directory.

Syntax

ComponentModule_Defaults(module var, settings var)

Parameters

module	The **Module** record of the current module
settings	A structure containing configuration options for the current component on the current page

Return Value

Ignored

ComponentModule_Initialize

Miva Merchant calls this function when displaying a page using an item based on the component. Use this function to add, delete, or edit values in the **settings** and **all_settings** structures that contain the data for display on the page (e.g., product name, list of shipping methods). Miva Merchant will call this function sequentially from any component extending the first component, for example, to add to the product name, or alter the order of the shipping methods in the list. Examples of implementations of **ComponentModule_Initialize** can be found in the **modules/component** directory.

Syntax

ComponentModule_Initialize(module var, item, all_settings var, settings var)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
all_settings	A structure containing all configuration options for all assigned components for the current page
settings	A structure containing configuration options for the current component for the current page

Return Value

1 on success

0 on error

ComponentModule_Page_Assign

Admin calls this function when you assign a corresponding item to a page. Examples of implementations of **ComponentModule_Page_Assign** can be found in the **modules/component** directory.

Syntax

ComponentModule_Page_Assign(module var, page var, item, settings var)

Parameters

module	The **Module** record of the current module
page	The **page** record for the current page
item	The **item** record for the current item
settings	A structure containing configuration options for the current component for the current page

Return Value

1 on success

0 on error

ComponentModule_Page_Unassign

Admin calls this function when you unassign an item from a page. Examples of implementations of **ComponentModule_Page_Unassign** can be found in the **modules/component** directory.

Syntax

ComponentModule_Page_Unassign(module var, page var, item, settings var)

Parameters

module	The **Module** record of the current module
page	The **page** record for the current page
item	The **item** record for the current item
settings	A structure containing configuration options for the current component for the current page

Return Value

1 on success

0 on error

ComponentModule_Render_Head

This function is called to allow modules to output content in the HTML **<head>** tag.

Supported API Version

9.07 and higher

Syntax

ComponentModule_Render_Head(module var, item, all_settings var, settings var)

Parameters

module	The **Module** record of the current module
item	The code of the **item** being rendered
all_settings	A structure containing all configuration options for all assigned components for the current page
settings	A structure containing configuration options for the current component for the current page

Return Value

Ignored

ComponentModule_Prerender

This function is called before **ComponentModule_Render_Start** to allow modules to separate their in-render loading activities from the actual output of content. This provides a mechanism similar to **ComponentModule_Initialize**, but with the ability to see the **param** value which will be passed to **ComponentModule_Render**.

ComponentModule_Prerender is called when an item is explicitly pre-rendered through the Template Manager's **TemplateManager_Component_Prerender** function or immediately before calling **ComponentModule_Render_Start**.

Supported API Version

5.72 and higher (new in PR8 Update 5)

Syntax

ComponentModule_Prerender(module var, item, all_settings var, settings var, param)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
all_settings	A structure containing all configuration options for all assigned components for the current page
settings	A structure containing configuration options for the current component for the current page
param	The string value that is passed through the **param** attribute in the **mvt:item** tag

Return Value

Ignored

ComponentModule_Render_End

Admin calls this function when the Template Manager encounters an item end tag (**<mvt:item />**) in the template. The function tells Miva Merchant what HTML source to place at that point on the page. Examples of implementations of **ComponentModule_Render_End** can be found in the **modules/component** directory.

Syntax

ComponentModule_Render_End(module var, item, all_settings var, settings var, param)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
all_settings	A structure containing all configuration options for all assigned components for the current page
settings	A structure containing configuration options for the current component for the current page
param	The string value that is passed through the **param** attribute in the **mvt:item** tag

Return Value

Ignored

ComponentModule_Render_Start

Admin calls this function when the Template Manager encounters an item start tag (**<mvt:item>**) in the template. The function tells Miva Merchant what HTML source to place at that point on the page. Examples of implementations of **ComponentModule_Render_Start** can be found in the **modules/component** directory.

Syntax

ComponentModule_Render_Start(module var, item, all_settings var, settings var, param)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
all_settings	A structure containing all configuration options for all assigned components for the current page
settings	A structure containing configuration options for the current component for the current page
param	The string value that is passed through the **param** attribute in the **mvt:item** tag

Return Value

Ignored

ComponentModule_Tabs

Miva Merchant calls this function when Admin displays an **Edit Page** configuration screen for a page using an item based on the component. It returns a string of the tabs that the component will generate. The string takes the following form:

```
<code>:<title>,<code2>,<title2>
```

The module may specify as many `<code>:<title>` pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Syntax

ComponentModule_Tabs (module var, item, settings var)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
settings	A structure containing configuration options for the current component for the current page

Return Value

A string describing the tabs to be added (see description above)

ComponentModule_Update

Miva Merchant calls this function when **Update** is selected on the Edit Page configuration screen for a page using an item based on the component. Examples of implementations of **ComponentModule_Update** can be found in the **modules/component** directory.

Syntax

ComponentModule_Update (module var, item, field_prefix, fields var, settings var)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
field_prefix	A component-specific structure name (as a string) in which all variables on a screen are stored
fields	A structure containing all variables that were submitted from the component's content screen
settings	A structure containing configuration options for the current component on the current page

Return Value

1 on success

0 on error

ComponentModule_Validate

Miva Merchant calls this function when **Validate** is selected on the **Edit Page** configuration screen for a page using an item based on the component. Examples of implementations of **ComponentModule_Validate** can be found in the **modules/component** directory.

Syntax

ComponentModule_Validate (module var, item, field_prefix, fields var)

Parameters

module	The **Module** record of the current module
item	The **item** record for the current item
field_prefix	A component-specific structure name (as a string) in which all variables on a screen are stored
fields	A structure containing all variables that were submitted from the component's content screen

Return Value

1 on success

0 on error

Component Module Provisioning Feature (component_prov)

Modules that implement the **Component Module Provisioning** feature (**component_prov**) support provisioning of their settings through the provisioning system.

Note: Modules must include the **provision_store** feature in order to make use of the **component_prov** feature.

The **component_prov** feature contains the following function:

- **ComponentModule_Provision**

ComponentModule_Provision

Admin calls this function when it processes the **provide.xml** file if it encounters a tag identifying this module: **<Module code=[*code for this module*]>**. Admin then passes the information contained by the tag to the function. The function can then perform its task — for example, writing data to the appropriate database table. In addition, it receives the item settings. Examples of implementations of **ComponentModule_Provision** can be found in the **modules/component** directory.

Syntax

ComponentModule_Provision (module var, provide_xml var, settings var)

Parameters

module	The **Module** record of the current module
provide_xml	A structure containing the contents of the XML provisioning file
settings	A structure containing configuration options for the current component on the current page

Return Value

1 on success

0 on error

Currency Formatting Feature (currency)

Each function belonging to the **Currency Formatting** feature (**currency**) adds formatting to a decimal number that is passed to it. It provides a place to specify, among other things, the currency symbol and number of decimal places for use in displaying a price value.

In addition to simple formatting, a **currency** module can convert the values of various currencies. It can display US currency, Canadian currency or Eurodollars, or use the daily rate of exchange to provide updated values from day to day. It can even do a lookup every hour to update the currency values.

The **currency** feature contains the following functions:

- **CurrencyModule_AddFormatting**
- **CurrencyModule_AddFormatPlainText**
- **CurrencyModule_AddFormatPlainTextShort**
- **CurrencyModule_Output_CurrencyFormat_JavaScript**

CurrencyModule_AddFormatting

Miva Merchant calls this function wherever it displays a price on one of the store pages. Miva Merchant will call this function only from the **currency** module selected by the store administrator using the **Edit Store** configuration screen. Examples of implementations of **CurrencyModule_AddFormatting** can be found in the **modules/currency** directory.

Syntax

CurrencyModule_AddFormatting (module var, value)

Parameters

module	The **Module** record of the current module
value	The currency amount represented as a number

Return Value

The formatted amount based on store currency settings (as a string)

CurrencyModule_AddFormatPlainText

This function provides alternate formatting to the **CurrencyModule_AddFormatting** function. It is intended for use by email modules. Examples of implementations of **CurrencyModule_AddFormatPlainText** can be found in the **modules/currency** directory.

Syntax

CurrencyModule_AddFormatPlainText (module var, value)

Parameters

module	The **Module** record of the current module
value	The currency amount represented as a number

Return Value

The formatted amount based on store currency settings (as a string)

CurrencyModule_AddFormatPlainTextShort

This function provides alternate formatting to the **CurrencyModule_AddFormatting** function. It is intended for use by email modules. Examples of implementations of **CurrencyModule_AddFormatPlainTextShort** can be found in the **modules/currency** directory.

Syntax

CurrencyModule_AddFormatPlainTextShort (module var, value)

Parameters

module	The **Module** record of the current module
value	The currency amount represented as a number

Return Value

The formatted amount based on store currency settings (as a string)

CurrencyModule_Output_CurrencyFormat_JavaScript

This function allows a currency module to output a currency formatter in JavaScript. For example, a store with the currency module installed can format a value on the fly (after the page has loaded) rather than formatting in MivaScript or template code before output.

Syntax

CurrencyModule_Output_CurrencyFormat_JavaScript (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 if you have output the **MMCurrencyFormatter** JavaScript function and are handling the output format yourself (no fallback code will be output)

0 if you have not output the **MMCurrencyFormatter** JavaScript function and expect the fallback code to be output instead

Shopping Interface Customer Actions Feature (custrt)

Modules that implement the **Shopping Interface Customer** feature (**custrt**) are called whenever a customer record is created or modified within the shopping interface.

The **custrt** feature includes the following functions:

- **Module_Customer_Runtime_ChangeEmailAddress**
- **Module_Customer_Runtime_ChangePassword**
- **Module_Customer_Runtime_Insert**
- **Module_Customer_Runtime_Update**
- **Module_Customer_Runtime_Validate**

Module_Customer_Runtime_ChangeEmailAddress

This function is called by Miva Merchant when a customer successfully completes an email address change operation. This allows modules using the **custrt** feature to make their own internal changes when a customer changes their email address.

Supported API Version

5.73 and higher (new in PR8 Update 7)

Syntax

Module_Customer_Runtime_ChangeEmailAddress (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **customer** record of the current customer

Return Value

1 on success
0 on error

Module_Customer_Runtime_ChangePassword

This function is called by Miva Merchant when a customer successfully completes a password change operation. This allows modules using the **custrt** feature to make their own internal changes when a customer changes their password.

Supported API Version

5.73 and higher (new in PR8 Update 7)

Syntax

Module_Customer_Runtime_ChangePassword (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **customer** record of the current customer

Return Value

1 on success

0 on error

Module_Customer_Runtime_Insert

Miva Merchant calls this function when a customer creates a new customer account at runtime.

Syntax

Module_Customer_Runtime_Insert (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **customer** record of the current customer

Return Value

1 on success

0 on error

Module_Customer_Runtime_Update

Miva Merchant calls this function when a customer updates a customer account at runtime.

Syntax

Module_Customer_Runtime_Update (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **customer** record of the current customer

Return Value

1 on success

0 on error

Module_Customer_Runtime_Validate

Miva Merchant calls this function when a customer record requires validation at runtime.

Syntax

Module_Customer_Runtime_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Domain-level Module Data Support Feature (data_domain)

Modules that implement the **Domain-level Module Data Support** feature (**data_domain**) are given an opportunity to create/initialize module specific data when installed within a Miva Merchant installation, modify that data when upgraded, and delete/clean up that data when removed.

The **data_domain** feature includes the following functions:

- **Module_Install**
- **Module_Uninstall**
- **Module_Upgrade**

Module_Install

Miva Merchant calls this function when the administrator adds the module to the domain. One employs this function only when the module performs some functionality or provides data that is shared between all of the stores in a domain. The purpose of this function is to create any databases and directories that may be required.

Syntax

Module_Install (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Module_Uninstall

Miva Merchant calls this function when the administrator removes the module from the domain. It should contain the code necessary to reverse what was done during **Module_Install**. Any action that is taken in **Module_Install** should be reversed in this function. It could be deleting a database or directory or deleting a user from the licensed-user database.

Syntax

Module_Uninstall (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Module_Upgrade

Miva Merchant calls this function when the administrator clicks **Update** on the **Edit Module** configuration screen. It provides a place to define what action admin must take to upgrade from one version of the module to another.

If the module has more than two versions that could be actively used, there might be new fields in the later version of the module. In this case, the databases would have to be upgraded carefully. For example, if there are four versions in use (v1.0, v1.1, v2.0, v3.0) and you are upgrading from v1.0 to v3.0, the code should first upgrade to v1.1, then to v2.0 and finally to v3.0.

Syntax

Module_Upgrade (module var, version)

Parameters

module The **Module** record of the current module

version The currently stored version of the module

Return Value

1 on success

0 on error

Store-level Module Data Support Feature (data_store)

Modules that implement the **Store-level Module Data Support** feature (**data_store**) are given an opportunity to create/initialize module specific data when assigned to a store, modify that data when upgraded, and delete/clean up that data when unassigned.

The **Store-level Module Data Support** feature includes the following functions:

- **Module_Install_Store**
- **Module_Uninstall_Store**
- **Module_Upgrade_Store**

Module_Install_Store

Miva Merchant calls this function when the store administrator assigns the module to the store. The purpose of this function is to create any databases and directories that may be required by the module for each store location.

Syntax

Module_Install_Store (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Module_Uninstall_Store

Miva Merchant calls this function when the store administrator removes a module from the store. It should contain the code necessary to reverse what was done during **Module_Install_Store**. Every action that is taken in **Module_Install_Store** should be reversed in this function. All databases and directories that have been established should be deleted and the store should be taken out of the "installed stores" database. Any other functions that were required to establish the store must be reversed and cleaned up.

Syntax

Module_Uninstall_Store (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Module_Upgrade_Store

Miva Merchant calls this function when a store administrator clicks **Update** on the **Edit Module** configuration screen. It provides a place for code altering the databases and directories that apply to a specific store during an module upgrade.

Syntax

Module_Upgrade_Store (module var, version)

Parameters

module The **Module** record of the current module

version The version of the module before update

Return Value

1 on success

0 on error

Designer Feature (designer)

Modules that implement the **Designer** feature (**designer**) provide methods to facilitate their use with Dreamweaver.

The **designer** feature includes the following functions:

- **DesignerComponentModule_Export**
- **DesignerComponentModule_ImportProvisionLines**

DesignerComponentModule_Export

This function is called by **TemplateManager_Export_Template** for modules containing feature **designer**. Its purpose is to export the template data from a given module, as HTML text or as a file, to be edited in an external editor and/or imported later. If using **output_type** "html", you can place data into **l.start_data** and **l.end_data**. This data will then be passed back to **TemplateManager_Export_Template** and written along with the other data. If using **output_type** "file", you must provide a file name in **l.start_data** (**l.end_data** will be ignored in this case).

In order for the linking to work correctly with **output_type** "file", **DesignerComponentModule_Export** must recursively call function **TemplateManager_Export_Template** with the current template source data and the file name you are returning in **l.start_data**.

Example

```
<MvFUNCTION NAME = "DesignerComponentModule_Export" PARAMETERS = "module var,
 page var, output_location, output_path, extfile_path, item, all_settings var,
 settings var, param, output_type var, start_data var, end_data var"
 STANDARDOUTPUTLEVEL = "">
    <MvCOMMENT>
    |
    |Set l.output_type to 'file', define the filename to output to in l.start_data,
    |set the template id of your module's current template version into l.templ_id.
    |Load the current template for your module (the data you want to export).
    |Send TemplateManager_Export_Template the new filename as l.start_data,
    |and the new source data as l.template:source.
    |The l.template:source data will then be written to the new file "my-file.htm"
    |through TemplateManager_Export_Template, and linked as an external file to
    |the current file when passed back through DesignerComponentModule_Export.
    |
    </MvCOMMENT>

    <MvASSIGN NAME = "l.output_type"    VALUE = "file">
    <MvASSIGN NAME = "l.start_data"     VALUE = "my-file.htm">
    <MvASSIGN NAME = "l.templ_id"       VALUE = 123>

    <MvIF EXPR = "{ NOT [ g.Module_Feature_TUI_DB ]
     .ManagedTemplateVersion_Load_Template_Current( l.templ_id, l.template ) }">
        <MvFUNCTIONRETURN VALUE = 0>
    </MvIF>

    <MvIF EXPR = "{ NOT [ g.Module_Feature_TUI_MGR ]
     .TemplateManager_Export_Template( l.page, l.output_location, l.output_path,
     l.extfile_path, l.all_settings, l.template:source, l.start_data ) }">
        <MvFUNCTIONRETURN VALUE = 0>
    </MvIF>

    <MvFUNCTIONRETURN VALUE = 1>
</MvFUNCTION>
```

Syntax

DesignerComponentModule_Export (module var, page var, output_root_option, output_dir, extfile_dir, item, all_settings var, settings var, param, output_type var, start_data var, end_data var)

Parameters

module	The **Module** record of the current module
page	The page to be exported
output_root_option	**H** for scripts (webroot) directory **D** for data directory
output_dir	The destination path for the resulting template files (e.g., **/editable_templates/store1/mm5**). Expects a leading slash (**/**).
extfile_dir	The root destination path for referenced external files (e.g., **/editable_templates/store1**). Expects a leading slash (**/**).
item	The item code
all_settings	Page attributes, such as items
settings	Local settings within function assigned to `'l.all_settings:' $ l.item` (for example, if **l.item** = `'my_item'` then it would be **l.all_settings:my_item**)
param	Item parameters
output_type	Type of output (HTML or file, all other types are ignored)
start_data	If **output_type** is "file", **start_data** must be the filename. If **output_type** is "html", then **start_data** must contain the start-data HTML to return.
end_data	If **output_type** is "file", **end_data** is ignored. If **output_type** is "html", this variable should be populated with the end-data HTML to return (if any).

Return Value

1 on success

0 on error

DesignerComponentModule_ImportProvisionLines

This function is called indirectly by **TUI_Import_Page** to generate provision code that can then be used by the provision features library to import the data within the files you saved in the editable directory. Place the edited template/page files you wish to import into a location that you set up in the store Admin under the Template Import/Export Settings tab. From the **Pages** screen in Admin, check the "Import" checkbox next to the page to be updated and click **Update**. This function will link the import file location to the actual import code, which then (during the provision step) looks at that file location, reads the contents of the file, and imports it into the current page item's template.

Syntax

DesignerComponentModule_ImportProvisionLines (module var, page_code, name, param, settings)

Parameters

module	The **Module** record of the current module
page_code	The code of the page to be imported
name	The item code
param	Item parameters
settings	Item settings (for example, if it is a template item it contains the template name, notes, etc.)

Return Value

The XML provision code (String)

Discount Feature (discount)

The discount subsystem provides a mechanism for applying discounts based on the criteria specified. Discounts can be applied to individual items, entire baskets, shipping costs, or custom qualifying criteria.

The **discount** feature includes the following functions:

- **DiscountModule_Capabilities**
- **DiscountModule_Discount_Basket**
- **DiscountModule_Discount_Items**
- **DiscountModule_Discount_PreItems**
- **DiscountModule_Discount_Shipping**
- **DiscountModule_Discount_ShippingMethodList**
- **DiscountModule_Field**
- **DiscountModule_Fields**
- **DiscountModule_Invalid**
- **DiscountModule_Item_Eligible**
- **DiscountModule_PriceGroup_Delete**
- **DiscountModule_Prompt**
- **DiscountModule_Provision_Settings**
- **DiscountModule_Update**
- **DiscountModule_Validate**

DiscountModule_Capabilities

This function describes the functionality that the discount module provides to the discount subsystem.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Capabilities(module var, capabilities var)

Parameters

module The **Module** record of the current module

capabilities An output structure containing information about the functionality of the discount module API that the module implements. The output structure is populated with the following members:

:preitems – (Boolean) If true, the module must implement the **DiscountModule_Discount_PreItems** function
:items – (Boolean) If true, the module supports discounts to individual items and must implement the **DiscountModule_Discount_Items** function
:basket – (Boolean) If true, the module supports discounts to the overall basket and must implement the **DiscountModule_Discount_Basket** function
:tax – (Boolean) If true, the module supports sales tax
:shipping – (Boolean) If true, the module supports discounts to shipping methods and must implement both the **DiscountModule_Discount_Shipping** and the **DiscountModule_Discount_ShippingMethodList** functions
:provision_settings – (Boolean) If true, the module supports provisioning.
:eligibility – (String) Controls

- whether the **Discounted Products/Subscriptions/Categories** buttons are shown in admin;
- whether function **DiscountModule_Item_Eligible** is required.

One of the following:

- **“”** (empty string) – **NULL**=display buttons/use assignment for modules with “items”
- **“builtin”** – Always display buttons, use builtin functionality
- **“module”** – Never display buttons, call **DiscountModule_Item_Eligible**

Return Value

Ignored

DiscountModule_Discount_Basket

This function is called after all the items are discounted for modules that have the “basket” capability. For example, the **basket.mv** module uses this function to apply basket-wide discounts instead of line item discounts.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Discount_Basket(module var, pricegroup var, discount_state var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added
discount_state	Internal discounting state used when calling module helper functions

Return Value

1 on success

0 on error

DiscountModule_Discount_Items

This function is called to apply discounts to individual items for modules that have the "items" capability. For example, the **buyxgety.mv** module uses this function to apply line item discounts.

Supported API Version

Introduced in Miva Merchant 9.01

Syntax

DiscountModule_Discount_Items(module var, pricegroup var, discount_state var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added. The output structure is populated with the following members: **:id** – ID of the specific price group record **:module_id** – member returned from the given **Module_Description** function **:name** – name of the module **:config** – module-specific configuration, stored as part of the pricegroup feature
discount_state	Internal discounting state used when calling module helper functions

Return Value

1 on success

0 on error

DiscountModule_Discount_PreItems

This function is called during the discounting process, before item discounts have been applied. Discount modules that need to add or change items in the basket should do so here.

Syntax

DiscountModule_Discount_PreItems(module var, pricegroup var, discount_state var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** for which the module is being called
discount_state	The current discount state

Return Value

1 on success

0 on error

DiscountModule_Discount_Shipping

This function is called to apply discounts to a given shipping method and shipping method option for modules that have the "shipping" capability, if the price group qualifies. For example, the **shipping_basket.mv** and **shipping_product.mv** modules uses this function to apply discounts to the shipping for a basket and an individual product, respectively.

Supported API Version

Introduced in Miva Merchant 9.01

Syntax

DiscountModule_Discount_Shipping(module var, pricegroup var, discount_state var, shipping_module var, shipping_method_code)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added. The output structure is populated with the following members: **:id** – ID of the specific price group record **:module_id** – member returned from the given **Module_Description** function **:name** – name of the module **:config** – module-specific configuration, stored as part of the pricegroup feature

discount_state	Internal discounting state used when calling module helper functions
shipping_module	The **Module** record of the shipping method module
shipping_method_code	The specific code of the shipping method option to apply the discount to

Return Value

1 on success

0 on error

DiscountModule_Discount_ShippingMethodList

This function is called to apply discounts to a given shipping method and all shipping method options for modules that have the "shipping" capability, if the price group qualifies. For example, the **shipping_basket.mv** and **shipping_product.mv** modules uses this function to apply discounts to the shipping for a basket and an individual product, respectively.

Supported API Version

Introduced in Miva Merchant 9.01

Syntax

DiscountModule_Discount_ShippingMethodList(module var, pricegroup var, discount_state, methods var, method_count)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added. The output structure is populated with the following members: **:id** – ID of the specific price group record **:module_id** – member returned from the given **Module_Description** function **:name** – name of the module **:config** – module-specific configuration, stored as part of the pricegroup feature
discount_state	Internal discounting state used when calling module helper functions
methods	Array of the shipping methods
method_count	Count of the shipping methods in the **methods** parameter

Return Value

1 on success

0 on error

DiscountModule_Field

This function draws the HTML input element(s) required for configuration of a single field from the list returned by **DiscountModule_Fields**. The module outputs the required HTML directly.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Field(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **DiscountModule_Fields**.

Return Value

Ignored

DiscountModule_Fields

This function returns a comma separated list of field identifiers for module-specific fields that should be present for a given price group.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Fields(module var, pricegroup var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added

Return Value

A comma separated list of field identifiers

DiscountModule_Invalid

When **DiscountModule_Invalid** returns **0**, this function is called for each field to determine which field(s) should be displayed in the invalid state.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Invalid(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **DiscountModule_Fields**.

Return Value

1 if the specified field is invalid

0 if the specified field is valid

DiscountModule_Item_Eligible

When `capabilities:eligibility="module"`, this function is called to determine whether a given item is discounted by a given price group.

Syntax

DiscountModule_Item_Eligible(module var, pricegroup var, discount_state var, item var, eligible var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** for which the module is being called
discount_state	The current discount state
item	The item for which eligibility is to be determined
eligible	Output – set to **1** if the item is eligible, **0** if not

Return Value

1 on success

0 on error

DiscountModule_PriceGroup_Delete

This function notifies the module that a price group is being deleted. When a price group is deleted, the database layer calls this function in order to give the module an opportunity to delete any records associated with the price group.

Supported API Version

9.01 and higher (introduced in Miva Merchant 9)

Syntax

DiscountModule_PriceGroup_Delete(module var, pricegroup var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being deleted

Return Value

1 on success

0 on error

DiscountModule_Prompt

This function is called for each field return by **DiscountModule_Fields** to obtain the prompt (descriptive text displayed to the left of the field).

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Prompt(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **DiscountModule_Fields**.

Return Value

Textual prompt (HTML permitted)

DiscountModule_Provision_Settings

This function is called when a price group is created using the **PriceGroup_Add** provisioning tag and the **<Settings>** tag is present. The module is expected to implement provisioning for whatever settings are normally maintained on a price-group by price-group basis.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Provision_Settings(module var, pricegroup var, provide_xml var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added
provide_xml	Provisioning-parsed XML from the **<Settings>** tag of a **PriceGroup_Add/Update** tag

Return Value

1 on success

0 on error

DiscountModule_Update

This function is called when a price group is modified so that the module can store any changes to the module controlled price group fields.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Update(module var, pricegroup var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added

Return Value

1 on success

0 on error

DiscountModule_Validate

This function is called to validate module-provided price group fields when creating or updating a price group record from the administrative interface.

Supported API Version

Introduced in Miva Merchant 9

Syntax

DiscountModule_Validate(module var, pricegroup var)

Parameters

module	The **Module** record of the current module
pricegroup	The **PriceGroup** record being edited or **NULL** if a price group is being added

Return Value

1 on success

0 on error

Data Export Feature (export)

Modules that implement the **Data Export** feature (**export**) provide user interface and operational elements for exporting data.

The **export** feature includes the following functions:

- **ExportModule_Export**
- **ExportModule_Screen**
- **ExportModule_Validate**

ExportModule_Export

Miva Merchant calls this function when a user submits information from the Export UI screen in the administration tool. This function provides a place for code to validate any input data that is required to run the module. For example, if the module requires a database name and the fields to be exported, this function must validate that the names of the database and fields exist and can be read.

Syntax

ExportModule_Export (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 on success

0 on error

ExportModule_Screen

Miva Merchant calls this function when a user submits information from the Export UI screen in the administration tool. This function provides the Admin with instructions about exporting data in a Miva Merchant database or file to a database or file outside of the Miva Merchant environment. For example, this type of module could read the customer database and write it to a file on another computer. Any data that is available to Miva Merchant can be exported.

Syntax

ExportModule_Screen (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

ExportModule_Validate

Miva Merchant calls this function when a user submits information from the Export UI screen in the administration tool. This function provides a place for code to validate any input data that is required to run the module. For example, if the module requires a database name and the fields to be exported, this function must validate that the names of the database and fields exist and can be read.

Syntax

ExportModule_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

External Requirement Verification Feature (externalreq)

Modules can implement the **External Requirement Verification** feature (**externalreq**) to verify that external requirements (required commerce libraries, SSL support, etc.) are met before the module is assigned to a store.

The **externalreq** feature includes the following function:

- **Module_External_Requirements_Met**

Module_External_Requirements_Met

The administrative UI calls this function when displaying one of the feature configuration screens, such as:

- **System Extension Configuration**
- **Shipping Configuration**
- **Payment Configuration**
- **Order Fulfillment Configuration**
- **Logging Configuration**
- **System Extension Configuration**
- **Utilities Configuration**

If the function does not return TRUE, the Admin will not include the module in the list of those assignable to the store. The module can be coded to test for the state of some arbitrary requirement — for example, that the server runs SSL — then return FALSE if the server fails the test. This would prevent store administrators from installing the module to the store until and unless the SSL requirement is met.

Syntax

Module_External_Requirements_Met (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Feed Feature (feed)

The **feed** feature provides a mechanism for modules to generate feeds in a variety of customizable formats.

The **feed** feature contains the following functions:

- **FeedModule_Capabilities**
- **FeedModule_Delete**
- **FeeModule_Field**
- **FeedModule_Fields**
- **FeedModule_Invalid**
- **FeedModule_Output**
- **FeedModule_Output_Custom**
- **FeedModule_Prompt**
- **FeedModule_Provision_Settings**
- **FeedModule_Update**
- **FeedModule_Validate**

FeedModule_Capabilities

This function describes the functionality that the **feed** module provides to the **feed** subsystem.

Syntax

FeedModule_Capabilities (module var, capabilities var)

Parameters

module The **Module** record of the current module

capabilities An output structure containing information about the functionality of the feed module API that the module implements. The output structure is populated with the following members:

:output_custom – (Boolean) If true, the module supports execution of a feed in a custom manner. The module must implement the **FeedModule_Output_Custom** function.
:provisioning_settings – (Boolean) If true, the module supports configuration of its settings through provisioning. The module must implement the **FeedModule_Provision_Settings** function.

Return Value

Ignored

FeedModule_Delete

This function notifies the module that a feed is being deleted. When a feed is deleted, the database layer calls this function in order to give the module an opportunity to delete any records associated with the feed.

Syntax

FeedModule_Delete(module var, feed var)

Parameters

module	The **Module** record of the current module
feed	The **feed** record being deleted

Return Value

1 on success

0 on error

FeedModule_Field

This function draws the HTML input element(s) required for configuration of a single field from the list returned by **FeedModule_Fields**. The module outputs the required HTML directly.

Syntax

FeedModule_Field(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **FeedModule_Fields**.

Return Value

Ignored

FeedModule_Fields

This function returns a comma separated list of field identifiers for module-specific fields that should be present for a given feed.

Syntax

FeedModule_Fields(module var, feed)

Parameters

module	The **Module** record of the current module
field_id	The **feed** record being edited or NULL if a feed is being added

Return Value

A comma separated list of field identifiers

FeedModule_Invalid

When **FeedModule_Invalid** returns **0**, this function is called for each field to determine which field(s) should be displayed in the invalid state.

Syntax

FeedModule_Invalid(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **FeedModule_Fields**.

Return Value

1 if the specified field is invalid

0 if the specified field is valid

FeedModule_Output

This function is called when a **feed** is processed and the module does not support the **:output_custom** capability.

Syntax

FeedModule_Output(module var, feed var)

Parameters

module	The **Module** record of the current module
feed	The **feed** record being executed

Return Value

1 on success

0 on error

FeedModule_Output_Custom

This function is called when a **feed** is processed and the module supports the **:output_custom** capability.

Syntax

FeedModule_Output_Custom(module var, feed var)

Parameters

module	The **Module** record of the current module
feed	The **feed** record being executed

Return Value

1 on success

0 on error

FeedModule_Prompt

This function is called for each field returned by **FeedModule_Fields** to obtain the prompt (the descriptive text displayed to the left of the field).

Syntax

FeedModule_Prompt(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **FeedModule_Fields**.

Return Value

Textual prompt (HTML permitted)

FeedModule_Provision_Settings

This function is called when a feed is created using the **Feed_Add/Update** provisioning tag and the **<Settings>** tag is present. The module is expected to implement provisioning for whatever settings are normally maintained on a feed by feed basis.

Syntax

FeedModule_Provision_Settings(module var, feed var, provide_xml var)

Parameters

module	The **Module** record of the current module
feed	The **feed** record being edited or NULL if a feed is being added
provide_xml	Provisioning-parsed XML from the tag of a **Feed_Add/Update** tag

Return Value

1 on success

0 on error

FeedModule_Update

This function is called when a feed is modified so that the module can store any changes to the module controlled **feed** fields.

Syntax

FeedModule_Update(module var, feed var)

Parameters

module	The **Module** record of the current module
feed	The **feed** record being edited or NULL if a feed is being added

Return Value

1 on success

0 on error

FeedModule_Validate

This function is called to validate module-provided feed fields when creating or updating a **feed** record from the administrative interface.

Syntax

FeedModule_Validate(module var, feed var)

Parameters

module	The **Module** record of the current module
feed	The **feed** record being edited or NULL if a feed is being added

Return Value

1 on success

0 on error

Box Custom Fields Feature (fields_box)

Modules that implement the **Box Custom Fields** feature (**fields_box**) provide one or more Box custom fields for display in the shopping and administrative interfaces.

The **fields_box** feature contains the following functions:

- **Module_Box_Field_Capabilities**
- **Module_Box_Field_Name**
- **Module_Box_Field_Query**
- **Module_Box_Field_Query_OrderBy**
- **Module_Box_Field_Query_OrderBy_LoadIndexRecord**
- **Module_Box_Field_Query_Search**
- **Module_Box_Field_Query_Value**
- **Module_Box_Field_Value**
- **Module_Box_Field_Value_Array**
- **Module_Box_Fields**
- **Module_Box_Set_Field**
- **Module_Box_Set_Field_Array**

Module_Box_Field_Capabilities

This function is used to retrieve a specific custom field code's capabilities.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Capabilities(module var, field_code, capabilities)

Parameters

module	The custom field module
field_code	The code of the custom field
capabilities	A structure defining the capabilities related to the custom field code. **:query** – Value can retrieved via **Module_Box_Field_Query_Value** **:search** – The custom field code can be searched in the query and must support **Module_Box_Field_Query_Search** **:orderby** – The custom field code can be used in the order by the query and must support **Module_Box_Field_Query_OrderBy**

Return Value

Ignored

Module_Box_Field_Name

This function returns the name of a Custom Box Field given its code.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Name (module var, code)

Parameters

module	The **Module** record of the current module
code	The Custom Box Field code set in **Module_Box_Fields**

Return Value

A string showing the field name

Module_Box_Field_Query

This function is called when custom fields are displayed on the **Box List** screen. It allows custom field values to be loaded into the box query, thus allowing them to be searched and sorted.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Query(module var, query var, field_code)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field

Return Value

1 if query modified

0 otherwise

Module_Box_Field_Query_OrderBy

This function is called when a custom field is being used in a Box sort on the Box configuration screen.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Query_OrderBy (module var, query var, field_code, direction)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
direction	The direction to sort the query data

Return Value

1 if direction was applied to the query

0 otherwise

Module_Box_Field_Query_OrderBy_LoadIndexRecord

This function is called when a column is being ordered by the custom field code and **MMBatchList** is attempting to find the positional location of a record.

Syntax

Module_Box_Field_Query_OrderBy_LoadIndexRecord(module var, loaded_record var, box_id, code)

Parameters

module	The **Module** record of the current module
loaded_record	The record for which the index is being calculated. The value of the specified box and code needs to loaded into **l.loaded_record** under the member of the aliased custom field code in the query. For example, if your custom field query is pulling from your **sNN_MyCustomFields** table with an alias of "**mcf**", then you would need to load the value for that box and custom field code into **l.loaded_record:mcf:value**.
box_id	The ID of the box for which custom fields are being queried
code	The custom field code by which the column is ordered

Return Value

1 on success

0 on error

Module_Box_Field_Query_Search

This function is called when a custom field is being used in a Box search on the Box configuration screen.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Query_Search (module var, query var, field_code, operator, value)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
operator	The search operator to be applied
value	The value to be compared against the custom field

Return Value

1 if query modified

0 otherwise

Module_Box_Field_Query_Value

This function is used to retrieve the value loaded through the query on the Box configuration screen.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Query_Value(module var, view_name, field_code)

Parameters

module	The custom field module
view_name	The database view name (for example, “Boxes”).
field_code	The code of the custom field

Return Value

The value of the custom field associated with the specified code

Module_Box_Field_Value

This function returns the string data of a Custom Box Field given its code and the box item’s ID.

Syntax

Module_Box_Field_Value (module var, prod_id, code)

Parameters

module	The **Module** record of the current module
box_id	The ID of the box item being passed in
code	The Custom Box Field code set in **Module_Box_Fields**

Return Value

A string showing the custom field data

Module_Box_Field_Value_Array

This function is called when the specific custom field code has a **:type** of "multitext". The template manager calls this function, which allows runtime templates to detect and handle the array structure as needed.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Field_Value_Array(module var, box_id, code, values var)

Parameters

module	The **Module** record of the current module
box_id	The ID of the box item being passed in
code	The Custom Box Field code set in **Module_Box_Fields**
values	An output array that contains all values associated with the custom field for the box

Return Value

The number of elements in the values array

Module_Box_Fields

This function returns the Custom Box Fields array with the member's code and name to be used by the other functions in the Custom Box Fields API.

Syntax

Module_Box_Fields (module var, fields var)

Parameters

module	The **Module** record of the current module
fields	An array with each element in the array having the following members: **name** – The field name visible in the Admin (as on Point & Click Administration of template) **code** – The code Miva Merchant uses to identify the field

Return Value

The number of elements added to **l.fields**

Module_Box_Set_Field

This function sets the new or updated value of the passed box's custom field.

Syntax

Module_Box_Set_Field (module var, box_id, code, value)

Parameters

module	The **Module** record of the current module
box_id	The ID of the box item passed in
code	The Custom Box Field code set in function **Module_Box_Fields**
value	The new value to be set for the box's custom field based on the passed code

Return Value

1 on success

0 on error

Module_Box_Set_Field_Array

This function is called when a "multitext" custom field is modified. The administrative interface calls this function during inline editing of box custom fields.

Supported API Version

9.12.00 and higher

Syntax

Module_Box_Set_Field_Array(module var, box_id, code, values var, value_count)

Parameters

module	The **Module** record of the current module
box_id	The ID of the box item passed in
code	The Custom Box Field code set in function **Module_Box_Fields**
values	The array of values associated with the custom field for the box
value_count	The number of elements in the values array

Return Value

1 on success

0 on error

Category Custom Fields Feature (fields_cat)

Modules that implement the **Category Custom Fields** feature (**fields_cat**) provide one or more Category custom fields for display in the shopping and administrative interfaces.

The **fields_cat** feature contains the following functions:

- **Module_Category_Field_Capabilities**
- **Module_Category_Field_Name**
- **Module_Category_Field_Query**
- **Module_Category_Field_Query_OrderBy**
- **Module_Category_Field_Query_OrderBy_LoadIndexRecord**
- **Module_Category_Field_Query_Search**
- **Module_Category_Field_Query_Value**
- **Module_Category_Field_Value**
- **Module_Category_Field_Value_Array**
- **Module_Category_Fields**
- **Module_Category_Set_Field**
- **Module_Category_Set_Field_Array**

Module_Category_Field_Capabilities

This function is used to retrieve a specific custom field code's capabilities.

Supported API Version

9.07 and higher

Syntax

Module_Category_Field_Capabilities(module var, field_code, capabilities)

Parameters

module	The custom field module
field_code	The code of the custom field
capabilities	A structure defining the capabilities related to the custom field code. **:query** – Value can retrieved via **Module_Category_Field_Query_Value** **:search** – The custom field code can be searched in the query and must support **Module_Category_Field_Query_Search** **:orderby** – The custom field code can be used in the order by the query and must support **Module_Category_Field_Query_OrderBy**

Return Value

Ignored

Module_Category_Field_Name

This function returns the name of a Custom Category Field given its code.

Syntax

Module_Category_Field_Name (module var, code)

Parameters

module	The **Module** record of the current module
code	The Custom Category Field code set in **Module_Category_Fields**

Return Value

A string of the field name

Module_Category_Field_Query

This function is called when custom fields are displayed on the **Category List** screen. It allows custom field values to be loaded into the category query, thus allowing them to be searched and sorted.

Supported API Version

9.07 and higher

Syntax

Module_Category_Field_Query(module var, query var, field_code)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field

Return Value

1 if query modified

0 otherwise

Module_Category_Field_Query_OrderBy

This function is called when a custom field is being used in a Category sort on the Category configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Category_Field_Query_OrderBy (module var, query var, field_code, direction)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
direction	The direction to sort the query data

Return Value

1 if direction was applied to the query

0 otherwise

Module_Category_Field_Query_OrderBy_LoadIndexRecord

This function is called when a column is being ordered by the custom field code and **MMBatchList** is attempting to find the positional location of a record.

Syntax

Module_Category_Field_Query_OrderBy_LoadIndexRecord(module var, loaded_record var, category_id, code)

Parameters

module	The **Module** record of the current module
loaded_record	The record for which the index is being calculated. The value of the specified category and code needs to loaded into **l.loaded_record** under the member of the aliased custom field code in the query. For example, if your custom field query is pulling from your **sNN_MyCustomFields** table with an alias of "**mcf**", then you would need to load the value for that category and custom field code into **l.loaded_record:mcf:value**.
category_id	The ID of the category for which custom fields are being queried
code	The custom field code by which the column is ordered

Return Value

1 on success

0 on error

Module_Category_Field_Query_Search

This function is called when a custom field is being used in a Category search on the Category configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Category_Field_Query_Search (module var, query var, field_code, operator, value)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
operator	The search operator to be applied
value	The value to be compared against the custom field

Return Value

1 if query modified
0 otherwise

Module_Category_Field_Query_Value

This function is used to retrieve the value loaded through the query on the Category configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Category_Field_Query_Value(module var, view_name, field_code)

Parameters

module	The custom field module
view_name	The database view name (for example, “Categories”).
field_code	The code of the custom field

Return Value

The value of the custom field associated with the specified code

Module_Category_Field_Value

This function returns the string data of a Custom Category Field given its code and the category item’s ID.

Syntax

Module_Category_Field_Value (module var, cat_id, code)

Parameters

module	The **Module** record of the current module
cat_id	The ID of the category item being passed in
code	The Custom Category Field code set in **Module_Category_Fields**

Return Value

A string describing the custom field data

Module_Category_Field_Value_Array

This function is called when the specific custom field code as a **:type** of "multitext". The template manager calls this function, which allows runtime templates to detect and handle the array structure as needed.

Supported API Version

9.08 and higher

Syntax

Module_Category_Field_Value_Array (module var, category_id, code, values var)

Parameters

module	The **Module** record of the current module
category_id	The ID of the category item being passed in
code	The Custom Category Field code set in **Module_Category_Fields**
values	An output array that contains all values associated with the custom field for the category

Return Value

The number of elements in the values array

Module_Category_Fields

This function returns the Custom Category Fields array with the member's code and name to be used by the other functions in the Custom Category Fields API.

Syntax

Module_Category_Fields (module var, fields var)

Parameters

module	The **Module** record of the current module
fields	The **fields** array containing the name and code of each custom field item

Return Value

The count of added elements

Module_Category_Set_Field

This function sets the new or updated value of the passed category's custom field.

Syntax

Module_Category_Set_Field (module var, cat_id, code, value)

Parameters

module	The **Module** record of the current module
cat_id	The ID of the category item passed in
code	The Custom Category Field code set in **Module_Category_Fields**
value	The new value to be set for the category's custom field based on the passed code

Return Value

1 on success

0 on error

Module_Category_Set_Field_Array

This function is called when a "multitext" custom field is modified. The administrative interface calls this function during inline editing of category custom fields.

Supported API Version

9.08 and higher

Syntax

Module_Category_Set_Field_Array(module var, category_id, code, values var, value_count)

Parameters

module	The **Module** record of the current module
category_id	The ID of the category item passed in
code	The Custom Category Field code set in **Module_Category_Fields**
values	The array of values associated with the custom field for the category
value_count	The number of elements in the values array

Return Value

1 on success

0 on error

Multiple Category Custom Fields Feature (fields_cat_map)

The **fields_cat_map** feature adds additional functionality to the **fields_cat** feature to allow multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use.

Note: The module must provide the **fields_cat_map** feature ***and*** report an API version of 5.72.

The **fields_cat_map** feature contains the following function:

- **Module_Category_Fields_Mapped**

Module_Category_Fields_Mapped

This function allows multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use. The module should load the custom fields for the specified category and place the values into **output_fields** using the member names specified by **field_map**.

Supported API Version

Introduced in PR8 Update 5

Syntax

Module_Category_Fields_Mapped (module var, cat_id, field_map var, output_fields var)

Parameters

module	The **Module** record of the current module
cat_id	The ID of the category for which custom fields are being queried

field_map	A structure that defines a mapping between custom field codes and the member used for output in **output_fields**. For example, if the module provides two custom fields "a" and "b", and the caller wants the values for those fields to be put into the "a_value" and "b_value" members of **output_fields**, **field_map** would look like this:
	`field_map:a="a_value"`
	`field_map:b="b_value"`

output_fields	Output structure that receives the custom fields

Return Value

Ignored

Customer Custom Fields Feature (fields_cust)

Modules that implement the **Customer Custom Fields** feature (**fields_cust**) provide one or more Customer custom fields for display in the shopping and administrative interfaces. This feature is activated under **Utilities** in the Administrative Interface.

The **fields_cust** feature contains the following functions:

- **Module_Customer_Field_Capabilities**
- **Module_Customer_Field_Name**
- **Module_Customer_Field_Query**
- **Module_Customer_Field_Query_OrderBy**
- **Module_Customer_Field_Query_OrderBy_LoadIndexRecord**
- **Module_Customer_Field_Query_Search**
- **Module_Customer_Field_Query_Value**
- **Module_Customer_Field_Value**
- **Module_Customer_Field_Value_Array**
- **Module_Customer_Fields**
- **Module_Customer_Set_Field**
- **Module_Customer_Set_Field_Array**

Module_Customer_Field_Capabilities

This function is used to retrieve a specific custom field code's capabilities.

Supported API Version

9.07 and higher

Syntax

Module_Customer_Field_Capabilities(module var, field_code, capabilities)

Parameters

module The custom field module

field_code The code of the custom field

capabilities A structure defining the capabilities related to the custom field code.

:query – Value can retrieved via **Module_Customer_Field_Query_Value**
:search – The custom field code can be searched in the query and must support **Module_Customer_Field_Query_Search**
:orderby – The custom field code can be used in the order by the query and must support **Module_Customer_Field_Query_OrderBy**

Return Value

Ignored

Module_Customer_Field_Name

This function returns the name of a Custom Customer Field given its code.

Syntax

Module_Customer_Field_Name (module var, code)

Parameters

module	The **Module** record of the current module
code	The Custom Customer Field code set in **Module_Customer_Fields**

Return Value

A string showing the field name

Module_Customer_Field_Query

This function is called when custom fields are displayed on the **Customer List** screen. It allows custom field values to be loaded into the customer query, thus allowing them to be searched and sorted.

Supported API Version

9.07 and higher

Syntax

Module_Customer_Field_Query(module var, query var, field_code)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field

Return Value

1 if query modified

0 otherwise

Module_Customer_Field_Query_OrderBy

This function is called when a custom field is being used in a Customer sort on the Customer configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Customer_Field_Query_OrderBy (module var, query var, field_code, direction)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
direction	The direction to sort the query data

Return Value

1 if direction was applied to the query

0 otherwise

Module_Customer_Field_Query_OrderBy_LoadIndexRecord

This function is called when a column is being ordered by the custom field code and **MMBatchList** is attempting to find the positional location of a record.

Syntax

Module_Customer_Field_Query_OrderBy_LoadIndexRecord(module var, loaded_record var, customer_id, code)

Parameters

module	The **Module** record of the current module
loaded_record	The record for which the index is being calculated. The value of the specified customer and code needs to loaded into **l.loaded_record** under the member of the aliased custom field code in the query. For example, if your custom field query is pulling from your **sNN_MyCustomFields** table with an alias of "**mcf**", then you would need to load the value for that product and custom field code into **l.loaded_record:mcf:value**.
customer_id	The ID of the product for which custom fields are being queried
code	The custom field code by which the column is ordered

Return Value

1 on success

0 on error

Module_Customer_Field_Query_Search

This function is called when a custom field is being used in a Customer search on the Customer configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Customer_Field_Query_Search (module var, query var, field_code, operator, value)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
operator	The search operator to be applied
value	The value to be compared against the custom field

Return Value

1 if query modified
0 otherwise

Module_Customer_Field_Query_Value

This function is used to retrieve the value loaded through the query on the Customer configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Customer_Field_Query_Value(module var, view_name, field_code)

Parameters

module	The custom field module
view_name	The database view name (for example, “Customers”).
field_code	The code of the custom field

Return Value

The value of the custom field associated with the specified code

Module_Customer_Field_Value

This function returns the string data of a Custom Customer Field given its code and the customer item’s ID.

Syntax

Module_Customer_Field_Value (module var, cust_id, code)

Parameters

module	The **Module** record of the current module
cust_id	The ID of the customer item being passed in
code	The Custom Customer Field code set in **Module_Customer_Fields**

Return Value

A string describing the custom field data

Module_Customer_Field_Value_Array

This function is called when the specific custom field code has a **:type** of "multitext".

Supported API Version

9.08 and higher

Syntax

Module_Customer_Field_Value_Array(module var, customer_id, code, values var)

Parameters

module	The **Module** record of the current module
customer_id	The ID of the customer item being passed in
code	The Custom Customer Field code set in **Module_Customer_Fields**
values	An output array that contains all values associated with the custom field for the customer

Return Value

The number of elements in the values array

Module_Customer_Fields

Miva Merchant calls this function when displaying the **Customer Configuration** screen in the Administrative Interface. Its purpose is to load the **l.fields** structure with the name and code for each custom field that the module supplies. The Customer Export module also calls this function.

Syntax

Module_Customer_Fields (module var, fields var)

Parameters

module	The **Module** record of the current module
fields	An array with each element in the array having the following members: **name** – The field name visible in the Admin (as on Point & Click Administration of template) **code** – The code Miva Merchant uses to identify the field

Return Value

The number of elements added to **l.fields**

Module_Customer_Set_Field

Miva Merchant calls this function from the Administrative Interface to update the value of a custom Customer field via the **value** parameter.

Syntax

Module_Customer_Set_Field (module var, cust_id, code, value)

Parameters

module	The **Module** record of the current module
cust_id	The ID of the customer item passed in
code	The Custom Customer Field code set in **Module_Customer_Fields**
value	The new value to be set for the customer's custom field based on the passed code

Return Value

1 on success

0 on error

Module_Customer_Set_Field_Array

This function is called when a "multitext" custom field is modified. The administrative interface calls this function during inline editing of customer custom fields.

Supported API Version

9.08 and higher

Syntax

Module_Customer_Set_Field_Array(module var, customer_id, code, values var, value_count)

Parameters

module	The **Module** record of the current module
customer_id	The ID of the customer item passed in
code	The Custom Customer Field code set in **Module_Customer_Fields**
values	The array of values associated with the custom field for the customer
value_count	The number of elements in the values array

Return Value

1 on success

0 on error

Custom Fields Map Feature (fields_cust_map)

The **fields_cust_map** feature adds additional functionality to the **fields_cust** feature to allow multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use.

The **fields_cust_map** feature contains the following function:

- **Module_Customer_Fields_Mapped**

Module_Customer_Fields_Mapped

This function allows multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use. The module should load the custom fields for the specified customer and place the values in **output_fields** using the member names specified by **field_map**.

Syntax

Module_Customer_Fields_Mapped(module var, customer_id, field_map var, output_fields var)

Parameters

module	The custom field module
customer_id	The ID of the customer for which custom fields are being queried

field_map	A structure that defines a mapping between custom field codes and the member used for output in **output_fields**. For example, if the module provides two custom fields "a" and "b", and the caller wants the values for those fields to be put into the "a_value" and "b_value" members of **output_fields**, **field_map** would look like this:
	`field_map:a="a_value"`
	`field_map:b="b_value"`

output_fields	Output structure that receives the custom fields

Return Value

Ignored

Order Custom Fields Feature (fields_ordr)

Modules that implement the Order Custom Fields feature (**fields_ordr**) provide one or more Order custom fields for display in the shopping and administrative interfaces. This feature is activated

under Utilities in the Administrative Interface. The **fields_ordr** feature is analogous to the **fields_cust** feature.

The **fields_ordr** feature contains the following functions:

- **Module_Order_Field_Capabilities**
- **Module_Order_Field_Name**
- **Module_Order_Field_Query**
- **Module_Order_Field_Query_OrderBy**
- **Module_Order_Field_Query_OrderBy_LoadIndexRecord**
- **Module_Order_Field_Query_Search**
- **Module_Order_Field_Query_Value**
- **Module_Order_Field_Value**
- **Module_Order_Field_Value_Array**
- **Module_Order_Fields**
- **Module_Order_Set_Field**
- **Module_Order_Set_Field_Array**

Module_Order_Field_Capabilities

This function is used to retrieve a specific custom field code's capabilities.

Syntax

Module_Order_Field_Capabilities(module var, field_code, capabilities)

Parameters

module	The custom field module
field_code	The code of the custom field
capabilities	A structure defining the capabilities related to the custom field code.

:query – Value can retrieved via **Module_Order_Field_Query_Value**
:search – The custom field code can be searched in the query and must support **Module_Order_Field_Query_Search**
:orderby – The custom field code can be used in the order by the query and must support **Module_Order_Field_Query_OrderBy**

Return Value

Ignored

Module_Order_Field_Name

This function returns the name of a Custom Order Field given its code.

Syntax

Module_Order_Field_Name(module var, code)

Parameters

module	The custom field module
code	The Custom Order Field code set in **Module_Order_Fields**

Return Value

A string showing the field name

Module_Order_Field_Query

This function is called when custom fields are displayed on the **Order List** screen. It allows custom field values to be loaded into the order query, thus allowing them to be searched and sorted.

Syntax

Module_Order_Field_Query(module var, query var, correlation, field_code)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field

Return Value

1 if query modified

0 otherwise

Module_Order_Field_Query_OrderBy

This function is called when a custom field is being used in an Order sort on the Order configuration screen.

Syntax

Module_Order_Field_Query_OrderBy (module var, query var, field_code, direction)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
direction	The direction to sort the query data

Return Value

1 if direction was applied to the query

0 otherwise

Module_Order_Field_Query_OrderBy_LoadIndexRecord

This function is called when a column is being ordered by the custom field code and **MMBatchList** is attempting to find the positional location of a record.

Syntax

Module_Order_Field_Query_OrderBy_LoadIndexRecord(module var, loaded_record var, order_id, code)

Parameters

module	The custom field module
loaded_record	The record for which the index is being calculated. The value of the specified order and code needs to loaded into **l.loaded_record** under the member of the aliased custom field code in the query. For example, if your custom field query is pulling from your table **sNN_MyCustomFields** with an alias of "**mcf**", then you would need to load the value for that order and custom field code into **l.loaded_record:mcf:value**.
order_id	The ID of the order for which custom fields are being queried
code	The custom field code the column is ordered by

Return Value

1 on success

0 on error

Module_Order_Field_Query_Search

This function is called when a custom field is being used in a Order search on the Order configuration screen.

Syntax

Module_Order_Field_Query_Search (module var, query var, field_code, operator, value)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
operator	The search operator to be applied
value	The value to be compared against the custom field

Return Value

1 if query modified

0 otherwise

Module_Order_Field_Query_Value

This function is used to retrieve the value loaded through the query on the Order configuration screen.

Syntax

Module_Order_Field_Query_Value(module var, view_name, field_code)

Parameters

module	The custom field module
view_name	The database view name (for example, “Orders”).
field_code	The code of the custom field

Return Value

The value of the custom field associated with the specified code

Module_Order_Field_Value

This function returns the string data of a Custom Order Field given its code and the order ID.

Syntax

Module_Order_Field_Value(module var, order_id, code)

Parameters

module	The custom field module
order_id	The ID of the order item being passed in
code	The Custom Order Field code set in **Module_Order_Fields**

Return Value

A string describing the custom field data

Module_Order_Field_Value_Array

This function is called when the specific custom field code has a **:type** of "multitext". The template manager calls this function, which allows runtime templates to detect and handle the array structure as needed.

Supported API Version

9.08 and higher

Syntax

Module_Order_Field_Value_Array(module var, order_id, code, values var)

Parameters

module	The custom field module
order_id	The ID of the order item being passed in
code	The Custom Order Field code set in **Module_Order_Fields**
values	An output array that contains all values associated with the custom field for the order

Return Value

The number of elements in the values array

Module_Order_Fields

This function returns the Custom Order Fields array with the member's code and name to be used by the other functions in the Custom Order Fields API.

Syntax

Module_Order_Fields(module var, fields var)

Parameters

module	The custom field module
fields	The fields array containing the name and code of each custom field item

Return Value

The number of elements in the fields array

Module_Order_Set_Field

This function sets the new or updated value of the passed order ID's custom field.

Syntax

Module_Order_Set_Field(module var, order_id, code, value)

Parameters

module	The custom field module
order_id	The ID of the order item being passed in
code	The Custom Order Field code set in **Module_Order_Fields**
value	The new value to be set for the order's custom field based on the passed code

Return Value

1 on success

0 on error

Module_Order_Set_Field_Array

This function is called when a "multitext" custom field is modified. The administrative interface calls this function during inline editing of order custom fields.

Supported API Version

9.08 and higher

Syntax

Module_Order_Set_Field_Array(module var, order_id, code, values var, value_count)

Parameters

module	The **Module** record of the current module
order_id	The ID of the order item being passed in

code	The Custom Order Field code set in **Module_Order_Fields**
values	The new value to be set for the order's custom field based on the passed code
value_count	The number of elements in the values array

Return Value

1 on success

0 on error

Multiple Order Custom Fields Feature (fields_ordr_map)

The **fields_ordr_map** feature adds additional functionality to the **fields_ordr** feature to allow multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use.

The **fields_ordr_map** feature contains the following function:

- **Module_Order_Fields_Mapped**

Module_Order_Fields_Mapped

This function allows multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use. The module should load the custom fields for the specified order and place the values in **output_fields** using the member names specified by **field_map**.

Syntax

Module_Order_Fields_Mapped(module var, order_id, field_map var, output_fields var)

Parameters

module	The custom field module
order_id	The ID of the order for which custom fields are being queried

field_map	A structure that defines a mapping between custom field codes and the member used for output in **output_fields**. For example, if the module provides two custom fields "a" and "b", and the caller wants the values for those fields to be put into the "a_value" and "b_value" members of **output_fields**, **field_map** would look like this:
	`field_map:a="a_value"`
	`field_map:b="b_value"`

output_fields	Output structure that receives the custom fields

Return Value

Ignored

Product Custom Fields Feature (fields_prod)

Modules that implement the **Product Custom Fields** feature (**fields_prod**) provide one or more Product custom fields for display in the shopping and administrative interfaces. This feature is activated under **Utilities** in the Administrative Interface. The **fields_prod** feature is analogous to the **fields_cust** feature.

The **fields_prod** feature contains the following functions:

- **Module_Product_Field_Capabilities**
- **Module_Product_Field_Name**
- **Module_Product_Field_Query**
- **Module_Product_Field_Query_OrderBy**
- **Module_Product_Field_Query_OrderBy_LoadIndexRecord**
- **Module_Product_Field_Query_Search**
- **Module_Product_Field_Query_Value**
- **Module_Product_Field_Value**
- **Module_Product_Field_Value_Array**
- **Module_Product_Fields**
- **Module_Product_Set_Field**
- **Module_Product_Set_Field_Array**

Module_Product_Field_Capabilities

This function is used to retrieve a specific custom field code's capabilities.

Supported API Version

9.07 and higher

Syntax

Module_Product_Field_Capabilities(module var, field_code, capabilities)

Parameters

module	The custom field module

field_code The code of the custom field

capabilities A structure defining the capabilities related to the custom field code.

:query – Value can retrieved via **Module_Product_Field_Query_Value**
:search – The custom field code can be searched in the query and must support **Module_Product_Field_Query_Search**
:orderby – The custom field code can be used in the order by the query and must support **Module_Product_Field_Query_OrderBy**

Return Value

Ignored

Module_Product_Field_Name

The UI calls this function when displaying a Custom Product Field on the **Product** screen (PROD) or **Product List** screen (PLIST). It returns a string for display as a field name for the field corresponding to **l.code**.

Syntax

Module_Product_Field_Name (module var, code)

Parameters

module The **Module** record of the current module

code The Custom Product Field code set in **Module_Product_Fields**

Return Value

A string showing the field name

Module_Product_Field_Query

This function is called when custom fields are displayed on the **Product List** screen (PLIST). It allows custom field values to be loaded into the product query, thus allowing them to be searched and sorted.

Supported API Version

9.07 and higher

Syntax

Module_Product_Field_Query(module var, query var, field_code)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field

Return Value

1 if query modified

0 otherwise

Module_Product_Field_Query_OrderBy

This function is called when a custom field is being used in a Product sort on the Product configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Product_Field_Query_OrderBy (module var, query var, field_code, direction)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
direction	The direction to sort the query data

Return Value

1 if direction was applied to the query

0 otherwise

Module_Product_Field_Query_OrderBy_LoadIndexRecord

This function is called when a column is being ordered by the custom field code and **MMBatchList** is attempting to find the positional location of a record.

Syntax

Module_Product_Field_Query_OrderBy_LoadIndexRecord(module var, loaded_record var, product_id, code)

Parameters

module	The **Module** record of the current module
loaded_record	The record for which the index is being calculated. The value of the specified product and code needs to loaded into **l.loaded_record** under the member of the aliased custom field code in the query. For example, if your custom field query is pulling from your **sNN_MyCustomFields** table with an alias of "**mcf**", then you would need to load the value for that product and custom field code into **l.loaded_record:mcf:value**.
product_id	The ID of the product for which custom fields are being queried
code	The custom field code by which the column is ordered

Return Value

1 on success

0 on error

Module_Product_Field_Query_Search

This function is called when a custom field is being used in a Product search on the Product configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Product_Field_Query_Search (module var, query var, field_code, operator, value)

Parameters

module	The custom field module
query	The SQL query being built. The custom field module can operate on this value using the **SQL_Query_XXX** functions.
field_code	The code of the custom field
operator	The search operator to be applied
value	The value to be compared against the custom field

Return Value

1 if query modified

0 otherwise

Module_Product_Field_Query_Value

This function is used to retrieve the value loaded through the query on the Product configuration screen.

Supported API Version

9.07 and higher

Syntax

Module_Product_Field_Query_Value(module var, view_name, field_code)

Parameters

module	The custom field module
view_name	The database view name (for example, “Products”).
field_code	The code of the custom field

Return Value

The value of the custom field associated with the specified code

Module_Product_Field_Value

The Admin calls this function to display the value of a custom field on the **Product Configuration** screen. The UI calls this function when displaying a custom product field on the **Product** page. It returns a string for display as a field value for the field corresponding to **l.code**.

Syntax

Module_Product_Field_Value (module var, prod_id, code)

Parameters

module	The **Module** record of the current module
prod_id	The ID of the product item being passed in
code	The Custom Product Field code set in **Module_Product_Fields**

Return Value

A string showing the custom field data

Module_Product_Field_Value_Array

This function is called when the specific custom field code has a **:type** of "multitext". The template manager calls this function, which allows runtime templates to detect and handle the array structure as needed.

Supported API Version

9.08 and higher

Syntax

Module_Product_Field_Value_Array(module var, product_id, code, values var)

Parameters

module	The **Module** record of the current module
product_id	The ID of the product item being passed in
code	The Custom Product Field code set in **Module_Product_Fields**
values	An output array that contains all values associated with the custom field for the product

Return Value

The number of elements in the values array

Module_Product_Fields

Miva Merchant calls this function when displaying the **Product** configuration screen and the **Edit Page: Product Display** configuration screen in the Administrative Interface. The Product Import and Export modules also calls this function. Its purpose is to load the **l.fields** structure with the name and code for each custom field that the module supplies. When viewing the point and click administration of the Product Display Layout, the field(s) created by the module will be available to select for display.

Syntax

Module_Product_Fields (module var, fields var)

Parameters

module	The **Module** record of the current module
fields	An array with each element in the array having the following members: **name** – The field name visible in the Admin (as on Point & Click Administration of template) **code** – The code Miva Merchant uses to identify the field

Return Value

The number of elements added to **l.fields**

Module_Product_Set_Field

Admin relies on this function to instruct it how to update the value (from the **value** parameter) of a custom product field (identified by the **code** parameter) for a given product (identified by the **prod_id** parameter). The **value** comes from manual input from an administrator, an import etc.

Syntax

Module_Product_Set_Field (module var, prod_id, code, value)

Parameters

module	The **Module** record of the current module
prod_id	The ID of the product item passed in
code	The Custom Product Field code set in function **Module_Product_Fields**
value	The new value to be set for the product's custom field based on the passed code

Return Value

1 on success

0 on error

Module_Product_Set_Field_Array

This function is called when a "multitext" custom field is modified. The administrative interface calls this function during inline editing of product custom fields.

Supported API Version

9.08 and higher

Syntax

Module_Product_Set_Field_Array(module var, product_id, code, values var, value_count)

Parameters

module	The **Module** record of the current module
product_id	The ID of the product item passed in
code	The Custom Product Field code set in function **Module_Product_Fields**
values	The array of values associated with the custom field for the product
value_count	The number of elements in the values array

Return Value

1 on success

0 on error

Multiple Product Custom Fields Feature (fields_prod_map)

The **fields_prod_map** feature adds additional functionality to the **fields_prod** feature to allow multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use.

Note: The module must provide the **fields_prod_map** feature ***and*** report an API version of 5.72.

The **fields_prod_map** feature contains the following function:

- **Module_Product_Fields_Mapped**

Module_Product_Fields_Mapped

This function allows multiple custom fields to be retrieved in a single call to the module. This improves performance when there are large numbers of custom fields in use. The module should load the custom fields for the specified product and place the values into **output_fields** using the member names specified by **field_map**.

Supported API Version

5.72 and higher (new in PR8 Update 5)

Syntax

Module_Product_Fields_Mapped (module var, product_id, field_map var, output_fields var)

Parameters

module	The **Module** record of the current module
product_id	The ID of the product for which custom fields are being queried

field_map	A structure that defines a mapping between custom field codes and the member used for output in **output_fields**. For example, if the module provides two custom fields "a" and "b", and the caller wants the values for those fields to be put into the "a_value" and "b_value" members of **output_fields**, **field_map** would look like this:
	`field_map:a="a_value"`
	`field_map:b="b_value"`

output_fields	Output structure that receives the custom fields

Return Value

Ignored

Order Fulfillment Feature (fulfill)

Modules that implement the **Order Fulfillment** feature (**fulfill**) are called to perform fulfillment operations when an order is placed in the shopping interface or when an administrator manually triggers a fulfillment module from the Administrative Interface.

The **fulfill** feature contains the following function:

- **FulfillmentModule_ProcessOrder**

FulfillmentModule_ProcessOrder

Miva Merchant calls this function after a successful run of **PaymentModule_Authorize**. At a basic level, it can be used to send the customer an order acknowledgement email, or it can send the customer an invoice and the shipping department a packing list. For more advanced order processing, **FulfillmentModule_ProcessOrder** can, for example, send a parts list that defines the components necessary for a custom built computer to the manufacturing floor.

Syntax

FulfillmentModule_ProcessOrder (module var, order var)

Parameters

module	The **Module** record of the current module
order	The **order** record of the current order

Return Value

1 on success

0 on error

Data Import Feature (import)

Modules that implement the **Data Import** feature (**import**) provide user interface and operational elements for importing data. Functions within this feature obtain data from outside the Miva Merchant environment and write it to files within Miva Merchant.

The **import** feature contains the following functions:

- **ImportModule_Capabilities**
- **ImportModule_Delete**
- **ImportModule_Delimited_Columns**
- **ImportModule_Delimited_Import_Begin**
- **ImportModule_Delimited_Import_End**
- **ImportModule_Delimited_Import_Record**
- **ImportModule_Import**
- **ImportModule_Persistent_Field**
- **ImportModule_Persistent_Fields**
- **ImportModule_Persistent_Invalid**
- **ImportModule_Persistent_Prompt**
- **ImportModule_Persistent_Provision**
- **ImportModule_Persistent_StatusFields**
- **ImportModule_Persistent_Update**
- **ImportModule_Persistent_Validate**
- **ImportModule_Raw_Import_Begin**
- **ImportModule_Raw_Import_Deserialize**
- **ImportModule_Raw_Import_End**
- **ImportModule_Raw_Import_Record**
- **ImportModule_Raw_Import_Serialize**
- **ImportModule_Screen**
- **ImportModule_Validate**

ImportModule_Capabilities

This function allows the module to tell the import subsystem what functionality it implements. Modules with an API version lower than 5.70 are assumed to have the following capabilities:

screen	yes
persistent	no
format	n/a
persistent provision	no

Supported API Version

5.70 and higher

Syntax

ImportModule_Capabilities(module var, capabilities var)

Parameters

module The **Module** record of the current module

capabilities An output structure containing information about the functionality the module provides.

screen – Boolean. If true, the module implements the **ImportModule_Validate**, **ImportModule_Import**, and **ImportModule_Screen** functions from earlier API versions and displays in the left navigation menu under **Utilities/Import Data**.

persistent – Boolean. If true, the module implements the **ImportModule_Persistent_*XXX*** functions and can be used to preconfigure one or more imports displayed on the **Import Data** screen.

format – Text, valid only for modules that also support the **:persistent** capability. Must be either "delimited" or "raw". A value of "delimited" indicates that the module imports data from delimited text formats (CSV, tab delimited, etc.) and wants to make use of the import subsystem's built-in delimited text parser, in which case the module must implement the **ImportModule_Delimited_*XXX*** functions. A value of "raw" indicates that the module provides its own parser. In this case, the import subsystem will still handle the upload of the import file but will hand the raw data off to the module through the **ImportModule_Raw_*XXX*** functions.

persistent_provision – Boolean. If true, the module supports configuration of persistent imports through the provisioning system and must implement the function **ImportModule_Persistent_Provision**.

Return Value

None. The module must not return any value from this function.

ImportModule_Delete

This function notifies the module that an import is being deleted. When an import is deleted, the database layer calls this function in order to give the module an opportunity to delete any records associated with the import.

Supported API Version

9.06 and higher

Syntax

ImportModule_Delete(module var, report var)

Parameters

module The **Module** record of the current module

report The **Import** record being deleted

Return Value

1 on success

0 on error

ImportModule_Delimited_Columns

For modules that import the **:persistent** capability with a **:format** of "delimited", this function defines the list of input columns that the module expects. The import subsystem will use this list of columns to provide the configuration user interface and also to perform column-header based automatic mapping of fields.

Supported API Version

5.70 and higher

Syntax

ImportModule_Delimited_Columns(module var, import var, columns var)

Parameters

module The **Module** record of the current module

import The **Import** record of the persistent import being configured, executed, or manipulated

columns An output array of columns supported by the module. Each entry in the array has the following members:

field – A variable name that will be used to pass data to the module when performing an input. Must meet the Miva Script structure member naming requirements.

name – The name of the column. This value will be displayed to the user during import configuration.

header – The header row value for this column. When automatic column mapping is enabled, this value will be used to match columns in the input file with columns supported by the module.

Return Value

The number of entries placed into the **columns** array

ImportModule_Delimited_Import_Begin

This function is called when the import of a delimited file is begun. It allows the module to perform any import initialization, validation of auto-mapped columns or other pre-import operations that may be required.

Supported API Version

5.70 and higher

Syntax

ImportModule_Delimited_Import_Begin(module var, import var, session var, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Delimited_Import_End

This function is called at the end of a delimited import to allow the module to perform any cleanup tasks that may be required.

Supported API Version

5.70 and higher

Syntax

ImportModule_Delimited_Import_End(module var, import var, session var, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Delimited_Import_Record

During a delimited import, this function is called once for each record.

Supported API Version

5.70 and higher

Syntax

ImportModule_Delimited_Import_Record(module var, import var, session var, record var, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
record	A structure containing members populated using the data from the delimited file. Each field that is configured and present in the input file will have a single member, with the member name being equal to the **:field** member returned from **ImportModule_Delimited_Columns**.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Import

Miva Merchant calls this function when a user submits information from the Import UI screen in the administration tool. This function provides the Admin with instructions about importing the data to a Miva Merchant database from a database or file outside of the Miva Merchant environment. For example, this type of module could read a product database outside of Miva Merchant and write it to the Miva Merchant products database. Any data in the Miva Merchant database can be imported.

Syntax

ImportModule_Import (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

ImportModule_Persistent_Field

This function draws a single configuration field.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Field(module var, field_id)

Parameters

module The **Module** record of the current module

field_id The field identifier. The value is one of the fields returned by **ImportModule_Persistent_Fields**.

Return Value

Ignored

ImportModule_Persistent_Fields

This function returns a comma separated list of field identifiers that are used to draw the configuration dialog for this import. Each identifier receives a single row in the configuration dialog, with each row consisting of a “prompt” (descriptive text to the left of the field) and a “field” (input controls, etc.)

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Fields(module var, import var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being configured

Return Value

A comma separated list of field identifiers

ImportModule_Persistent_Invalid

When **ImportModule_Persistent_Validate** fails, this function is called for each field to determine whether the field should be displayed in an invalid state. Presently this means that the field's prompt is displayed in red text.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Invalid(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The identifier of the field being queried

Return Value

1 if the field should be displayed as invalid

0 if the field should be displayed as valid

ImportModule_Persistent_Prompt

This function returns the prompt (descriptive text displayed to the left of the field) for a single field from the list of fields returned by **ImportModule_Persistent_Fields**.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Prompt(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The field identifier for which to return the prompt

Return Value

The textual prompt, as it should be displayed to the user. HTML is permitted.

ImportModule_Persistent_Provision

This function performs configuration provisioning of a persistent import.

Note: The current version of the software only supports creation of persistent imports through provisioning and does not allow updates of existing imports.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Provision(module var, import var, settings_xml var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being provisioned. The module may store configuration values in the member **:config**.
settings_xml	Provisioning XML (parsed by **xml_parse**) in the same format as used by the rest of the provisioning code

Return Value

1 on success

0 on error

Important: If **0** is returned, the import subsystem will abort creation of the provisioned import.

Note: The module should report provisioning warnings or errors using **PRV_LogMessage** or **PRV_LogError**.

ImportModule_Persistent_StatusFields

While processing an import file, the import subsystem allows modules to define module-specific status fields that are displayed to the user. This function describes the status fields provided by the module.

Note: During the import process, modules can control the values of the status fields by using the **Import_Session_StatusField_*XXX*** functions provided in **features/imp/imp_ut.mv**.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_StatusFields(module var, import var, status_fields var)

Parameters

module The **Module** record of the current module

import The **Import** record being imported

status_fields An output array describing the status fields that should be displayed to the user. Each element in the array has the following members:

code – A unique code that is used to identify the status field

prompt – Descriptive text that is displayed to the left of the status field value in the import run dialog

initial – The initial value of the field

Return Value

The number of entries the module placed into the **status_fields** array

ImportModule_Persistent_Update

This function is called to update the module-specific configuration settings of a persistent import.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Update(module var, import var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being updated. Modules should place their configuration values in the **:config** member of this structure.

Return Value

1 on success

0 on error

ImportModule_Persistent_Validate

This function is called to validate the configuration fields defined by **ImportModule_Fields**.

Supported API Version

5.70 and higher

Syntax

ImportModule_Persistent_Validate(module var, import var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being configured

Return Value

1 if all fields are valid

0 if any field is invalid

Note: Modules can report validation errors through **ImportModule_Persistent_Invalid** or by calling the **FieldError** function.

ImportModule_Raw_Import_Begin

This function is called at the beginning of a persistent import using the "raw" format. It allows the module to perform any start-of-import initialization that is required.

Supported API Version

5.70 and higher

Syntax

ImportModule_Raw_Import_Begin(module var, import var, session var, filename, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
filename	The name of the import file. The uploaded import file is stored using a unique temporary filename in the **mivadata** directory.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Raw_Import_Deserialize

In order to prevent timeouts, the import subsystem will occasionally suspend and resume the import process. For delimited imports, this behavior is handled behind the scenes and requires no special handling in the import module. For raw imports, the import subsystem provides the functions **ImportModule_Raw_Import_Serialize** and **ImportModule_Raw_Import_Deserialize** to allow the module to save any state information that would be lost when the current script terminates and a new script is executed (for example, **xml_parse_section** internal state).

When the session needs to be suspended, the import subsystem calls **ImportModule_Raw_Import_Serialize**. The module should perform whatever tasks are necessary to retain its state, storing any data in the **run_data** parameter. When the import process resumes, **ImportModule_Raw_Import_Deserialize** will be called, with the same values in **run_data**, allowing the module to restore its state information (resume an **xml_parse**, etc.).

Supported API Version

5.70 and higher

Syntax

ImportModule_Raw_Import_Deserialize(module var, import var, session var, filename, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
filename	The name of the import file. The uploaded import file is stored using a unique temporary filename in the **mivadata** directory.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Raw_Import_End

This function is called at the conclusion of a raw import to allow the module to perform any post-import cleanup that is required.

Supported API Version

5.70 and higher

Syntax

ImportModule_Raw_Import_End(module var, import var, session var, filename, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
filename	The name of the import file. The uploaded import file is stored using a unique temporary filename in the **mivadata** directory.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Raw_Import_Record

This function is called to perform the actual import operations. The import subsystem will call this function repeatedly until it returns **-1** (indicating that the import is complete), or **0** (indicating an error).

Note: The import module is expected to use the **run_data** parameter to maintain variables allowing it to track its current position within the file or the internal state of the import operation.

Supported API Version

5.70 and higher

Syntax

ImportModule_Raw_Import_Record(module var, import var, session var, filename, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
filename	The name of the import file. The uploaded import file is stored using a unique temporary filename in the **mivadata** directory.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

-1 when the import has completed

ImportModule_Raw_Import_Serialize

In order to prevent timeouts, the import subsystem will occasionally suspend and resume the import process. For delimited imports, this behavior is handled behind the scenes and requires no special handling in the import module. For raw imports, the import subsystem provides the functions **ImportModule_Raw_Import_Serialize** and **ImportModule_Raw_Import_Deserialize** to allow the module to save any state information that would be lost when the current script terminates and a new script is executed (for example, **xml_parse_section** internal state).

When the session needs to be suspended, the import subsystem calls **ImportModule_Raw_Import_Serialize**. The module should perform whatever tasks are necessary to retain its state, storing any data in the **run_data** parameter. When the import process resumes, **ImportModule_Raw_Import_Deserialize** will be called, with the same values in **run_data**, allowing the module to restore its state information (resume an **xml_parse**, etc.).

Supported API Version

5.70 and higher

Syntax

ImportModule_Raw_Import_Serialize(module var, import var, session var, filename, run_data var)

Parameters

module	The **Module** record of the current module
import	The **Import** record being executed
session	An opaque structure that is passed to import subsystem helper functions to identify the import operation. The module should not manipulate this value.
filename	The name of the import file. The uploaded import file is stored using a unique temporary filename in the **mivadata** directory.
run_data	A structure containing information about this import execution. Members placed into this structure will be retained throughout the import process and passed to other **ImportModule_Delimited_Import_*XXX*** functions.

Return Value

1 on success

0 on error

ImportModule_Screen

Miva Merchant calls this function when a user submits information from the Import UI screen in the administration tool (by clicking the module's link under **Utilities > Import Data** in the Admin left menu pane). It provides Admin with instructions on how to build a UI screen for the module.

Syntax

ImportModule_Screen (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 on success

0 on error

ImportModule_Validate

Miva Merchant calls this function when a user submits information from the Import UI screen in the administration tool. This function provides a place for code to validate any input data that is required to run the module. For example, if the module requires a database name and the fields to be imported, this function must validate that the names of the database and fields exist and can be written to.

Syntax

ImportModule_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

JavaScript Object Notation Feature (json)

The **json** feature allows modules to implement functions that can be accessed via AJAX calls to **json.mvc**. The **json.mvc** file provides a mechanism for client-server development where JavaScript (or other client-side code) makes calls to functions provided via **json.mvc**. Input is generally by URI or form POST data, and output is JSON encoded. Modules can also use this mechanism to provide content without the **<DOCTYPE>** and **<HTML>** tags that are always present in pages loaded through **admin.mvc**.

Module functions are accessed through a URL to **json.mvc** with the following parameters:

- **Function** – String (required). Must always be set to **Module**.
- **Module_Code** – String (required). The code of the module that provides the JSON function.
- **Module_Function** – String (optional). A value that indicates to the module what function should be performed. Made available to the module as **g.Module_Function**.
- **Session_Type** – String (optional). If not specified, **json.mvc** does no session verification. If "runtime", **json.mvc** will load a shopping interface basket, if one exists, or create a new one. If "admin", **json.mvc** will verify that the caller has valid administrative interface credentials before calling the module.

The **json** feature contains the following function:

- **Module_JSON**

Module_JSON

This function is called when a module function is requested through **json.mvc**. By convention, the module-specific function will be indicated by **g.Module_Function**, but it is not required. The module should do whatever operation the function entails, outputting a valid JSON formatted response formatted using the helper functions in **json.mv**.

Supported API Version

Introduced in PR8 Update 5

Syntax

Module_JSON(module var)

Parameters

module	The **Module** record of the current module

Return Value

Ignored

JavaScript Object Notation Upload Feature (json_upload)

The **json_upload** feature enables uploading files to Miva Merchant via JSON.

The **json_upload** feature contains the following functions:

- **JSON_Module_Upload_ProcessFileUpload**
- **JSON_Module_Upload_ValidateFileUpload**

JSON_Module_Upload_ProcessFileUpload

Miva Merchant calls this function when an uploaded file is ready to be processed. This is only done if the file passed **JSON_Module_Upload_ValidateFileUpload**.

Syntax

JSON_Module_Upload_ProcessFileUpload(module var, field, filename, status, tempfile, content_type, size)

Parameters

module	The **Module** record of the current module
field	The name of the form input of the uploaded file
filename	The filename of the uploaded file
status	The upload status. This will be one of the following strings: • 'SKIPPED' – the file was skipped (not uploaded at all); • 'COMPLETE' – the file uploaded completely; • 'PARTIAL' – the file was partially uploaded.
tempfile	Temporary file location of the uploaded file. This file will be deleted as soon as **Miva_ProcessFileUpload()** terminates.
content_type	Declared content type of the uploaded file
size	The size of the original file (un-truncated)

Return Value

No return value

Note: If an error occurs, return it with **JSON_FileUpload_Error(error_code, error_message)**, which will set a special error code that is processed and returns the error through a JSON response.

JSON_Module_Upload_ValidateFileUpload

Miva Merchant calls this function during a JSON file upload to validate that the file being uploaded.

Syntax

JSON_Module_ValidateFileUpload(module var, field, filename, content_type)

Parameters

module	The **Module** record of the current module
field	The name of the form input of the uploaded file
filename	The filename of the uploaded file
content_type	Declared content type of the uploaded file

Return Value

0: accept unless unlimited

A positive number: sets the maximum size of the uploaded content (in bytes)

Any number less than 0: reject uploaded file

Shopping Interface Activity Logging Feature (log)

Modules that implement the **Shopping Interface Activity Logging** feature (**log**) are called to log display and action operations that occur within the shopping interface. Logging functions provide a place to insert supplementary instructions for Miva Merchant at various points during the shoppers session such as after the display of each screen, after processing each form submission and after processing a UI Exception. The logging functions are counterparts to the system functions that Miva Merchant calls before rather than after each activity. Modules can use this feature to gather information from store shoppers and write the information to a log file. The log information can then be used for statistical analysis of the store, for example traffic volume and patterns, purchase volume and patterns, etc.

The **log** feature contains the following functions:

- **LogModule_Action**
- **LogModule_Screen**
- **LogModule_UIException**

LogModule_Action

Miva Merchant calls this function after responding to a shopper submitting a form with an Action variable, such as when the shopper clicks **Add to Basket**.

Syntax

LogModule_Action (module var, action)

Parameters

module	The **Module** record of the current module
action	The action code as a string value

Return Value

1 on success

0 on error

LogModule_Screen

Miva Merchant calls this function after displaying a screen to the shopper.

Syntax

LogModule_Screen (module var, screen)

Parameters

module	The **Module** record of the current module
screen	The screen code as a string value

Return Value

1 on success

0 on error

LogModule_UIException

Miva Merchant calls this function after processing a UI Exception, for example an error with the input received from the shopper.

Syntax

LogModule_UIException (module var, code)

Parameters

module	The **Module** record of the current module
code	The exception code as a string value

Return Value

1 on success

0 on error

Category Configuration Change Notification Feature (not_cat)

Modules that implement the **Category Configuration Change Notification** feature (**not_cat**) are notified when a category is created, updated or deleted.

The **not_cat** feature contains the following functions:

- **Module_Notify_Category_Delete**
- **Module_Notify_Category_Insert**
- **Module_Notify_Category_Update**

Module_Notify_Category_Delete

This function notifies the module that a category has been deleted. When the database layer updates the status of the category, it calls this function to notify modules that have implemented the **not_cat** feature.

Syntax

Module_Notify_Category_Delete(module var, category var)

Parameters

module	The **Module** record of the current module
category	The **Category** record of the category being deleted

Return Value

Ignored

Module_Notify_Category_Insert

This function notifies the module that a category has been added. When the database layer updates the status of the category, it calls this function to notify modules that have implemented the **not_cat** feature.

Syntax

Module_Notify_Category_Insert(module var, category var)

Parameters

module	The **Module** record of the current module
category	The **Category** record of the category being inserted

Return Value

Ignored

Module_Notify_Category_Update

This function notifies the module that a category has been updated. When the database layer updates the status of the category, it calls this function to notify modules that have implemented the **not_cat** feature.

Supported API Version

9.03 and higher

Syntax

Module_Notify_Category_Update(module var, category var)

Parameters

module	The **Module** record of the current module
category	The **Category** record of the category being updated

Return Value

Ignored

Customer Configuration Change Notification Feature (not_cust)

Modules that implement the **Customer Configuration Change Notification** feature (**not_cust**) are notified when a customer is created, updated or deleted.

The **not_cust** feature contains the following functions:

- **Module_Notify_Customer_Delete**
- **Module_Notify_Customer_Insert**
- **Module_Notify_Customer_Update**

Module_Notify_Customer_Delete

This function notifies the module that a customer has been deleted. When the database layer updates the status of the customer, it calls this function to notify modules that have implemented the **not_cust** feature.

Supported API Version

PR8 and higher

Syntax

Module_Notify_Customer_Delete(module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **Customer** record of the customer being deleted

Return Value

Ignored

Module_Notify_Customer_Insert

This function notifies the module that a customer has been added. When the database layer updates the status of the customer, it calls this function to notify modules that have implemented the **not_cust** feature.

Supported API Version

PR8 and higher

Syntax

Module_Notify_Customer_Insert(module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **Customer** record of the customer being inserted

Return Value

Ignored

Module_Notify_Customer_Update

This function notifies the module that a customer has been updated. When the database layer updates the status of the customer, it calls this function to notify modules that have implemented the **not_cust** feature.

Supported API Version

9.03 and higher

Syntax

Module_Notify_Customer_Update(module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **Customer** record of the customer being updated

Return Value

Ignored

Digital Download Notification Feature (not_digital)

Modules that implement the **Digital Download Notification** feature (**not_digital**) are notified when a digital download is created or deleted.

The **not_digital** feature contains the following functions:

- **Module_Notify_DigitalDownload_Created**
- **Module_Notify_DigitalDownload_Deleted**

Module_Notify_DigitalDownload_Created

This function is called when a digital download is created. For example, **templateorderemails.mv** is called to send a notification email when a digital download is created.

Supported API Version

Introduced in Miva Merchant v9.05

Syntax

Module_Notify_DigitalDownload_Created(module var, dl var)

Parameters

module	The **Module** record of the current module
dl	Copy of the **sNN_DigitalDownload** record just created

Return Value

1 on success
0 on error

Module_Notify_DigitalDownload_Deleted

This function is called when a digital download is deleted. For example, **templateorderemails.mv** is called to send a notification email when a digital download is deleted.

Supported API Version

Introduced in Miva Merchant v9.05

Syntax

Module_Notify_DigitalDownload_Deleted(module var, dl var)

Parameters

module	The **Module** record of the current module
dl	Copy of the **sNN_DigitalDownload** record just deleted

Return Value

1 on success

0 on error

Customer Field Configuration Change Notification Feature (not_fields)

Modules that implement the **Customer Field Configuration Change Notification** feature (**not_fields**) are notified whenever the Customer Field configuration is updated. This feature allows a module to make changes to templates when an administrator modifies the displays settings for the customer fields.

The **not_fields** feature contains the following function:

- **Module_Notify_StandardFields**

Module_Notify_StandardFields

This function is called when a user changes the settings on the **Customer Fields** tab of the **Edit Store** screen or when the same settings are changed via provisioning. These settings control which customer fields are required, which customer fields are optional, whether the "ship to" or the "bill to" address is the primary address, etc.

Syntax

Module_Notify_StandardFields (module var, standardfields var)

Parameters

module	The **Module** record of the current module
standardfields	The newly updated values from the **sNN_StandardFields** table (a **standardfields** record)

Return Value

Ignored

Gift Certificate Change Notification Feature (not_giftcert)

Modules that implement the **Gift Certificate Change Notification** feature (**not_giftcert**) are notified when the status of a gift certificate is changed.

Note: The **not_giftcert** feature is new to Miva Merchant v9.03.

The **not_giftcert** feature contains the following functions:

- **Module_Notify_GiftCertificate_Created**
- **Module_Notify_GiftCertificate_Deleted**
- **Module_Notify_GiftCertificate_Redeemed**
- **Module_Notify_GiftCertificate_Updated**

Module_Notify_GiftCertificate_Created

This function notifies the module that a gift certificate has been added. When the database layer inserts a gift certificate, it calls this function to notify modules that have implemented the **not_giftcert** feature.

Syntax

Module_Notify_GiftCertificate_Created(module var, giftcert var)

Parameters

module	The **Module** record of the current module
giftcert	The **GiftCertificate** record of the gift certificate being inserted

Return Value

Ignored

Module_Notify_GiftCertificate_Deleted

This function notifies the module that a gift certificate has been deleted. When the database layer deletes a gift certificate, it calls this function to notify modules that have implemented the **not_giftcert** feature.

Syntax

Module_Notify_GiftCertificate_Deleted(module var, giftcert var)

Parameters

module	The **Module** record of the current module
giftcert	The **GiftCertificate** record of the gift certificate being deleted

Return Value

Ignored

Module_Notify_GiftCertificate_Redeemed

This function notifies the module that a gift certificate has been redeemed. When the runtime layer redeems a gift certificate, it calls this function to notify modules that have implemented the **not_giftcert** feature.

Syntax

Module_Notify_GiftCertificate_Redeemed(module var, customer var, giftcert var)

Parameters

module	The **Module** record of the current module
customer	The **Customer** record of the user redeeming the gift certificate
giftcert	The **GiftCertificate** record of the gift certificate being redeemed

Return Value

Ignored

Module_Notify_GiftCertificate_Updated

This function notifies the module that a gift certificate has been updated. When the database layer updates a gift certificate, it calls this function to notify modules that have implemented the **not_giftcert** feature.

Syntax

Module_Notify_GiftCertificate_Updated(module var, giftcert var)

Parameters

module	The **Module** record of the current module
giftcert	The **GiftCertificate** record of the gift certificate being updated

Return Value

Ignored

Image Change Notification Feature (not_image)

Modules that implement the **Image Change Notification** feature (**not_image**) are notified when the status of an image is changed.

The **not_image** feature contains the following functions:

- **Module_Notify_Image_Delete**
- **Module_Notify_Image_Insert**

Module_Notify_Image_Delete

This function notifies the module that an image has been deleted. When the database layer deletes an image, it calls this function to notify modules that have implemented the **not_image** feature.

Supported API Version

Introduced in Miva Merchant v9.03

Syntax

Module_Notify_Image_Delete(module var, image var)

Parameters

module	The **Module** record of the current module
image	The **Image** record of the image being deleted

Return Value

Ignored

Module_Notify_Image_Insert

This function notifies the module that an image has been inserted. When the database layer inserts an image, it calls this function to notify modules that have implemented the **not_image** feature.

Supported API Version

Introduced in Miva Merchant v9.03

Syntax

Module_Notify_Image_Insert(module var, image var)

Parameters

module	The **Module** record of the current module
image	The **Image** record of the image being inserted

Return Value

Ignored

Order Status Change Notification Feature (not_order)

Modules that implement the **Order Status Change Notification** feature (**not_order**) are notified when the status of an Order record is changed.

The **not_order** feature contains the following functions:

- **Module_Notify_Order_BatchChange**
- **Module_Notify_Order_Delete**
- **Module_Notify_Order_Insert**
- **Module_Notify_Order_StatusChange**
- **Module_Notify_Order_TotalChange**

Module_Notify_Order_BatchChange

This function notifies the module that one or more **Order** records have changed. When the database layer updates the batch ID of one or more orders, it calls this function to notify modules that have implemented the **not_order** feature. Multiple order records updated in a single operation (such as when all unbatched orders are batched) cause a single notification with all of the orders passed in a group.

Supported API Version

9.00 and higher

Syntax

Module_Notify_Order_BatchChange (module var, order_count, original_orders var, orders var)

Parameters

module	The **Module** record of the current module
order_count	The number of **Order** records in **original_orders** and **orders**

original_orders	An array of **Order** records representing the original state of orders
orders	An array of **Order** records representing the new state of orders

Return Value

Ignored

Module_Notify_Order_Delete

This function is called by modules implementing the **not_order** feature when an order is deleted. For example, **orderhistorynotes.mv** uses this function to create an audit trail record on order deletion.

Supported API Version

9.04 and higher

Syntax

Module_Notify_Order_Delete (module var, order var)

Parameters

module	The **Module** record of the current module
order	A copy of the order that was deleted

Return Value

Ignored

Module_Notify_Order_Insert

This function is called by modules implementing the **not_order** feature when an order is inserted. For example, **orderhistorynotes.mv** uses this function to create an audit trail record on order creation.

Supported API Version

9.04 and higher

Syntax

Module_Notify_Order_Delete (module var, order var)

Parameters

module	The **Module** record of the current module
order	A copy of the order that was inserted

Return Value

Ignored

Module_Notify_Order_StatusChange

This function notifies the module that the status of an order record has changed. When the database layer updates the status of an order, it calls this function to notify modules that have implemented the **not_order** feature.

Syntax

Module_Notify_Order_StatusChange (module var, original_status, order var)

Parameters

module	The **Module** record of the current module
original_status	The original numeric status of the order that was modified
order	The full **Order** record, post modification

Return Value

Ignored

Module_Notify_Order_TotalChange

This function notifies the module that the order total has changed. This function is called every time **Order_Update_Total** is called.

Syntax

Module_Notify_Order_TotalChange (module var, original_total, order var)

Parameters

module	The **Module** record of the current module
original_total	The original total of the order
order	The record of the order

Return Value

Ignored

OrderItem Status Change Notification Feature (not_orderitem)

Modules that implement the **OrderItem Status Change Notification** feature (**not_orderitem**) are notified when the status of one or more OrderItem records is changed.

The **not_orderitem** feature contains the following functions:

- **Module_Notify_OrderItem_Delete**
- **Module_Notify_OrderItem_Insert**
- **Module_Notify_OrderItem_StatusChange**
- **Module_Notify_OrderItem_Update**

Module_Notify_OrderItem_Delete

This function is called by Miva Merchant when an **OrderItem** record is deleted. As with **Module_Notify_OrderItem_StatusChange**, Miva Merchant only calls the function once, passing the list of order items deleted.

Supported API Version

5.73 and higher (new in PR8 Update 7)

Syntax

Module_Notify_OrderItem_Delete (module var, count, original_orderitems var, orderitems var)

Parameters

module	The **Module** record of the current module
count	The count of changed **orderitems**
original_orderitems	An array of **OrderItem** records representing the original state of the items
orderitems	An array of **OrderItem** records representing the new state of the items

Return Value

Ignored

Module_Notify_OrderItem_Insert

This function is called by modules implementing the **not_orderitem** feature when an order item is added to an order. For example, **orderhistorynotes.mv** uses this function to create an audit trail record on **OrderItem_Insert**.

Supported API Version

Introduced in Miva Merchant 9.04

Syntax

Module_Notify_OrderItem_Insert (module var, orderitem var)

Parameters

module	The **Module** record of the current module
orderitem	The **OrderItem** record being inserted

Return Value

Ignored

Module_Notify_OrderItem_StatusChange

This function notifies the module that one or more OrderItem records have changed. When the database layer updates the status of one or more OrderItems, it calls this function to notify modules that have implemented the **not_orderitem** feature. Multiple OrderItem records updated in a single operation (such as when the status of an OrderShipment record changes) cause a single notification with all of the OrderItems passed in a group.

Syntax

Module_Notify_OrderItem_StatusChange (module var, orderitem_count, original_orderitems var, orderitems var)

Parameters

module	The **Module** record of the current module
orderitem_count	The number of **OrderItem** records in **original_orderitems** and **orderitems**
original_orderitems	An array of **OrderItem** records representing the original state of the items
orderitems	An array of **OrderItem** records representing the new state of the items

Return Value

Ignored

Module_Notify_OrderItem_Update

This function is called by modules implementing the **not_orderitem** feature when an order item is updated in an order. For example, **orderhistorynotes.mv** uses this function to create an audit trail record on **OrderItem_Update**.

Supported API Version

Introduced in Miva Merchant 9.04

Syntax

Module_Notify_OrderItem_Update (module var, orderitem var)

Parameters

module	The **Module** record of the current module
original_orderitem	Copy of the **OrderItem** record before modification
orderitem	Copy of the **OrderItem** record as it was changed

Return Value

Ignored

OrderReturn Status Change Notification Feature (not_orderreturn)

Modules that implement the **OrderReturn Status Change Notification** feature (**not_orderreturn**) are notified when the status of one or more OrderReturn records are changed.

The **not_orderreturn** feature contains the following function:

- **Module_Notify_OrderReturn_Delete**
- **Module_Notify_OrderReturn_Insert**
- **Module_Notify_OrderReturn_StatusChange**

Module_Notify_OrderReturn_Delete

This function is called by modules implementing the **not_orderreturn** feature when an order return is deleted from an order. For example, **orderhistorynotes.mv** uses this function to create an audit trail record on order return deletion.

Supported API Version

9.04 and higher

Syntax

Module_Notify_OrderReturn_Delete (module var, orderreturn var)

Parameters

module	The **Module** record of the current module
orderreturn	A copy of the **OrderReturn** record that was deleted

Return Value

Ignored

Module_Notify_OrderReturn_Insert

This function is called by modules implementing the **not_orderreturn** feature when an order return is added to an order. For example, **orderhistorynotes.mv** uses this function to create an audit trail record on order return creation.

Supported API Version

9.04 and higher

Syntax

Module_Notify_OrderReturn_Insert (module var, orderreturn var)

Parameters

module	The **Module** record of the current module
orderreturn	A copy of the **OrderReturn** record that was added

Return Value

Ignored

Module_Notify_OrderReturn_StatusChange

This function notifies the module that one or more OrderReturn records have changed. When the database layer updates the status of one or more OrderReturns, it calls this function to notify modules that have implemented the **not_orderreturn** feature. Multiple OrderReturn records updated in a single operation are grouped into batches by their associated Order ID, resulting in a single notification for each unique Order.

Syntax

Module_Notify_OrderReturn_StatusChange (module var, orderreturn_count, original_orderreturns var, orderreturns var)

Parameters

module	The **Module** record of the current module
orderreturn_count	The number of **OrderReturn** records in **original_orderreturns** and **orderreturns**
original_orderreturns	An array of **OrderReturn** records representing the original state of the returns
orderreturns	An array of **OrderReturn** records representing the new state of the returns

Return Value

Ignored

OrderShipment Status Change Notification Feature (not_ordershpmnt)

Modules that implement the **OrderShipment Status Change Notification** feature (**not_ordershpmnt**) are notified when the status of one or more OrderShipment records are changed.

The **not_ordershpmnt** feature contains the following functions:

- **Module_Notify_OrderShipment_Delete**
- **Module_Notify_OrderShipment_Insert**
- **Module_Notify_OrderShipment_StatusChange**

Module_Notify_OrderShipment_Delete

This function notifies the module that an order shipment has been deleted. When the database layer deletes the order shipment, it calls this function to notify modules that have implemented the **not_ordershpmnt** feature.

Supported API Version

9.04 and higher

Syntax

Module_Notify_OrderShipment_Delete(module var, ordershipment var)

Parameters

module	The **Module** record of the current module
ordershipment	The Order Shipment record of the shipment being deleted

Return Value

Ignored

Module_Notify_OrderShipment_Insert

This function notifies the module that an order shipment has been added. When the database layer inserts the order shipment, it calls this function to notify modules that have implemented the **not_ordershpmnt** feature.

Supported API Version

9.04 and higher

Syntax

Module_Notify_OrderShipment_Insert(module var, ordershipment var)

Parameters

module	The **Module** record of the current module
ordershipment	The Order Shipment record of the shipment being inserted

Return Value

Ignored

Module_Notify_OrderShipment_StatusChange

This function notifies the module that one or more OrderShipment records have changed. When the database layer updates the status of one or more OrderShipments, it calls this function to notify modules that have implemented the **not_ordershpmnt** feature. Multiple OrderShipment records updated in a single operation are grouped into batches by their associated Order ID, resulting in a single notification for each unique Order. Notifications are sent whenever the numeric state of a shipment changes or when the tracking information or label information changes.

Syntax

Module_Notify_OrderShipment_StatusChange (module var, ordershipment_count, original_ordershipments var, ordershipments var)

Parameters

module	The **Module** record of the current module
ordershipment_count	The number of **OrderShipment** records in **original_ordershipments** and **ordershipments**

original_ordershipments	An array of **OrderShipment** records representing the original state of the shipments
ordershipments	An array of **OrderShipment** records representing the new state of the shipments

Return Value

Ignored

Payment Status Change Notification Feature (not_payment)

Modules that implement the **Payment Status Change Notification** feature (**not_payment**) are notified when the status of one or more payments has changed.

The **not_payment** feature contains the following function:

- **Module_Notify_Payment_AuthorizationFailure**

Module_Notify_Payment_AuthorizationFailure

This function notifies the module when a payment authorization fails.

Syntax

Module_Notify_Payment_AuthorizationFailure (module var, basket var, source, source_id, payment_module var, payment_module_data var, amount)

Parameters

module	The **Module** record of the current module
basket	The **Basket** record of the order that you attempted to place
source	One of the following: • "user" (admin user) • "shopper" (runtime user) • "subscription" (subscription subsystem) • "other"
source_id	One of the following: • If source is "user", then the admin ID • If source is "shopper", then the customer ID • If source is "subscription", then the subscription record ID • All others, **0**
payment_module	The **Module** record of the payment module being used

payment_module_data	The payment method code used for the payment module
amount	The amount that you attempted to authorize

Return Value

Ignored

Product Configuration Change Notification Feature (not_prod)

Modules that implement the **Product Configuration Change Notification** feature (**not_prod**) are notified when a product is created or deleted.

The **not_prod** feature contains the following functions:

- **Module_Notify_Product_Delete**
- **Module_Notify_Product_Insert**
- **Module_Notify_Product_Update**

Module_Notify_Product_Delete

This function notifies the module that a product has been deleted. When the database layer updates the status of the product, it calls this function to notify modules that have implemented the **not_prod** feature.

Supported API Version

PR8 and higher

Syntax

Module_Notify_Product_Delete(module var, product var)

Parameters

module	The **Module** record of the current module
product	The **Product** record of the product being deleted

Return Value

Ignored

Module_Notify_Product_Insert

This function notifies the module that a product has been added. When the database layer updates the status of the product, it calls this function to notify modules that have implemented the **not_prod** feature.

Supported API Version

PR8 and higher

Syntax

Module_Notify_Product_Insert(module var, product var)

Parameters

module	The **Module** record of the current module
product	The **Product** record of the product being inserted

Return Value

Ignored

Module_Notify_Product_Update

This function notifies the module that a product has been updated. When the database layer updates the status of the product, it calls this function to notify modules that have implemented the **not_prod** feature.

Supported API Version

9.03 and higher

Syntax

Module_Notify_Product_Update(module var, product var)

Parameters

module	The **Module** record of the current module
product	The **Product** record of the product being updated

Return Value

Ignored

SEO Settings Change Notification Feature (not_seo)

Modules that implement the **SEO Settings Change Notification** feature (**not_seo**) are notified when the SEO Settings configuration changes.

The **not_seo** feature contains the following function:

- **Module_Notify_SEOSettings**

Module_Notify_SEOSettings

When the domain-level SEO settings are modified, this function is called for all **not_seo** modules in each configured store, allowing modules to perform any actions that are dependent on the SEO settings. For example, the MMUI and CSSUI modules use this feature to create or destroy the SEO Sitemap page when the domain-level sitemap feature is toggled on or off.

Note: This function is called after the SEO settings have been updated.

Syntax

Module_Notify_SEOSettings (module var, seo_settings var)

Parameters

module	The **Module** record of the current module
seo_settings	A record containing the current SEO settings

Return Value

1 on success
0 on error

Subscription Settings Change Notification Feature (not_subscript)

Modules that implement the **Subscription Settings Change Notification** feature (**not_subscript**) are notified when a subscription is created, modified or deleted.

The **not_subscript** feature contains the following functions:

- **Module_Notify_Subscription_Changed**
- **Module_Notify_Subscription_Created**

- **Module_Notify_Subscription_Deleted**

Module_Notify_Subscription_Changed

This function notifies the module that a subscription has changed (e.g., the status is changed to "cancelled"). When the database layer modifies a subscription, it calls this function to notify modules that have implemented the **not_subscript** feature.

Syntax

Module_Notify_Subscription_Changed (module var, original_subscription var, subscription var)

Parameters

module	The **Module** record of the current module
original_subscription	The original subscription record
subscription	The modified subscription record

Return Value

Ignored

Module_Notify_Subscription_Created

This function notifies the module that a subscription has been created. When the database layer creates a subscription, it calls this function to notify modules that have implemented the **not_subscript** feature.

Syntax

Module_Notify_Subscription_Created (module var, subscription var)

Parameters

module	The **Module** record of the current module
subscription	The subscription record

Return Value

Ignored

Module_Notify_Subscription_Deleted

This function notifies the module that a subscription has been deleted. When the database layer deletes a subscription, it calls this function to notify modules that have implemented the **not_subscript** feature.

Syntax

Module_Notify_Subscription_Deleted (module var, subscription var)

Parameters

module	The **Module** record of the current module
subscription	The subscription record

Return Value

Ignored

URI Configuration Change Notification Feature (not_uri)

Modules that implement the **URI Configuration Change Notification** feature (**not_uri**) are notified when a URI is created, updated, or deleted. It should be noted that only modules with the **not_uri** feature in the store associated with the URI will be notified.

The **not_uri** feature contains the following functions:

- **Module_Notify_URI_Insert**
- **Module_Notify_URI_Update**
- **Module_Notify_URI_Delete**

Module_Notify_URI_Insert

This function is called when a URI is inserted.

Syntax

Module_Notify_URI_Insert(module var, uri var)

Parameters

module	The **Module** record of the current module
uri	The URI record

Return Value

Ignored

Module_Notify_URI_Update

This function is called when a URI is updated.

Syntax

Module_Notify_URI_Update(module var, uri var)

Parameters

module	The **Module** record of the current module
uri	The URI record

Return Value

Ignored

Module_Notify_URI_Delete

This function is called when a URI is deleted.

Syntax

Module_Notify_URI_Delete(module var, uri var)

Parameters

module	The **Module** record of the current module
uri	The URI record

Return Value

Ignored

Payment Processing Feature (payment)

Modules that implement the **Payment Processing** feature (**payment**) provide payment processing and order authorization functionality for the shopping and administrative interfaces. The order authorization functions mirror the payment functions but differ slightly in the handling of prompts and field validation. For example, the CVV2 field may be required by the shopping interface but optional in the administrative interface.

The **payment** feature contains the following functions:

- **PaymentModule_Authorize** (deprecated)
- **PaymentModule_Balance**
- **PaymentModule_Capabilities**
- **PaymentModule_Enabled_Methods**
- **PaymentModule_LeftNavigation**
- **PaymentModule_Manipulate_Shipping**
- **PaymentModule_Method_Capabilities**
- **PaymentModule_Order_Authorize**
- **PaymentModule_Order_Authorize_Field**
- **PaymentModule_Order_Authorize_Fields**
- **PaymentModule_Order_Authorize_Hide_Additional_Fields**
- **PaymentModule_Order_Authorize_Invalid**
- **PaymentModule_Order_Authorize_Methods**
- **PaymentModule_Order_Authorize_PaymentCard**
- **PaymentModule_Order_Authorize_Prompt**
- **PaymentModule_Order_Authorize_Validate**
- **PaymentModule_Order_Content**
- **PaymentModule_Order_Delete**
- **PaymentModule_Order_Head**
- **PaymentModule_OrderPayment_Capture**
- **PaymentModule_OrderPayment_Refund**
- **PaymentModule_OrderPayment_VOID**
- **PaymentModule_Order_Tabs**
- **PaymentModule_Order_Update**
- **PaymentModule_Order_Validate**
- **PaymentModule_Payment_Description**
- **PaymentModule_Payment_Field**
- **PaymentModule_Payment_Fields**
- **PaymentModule_Payment_Hide_Additional_Fields**
- **PaymentModule_Payment_Invalid**
- **PaymentModule_Payment_Message**
- **PaymentModule_Payment_Methods**
- **PaymentModule_Payment_Prompt**
- **PaymentModule_Payment_URL**
- **PaymentModule_Payment_Validate**
- **PaymentModule_Process** (deprecated)
- **PaymentModule_Report_Description**
- **PaymentModule_Report_Fields**

- **PaymentModule_Report_Label**
- **PaymentModule_Report_Value**
- **PaymentModule_Runtime_Authorize**
- **PaymentModule_Runtime_Authorize_PaymentCard**
- **PaymentModule_Runtime_SplitPayment_Authorize**
- **PaymentModule_Runtime_SplitPayment_Prepare**
- **PaymentModule_Runtime_SplitPayment_Rollback**

PaymentModule_Authorize

This function authorizes payment. It is called during the checkout process to perform module-specific payment operations when an order is placed from the shopping interface. It is called from the administrative interface when performing a payment operation using an obsolete module.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **PaymentModule_Runtime_Authorize**.

Supported API Version

Supported in versions less than 5.60

Syntax

PaymentModule_Authorize (module var, code, total, pay_data var, secure_data var)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**
total	The total amount of the **Order** to be authorized
pay_data	Non-secure data that is stored unencrypted in an **OrderPayment** record
secure_data	Secure data that is encrypted and stored in an **OrderPayment** record

Return Value

1 on success

0 on error

PaymentModule_Balance

This function is called by modules implementing the feature **payment** and the capability “balance”. For example, **customercredit.mv** returns the current customer credit balance.

Supported API Version

9.03 and higher

Syntax

PaymentModule_Balance(module var, data)

Parameters

module	The **Module** record of the current module
data	Module specific data

Return Value

The balance available to apply to the purchase, presented as a payment option.

PaymentModule_Capabilities

This function describes additional functionality that the module provides for the calculation of payment methods.

Supported API Version

9.03 and higher

Syntax

PaymentModule_Capabilities(module var, capabilities var)

Parameters

module	The **Module** record of the current module
capabilities	An output structure containing information about additional functionality that the module implements. The output structure is populated with the following members: **:split** – (Boolean) if true, the module supports splitting payment of an order across multiple payment options, and must implement **PaymentModule_Runtime_SplitPayment_Prepare**, **PaymentModule_Runtime_SplitPayment_Authorize**, and **PaymentModule_Runtime_SplitPayment_Rollback** **:balance** – (Boolean) if true, the module supports maintaining a balance of money available to apply to an order, and must implement **PaymentModule_Balance**

Return Value

Ignored

PaymentModule_Enabled_Methods

This function populates **methods** with a list of all supported and enabled payment methods.

Note: The methods must be enabled.

Supported API Version

9.05 and higher

Syntax

PaymentModule_Enabled_Methods (module var, methods var)

Parameters

module	The **Module** record of the current module
methods	A list of supported payment methods

Return Value

The number of methods in **methods**

PaymentModule_LeftNavigation

This function draws (optional) left navigation links. In 5.5 PR5 and later, payment module left navigation links are displayed under the Utilities item. Items can be drawn using **LeftNavigation_Dot** and **LeftNavigation_Level**.

Syntax

PaymentModule_LeftNavigation (module var, indent)

Parameters

module	The **Module** record of the current module
indent	The appropriate indentation level for module-drawn elements

Return Value

Ignored

PaymentModule_Manipulate_Shipping

This function allows a payment module to modify shipping charges or add handling charges. This function is called by **Action_PaymentManipulateShipping**, which in a default installation is called between **Action_CalculateShipping** and **Action_CalculateTax**.

Syntax

PaymentModule_Manipulate_Shipping (module var, code)

Parameters

module The **Module** record of the current module

code A payment method code returned by **PaymentModule_Payment_Methods**

Return Value

1 on success

0 on error

PaymentModule_Method_Capabilities

This function determines the capabilities of a single payment method. For example, you can determine whether a particular payment method supports MivaPay. This is useful if the module itself supports MivaPay but the particular methods do not.

Syntax

PaymentModule_Method_Capabilities (module var, method_code, capabilities var)

Parameters

module The **Module** record of the current module

method_code The payment method code to be used with MivaPay

capabilities An output structure containing information about the functionality the module provides.

mivapay (Boolean) – If true, the payment method supports being used with MivaPay

Return Value

Ignored

PaymentModule_Order_Authorize

This function performs an authorization when requested through the administrative interface. When called, the module should perform whatever operations are required when a new authorization is created for an order from the administrative interface.

Supported API Version

5.60 and higher

Syntax

PaymentModule_Order_Authorize (module var, code, order var, amount)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**
order	The **Order** record of the order being displayed
amount	The amount to be authorized

Return Value

1 on success

0 on error

Note: Error messages should be returned through **Error**.

PaymentModule_Order_Authorize_Field

Miva Merchant calls this function when displaying payment information in the administrative interface.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Field(module var, order var, pay_data, field_id)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
pay_data	A payment method code returned by **PaymentModule_Order_Authorize_Methods**
field_id	A field identifier returned by **PaymentModule_Order_Authorize_Fields**

Return Value

1 on success

0 on error

PaymentModule_Order_Authorize_Fields

This function defines the input fields required for an order in the administrative interface.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Fields(module var, order var, pay_data)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
pay_data	A payment method code returned by **PaymentModule_Order_Authorize_Methods**

Return Value

A comma separated list of module-specific field identifiers

PaymentModule_Order_Authorize_Hide_Additional_Fields

This function hides module specific input fields — for example, fields that provide configuration information and status to the external authorization process.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Hide_Additional_Fields(module var, order var, pay_data)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
pay_data	A payment method code returned by **PaymentModule_Order_Authorize_Methods**

Return Value

1 on success

0 on error

PaymentModule_Order_Authorize_Invalid

This function indicates whether or not invalid data has been entered on an order form.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Invalid(module var, order var, pay_data, field_id)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
pay_data	A payment method code returned by **PaymentModule_Order_Authorize_Methods**
field_id	A field identifier returned by **PaymentModule_Order_Authorize_Fields**

Return Value

1 if the field value is invalid

0 if the field value is valid

PaymentModule_Order_Authorize_Methods

The administrative interface calls this function to display payment methods for order authorization. It loads the **l.methods** structure with information about available payment methods offered via this module, such as **name** and **code**.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Methods(module var, order var, methods var)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
methods	A variable that receives an array of supported payment methods

Return Value

The count of elements in **methods**.

PaymentModule_Order_Authorize_PaymentCard

This function performs an authorization of an order payment with a stored MivaPay payment card. This function is used in place of **PaymentModule_Order_Authorize** when MivaPay is implemented. When called, the module should perform whatever operations are required when a new authorization is created for an order from the administrative interface.

Supported API Version

9.06 and higher

Syntax

PaymentModule_Order_Authorize_PaymentCard (module var, module_data, order var, paymentcard var, total)

Parameters

module	The **Module** record of the current module
module_data	The payment method code
order	The **Order** record of the order being displayed
paymentcard	The payment card record being used for the order authorization
total	The total amount to be authorized

Return Value

1 on success

0 on error

Note: Error messages should be returned through **Error**.

PaymentModule_Order_Authorize_Prompt

This function provides a prompt for the field named in the **field_id** parameter originally named in **PaymentModule_Order_Authorize_Fields**.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Prompt(module var, order var, pay_data, field_id)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
pay_data	A payment method code returned by **PaymentModule_Order_Authorize_Methods**.
field_id	A field identifier returned by **PaymentModule_Order_Authorize_Fields**.

Return Value

A text prompt that is displayed to the user

PaymentModule_Order_Authorize_Validate

This function validates payment fields in the administrative interface. When called, the module verifies that all input fields drawn by **PaymentModule_Order_Authorize_Field** for the specified payment method were filled out correctly, maintaining whatever internal state is required for **PaymentModule_Order_Authorize_Invalid** to indicate specific fields that failed validation. Typically, a payment module will not interact with an external gateway for this stage of the validation and will instead report gateway errors when **PaymentModule_Authorize** or **PaymentModule_Runtime_Authorize** is called.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Authorize_Validate(module var, order var, pay_data)

Parameters

module	The **Module** record of the current module
order	The **Order** record of the order being displayed
pay_data	A payment method code returned by **PaymentModule_Order_Authorize_Methods**

Return Value

1 if all fields pass validation

0 if any fields fail validation

PaymentModule_Order_Content

This function renders the content of a specific tab. If the current value of **tab** does not match one or more of the module's tabs, the content for those tabs should be output using hidden form fields.

Syntax

PaymentModule_Order_Content (module var, tab, load_fields, pay_data, secure_data)

Parameters

module	The **Module** record of the current module
tab	The code of the currently displayed tab
load_fields	A boolean value indicating if initial values for all input fields should be loaded from the database
pay_data	The non-secure payment data from the associated **OrderPayment** record
secure_data	The decrypted secure payment data from the associated **OrderPayment** record

Return Value

1 on success

0 on error

PaymentModule_Order_Delete

This function is called when an Order is deleted. When an order containing multiple OrderPayment records is deleted, this function is called for each OrderPayment within the Order.

Note: Because the order being deleted is not passed to the module as a parameter, the order must be inferred by examining global variables. Refer to *Appendix D: Deleting Orders* on page 383 for further details.

Syntax

PaymentModule_Order_Delete (module var, pay_data, secure_data)

Parameters

module	The **Module** record of the current module
pay_data	The non-secure payment data from the associated **OrderPayment** record
secure_data	The decrypted secure payment data from the associated **OrderPayment** record

Return Value

1 if all fields pass validation

0 if any fields fail validation

PaymentModule_Order_Head

This function allows the module to output content in the HTML **<head>** tag of the Legacy Order Processing and new Order Management tab screens. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

PaymentModule_Order_Head(module var, tab, order var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
order	The **Order** record of the order being displayed

Return Value

1 on success

0 on error

PaymentModule_Order_Tabs

This function returns a string that identifies the tabs to be added to the default list of system generated tabs. The string takes the following form:

```
<code>:<Description>,<code2>:<Description2>
```

The module can specify as many *<code>:<Description>* pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Syntax

PaymentModule_Order_Tabs (module var)

Parameters

module	The **Module** record of the current module

Return Value

A string describing the tabs to be added (see description above)

PaymentModule_Order_Update

This function updates content from module-provided order tab(s). It is called when an order is updated using the legacy order processing or when data from a module-specific tab is saved.

Syntax

PaymentModule_Order_Update (module var, pay_data var, secure_data var)

Parameters

module	The **Module** record of the current module
pay_data	The non-secure payment data from the associated **OrderPayment** record
secure_data	The decrypted secure payment data from the associated **OrderPayment** record

Return Value

1 on success

0 on error

PaymentModule_Order_Validate

This function validates content on module-provided order tab(s). It is called when an order is updated using the legacy order processing, or when data from a module-specific tab is saved.

Syntax

PaymentModule_Order_Validate (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 on success

0 on error

PaymentModule_OrderPayment_Capture

This function performs a full or split capture of a previous authorization when requested through the administrative interface. When called, the module should perform whatever operations are required to capture the provided authorization.

Supported API Version

5.60 and higher

Syntax

PaymentModule_OrderPayment_Capture (module var, order var, auth_payment var, auth_pay_data var, auth_secure_data var, amount)

Parameters

module	The **Module** record of the current module
order	The **Order** for which payment is being captured
auth_payment	The **OrderPayment** record that is being captured
auth_pay_data	The deserialized **pay_data** from **auth_payment**
auth_secure_data	The decrypted and deserialized **pay_secdat** from **auth_payment**
amount	The amount to be authorized

Return Value

1 on success

0 on error

Note: Error messages should be returned through **Error**.

PaymentModule_OrderPayment_Refund

This function performs a full or partial refund of a previous capture when requested through the administrative interface. When called, the module should perform whatever operations are required to refund the provided capture.

Supported API Version

5.60 and higher

Syntax

PaymentModule_OrderPayment_Refund (module var, order var, capture_payment var, capture_pay_data var, capture_secure_data var, amount)

Parameters

module	The **Module** record of the current module
order	The **Order** for which payment is being refunded
capture_payment	The **OrderPayment** record that is being refunded
capture_pay_data	The deserialized **pay_data** from **capture_payment**
capture_secure_data	The decrypted and deserialized **pay_secdat** from **capture_payment**
amount	The amount to be refunded

Return Value

1 on success

0 on error

Note: Error messages should be returned through **Error**.

PaymentModule_OrderPayment_VOID

This function performs a full or partial void of a previous authorization when requested through the administrative interface. When called, the module should perform whatever operations are required to void the provided authorization.

Supported API Version

5.60 and higher

Syntax

PaymentModule_OrderPayment_VOID (module var, order var, auth_payment var, auth_pay_data var, auth_secure_data var, amount)

Parameters

module	The **Module** record of the current module
order	The **Order** for which payment is being voided
auth_payment	The **OrderPayment** record that is being voided
auth_pay_data	The deserialized **pay_data** from **auth_payment**
auth_secure_data	The decrypted and deserialized **pay_secdat** from **auth_payment**
amount	The amount to be voided

Return Value

1 on success

0 on error

Note: Error messages should be returned through **Error**.

PaymentModule_Payment_Description

This function describes a payment method provided by this module.

Syntax

PaymentModule_Payment_Description (module var, code)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**

Return Value

A string describing the payment method

PaymentModule_Payment_Field

This function draws a checkout input field. When called, the module outputs any HTML required to display the specified checkout field.

Syntax

PaymentModule_Payment_Field (module var, code, field_id)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**
field_id	A field identifier returned by **PaymentModule_Payment_Fields**

Return Value

1 on success

0 on error

PaymentModule_Payment_Fields

This function defines input fields to be used during the checkout process.

Syntax

PaymentModule_Payment_Fields (module var, code)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**

Return Value

A comma separated list of module-specific field identifiers

PaymentModule_Payment_Hide_Additional_Fields

This function hides module-specific input fields. Payment modules that transfer control to an external checkout process frequently need to pass additional hidden form fields which provide configuration information and status to the external process. This function provides the module with an opportunity to hide any additional form fields that need to be submitted when transferring to the URL returned by **PaymentModule_Payment_URL**.

Syntax

PaymentModule_Payment_Hide_Additional_Fields (module var, code)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**

Return Value

1 on success

0 on error

PaymentModule_Payment_Invalid

This function indicates whether a checkout field should be flagged as invalid.

Syntax

PaymentModule_Payment_Invalid (module var, code, field_id)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**
field_id	A field identifier returned by **PaymentModule_Payment_Fields**

Return Value

1 if the value entered in the field is invalid

0 if the value entered in the field is valid

PaymentModule_Payment_Message

This function allows an (optional) informational message to be displayed. If the function returns a non-empty string, it is displayed prior to any payment fields during the checkout process.

Syntax

PaymentModule_Payment_Message (module var, code)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**

Return Value

An informational message or an empty string

PaymentModule_Payment_Methods

This function returns an array of payment methods supported by this module.

Syntax

PaymentModule_Payment_Methods (module var, methods var)

Parameters

module	The **Module** record of the current module
methods	An array of structures containing the following members: **code** – A module-specific code identifying this payment method **name** – A descriptive name for the payment method that will be displayed to the shopper

Return Value

The count of elements in **methods**

PaymentModule_Payment_Prompt

This function provides a prompt for a checkout input field.

Syntax

PaymentModule_Payment_Prompt (module var, code, field_id)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**
field_id	A field identifier returned by **PaymentModule_Payment_Fields**

Return Value

A textual prompt that is displayed to the shopper

PaymentModule_Payment_URL

This function provides support for external payment processing systems. If a payment module needs to hand control of the checkout process to an external website, it may do so by returning a URL from this function. If a URL is returned, the form in the checkout interface where payment information is collected will be submitted to that URL rather than Miva Merchant.

Syntax

PaymentModule_Payment_URL (module var)

Parameters

module The **Module** record of the current module

Return Value

A URL (if required) or an empty string to continue the checkout process within Miva Merchant

PaymentModule_Payment_Validate

This function validates checkout fields. When called, the module should verify that all input fields drawn by **PaymentModule_Payment_Field** for the specified payment method were filled out correctly maintaining whatever internal state is required for **PaymentModule_Payment_Invalid** to indicate specific fields that failed validation.

Typically, a payment module will not interact with an external gateway for this stage of the validation, and will instead report gateway errors when **PaymentModule_Authorize** or **PaymentModule_Runtime_Authorize** is called.

Syntax

PaymentModule_Payment_Validate (module var, code)

Parameters

module The **Module** record of the current module

code A payment method code returned by **PaymentModule_Payment_Methods**

Return Value

1 if all fields pass validation

0 if any fields fail validation

PaymentModule_Process

This function captures payment information. It is called when processing orders using the legacy order processing interface or when capturing a legacy authorization from the administrative interface. Information regarding the state of the shopping session can be obtained from the global variable **Basket**.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **PaymentModule_OrderPayment_Capture**.

Supported API Version

Supported in versions less than 5.60

Syntax

PaymentModule_Process (module var, pay_data var, secure_data var, order var)

Parameters

module	The **Module** record of the current module
pay_data	Non-secure payment data from the associated **OrderPayment** record
secure_data	Decrypted secure payment data from the associated **OrderPayment** record
order	The **Order** record with which the payment transaction is associated

Return Value

1 on success

0 on error

Note: Descriptive error messages can be reported by calling **Message_Error**.

PaymentModule_Report_Description

This function provides a textual description of a payment record for reports.

Syntax

PaymentModule_Report_Description (module var, pay_data)

Parameters

module	The **Module** record of the current module
pay_data	Non-secure payment data from the associated **OrderPayment** record

Return Value

A string containing a description of the payment method

PaymentModule_Report_Fields

This function defines transaction fields to be displayed in reports and order detail.

Syntax

PaymentModule_Report_Fields (module var, pay_data, secure_data)

Parameters

module	The **Module** record of the current module
pay_data	Non-secure payment data from the associated **OrderPayment** record
secure_data	Decrypted secure payment data from the associated **OrderPayment** record

Return Value

A comma separated list of module-specific field identifiers

PaymentModule_Report_Label

This function provides a label for a field displayed in reports and order detail.

Syntax

PaymentModule_Report_Label (module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	A field identifier returned by **PaymentModule_Report_Fields**

Return Value

A text label that is displayed to the user

PaymentModule_Report_Value

This function provides a value for a field displayed in reports and order detail.

Syntax

PaymentModule_Report_Value (module var, field_id, pay_data, secure_data)

Parameters

module	The **Module** record of the current module
field_id	A field identifier returned by **PaymentModule_Report_Fields**
pay_data	Non-secure payment data from the associated **OrderPayment** record
secure_data	Decrypted secure payment data from the associated **OrderPayment** record

Return Value

A value that is displayed to the user

PaymentModule_Runtime_Authorize

This function performs an authorization when an order is created through the shopping interface. When called, the module should perform whatever operations are desired when an order is placed within the shopping interface and create one or more OrderPayment records reflecting the results. Information regarding the state of the shopping session can be obtained from the global variable **Basket**.

Note: Modules that implement this function should still implement **PaymentModule_Authorize** for interoperability.

Supported API Version

5.60 and higher

Syntax

PaymentModule_Runtime_Authorize (module var, code, total)

Parameters

module	The **Module** record of the current module
code	A payment method code returned by **PaymentModule_Payment_Methods**
total	The total amount of the **Order** to be authorized

Return Value

1 on success

0 on error

Note: Descriptive error messages can be reported by calling **Message_Error**.

PaymentModule_Runtime_Authorize_PaymentCard

This function performs an authorization when an order is created through the shopping interface with a stored MivaPay payment card. This function is used in place of **PaymentModule_Runtime_Authorize** when MivaPay is implemented. When called, the module should perform whatever operations are desired when an order is placed within the shopping interface and create one or more OrderPayment records reflecting the results. Information regarding the state of the shopping session can be obtained from the global variable **Basket**.

Supported API Version

9.06 and higher

Syntax

PaymentModule_Runtime_Authorize_PaymentCard (module var, module_data, paymentcard var, total)

Parameters

module	The **Module** record of the current module
module_data	Module specific information (such as a method code) to identify a payment method within the module
paymentcard	The payment card record being used for the runtime authorization
total	The total amount of the **Order** to be authorized

Return Value

1 on success

0 on error

Note: Descriptive error messages can be reported by calling **Message_Error**.

PaymentModule_Runtime_SplitPayment_Authorize

This function performs authorization (capture) of a split payment.

For example, **customercredit.mv** debits the amount of "total" from the customer's account credit or calls **[g.Module_Library_Utilities].Message_Error** with "Insufficient account credit balance" if the total exceeds the customer's credit balance.

Supported API Version

9.03 and higher

Syntax

PaymentModule_Runtime_SplitPayment_Authorize(module var, module_data, total, split_payment_data var)

Parameters

module	The **Module** record of the current module
module_data	Module-specific information, such as a method code, to identify a payment method within the module
total	Amount to charge the module. This may not be the full amount of the order.
split_payment_data	Module-specific context common to **PaymentModule_Runtime_SplitPayment_Authorize**, **PaymentModule_Runtime_SplitPayment_Prepare**, and **PaymentModule_Runtime_SplitPayment_Rollback**

Return Value

1 on success

0 on error

PaymentModule_Runtime_SplitPayment_Prepare

This function prepares a split payment against a given payment method.

For example, **customercredit.mv** determines if the customer account credit is equal or greater than the "total" passed in or calls **[g.Module_Library_Utilities].Message_Error** with "Insufficient account credit balance" if "total" exceeds the customer's credit balance.

Supported API Version

9.03 and higher

Syntax

PaymentModule_Runtime_SplitPayment_Prepare(module var, module_data, total, split_payment_data var)

Parameters

module	The **Module** record of the current module
module_data	Module-specific information, such as a method code, to identify a payment method within the module
total	Amount to charge against the payment method.
split_payment_data	Module-specific context common to **PaymentModule_Runtime_SplitPayment_Authorize**, **PaymentModule_Runtime_SplitPayment_Prepare**, and **PaymentModule_Runtime_SplitPayment_Rollback**

Return Value

1 on success

0 on error

PaymentModule_Runtime_SplitPayment_Rollback

This function rolls back a split payment against a given payment method.

For example, **customercredit.mv** deletes the **OrderPayment** and **CustomerCreditHistory** information and credits the customer's account with the amount in "total".

Supported API Version

9.03 and higher

Syntax

PaymentModule_Runtime_SplitPayment_Rollback(module var, module_data, total, split_payment_data var)

Parameters

module	The **Module** record of the current module
module_data	Module-specific information, such as a method code, to identify a payment method within the module
total	Amount to charge against to roll back.
split_payment_data	Module-specific context common to **PaymentModule_Runtime_SplitPayment_Authorize**, **PaymentModule_Runtime_SplitPayment_Prepare**, and **PaymentModule_Runtime_SplitPayment_Rollback**

Return Value

1 on success

0 on error

Product Facet Feature (prod_facet)

Modules that implement the **Product Facet** feature (**prod_facet**) enable faceted navigation on pages that display a product list. This feature allows a module to provide facets which are used in runtime search. It includes functions to provide a list of facets and facet values for a given list of products to be displayed in runtime, as well as functions to modify the search query used to search for products once a facet value is selected.

The **prod_facet** feature includes the following functions:

- **Module_Product_Facets**
- **Module_Product_Facet_Query_Search**
- **Module_Product_Facet_Value_Prompt**
- **Module_Product_Facet_Value_Prompt_JavaScript**
- **Module_Product_Facet_Value_Selected**
- **Module_Product_Facet_Values**
- **Module_Product_Facet_Values_Query**

Module_Product_Facets

This function returns all facets provided by the module.

Syntax

Module_Product_Facets (module var, facets var)

Parameters

module	The **Module** record of the current module
facets	The output structure is populated with the following members: **:code** – The facet code **:name** – The facet name

Return Value

The number of facets in the **facets** array

Module_Product_Facet_Query_Search

This function applies the specified facet and value to a query.

Syntax

Module_Product_Facet_Query_Search(module var, query var, facet_code, facet_values)

Parameters

module	The **Module** record of the current module
query	Modules may modify the query as needed (it will be discarded after the call).
facet_code	The code of the facet being applied to the search results
facet_values	An array of values to be applied to the search results. If more than one value, the query should return products that have any of the input values.

Return Value

1 on success

0 on error

Module_Product_Facet_Value_Prompt

This function allows a facet module to provide formatting or other transformation of a facet value before it is displayed to the user. When the runtime code displays facet values, it calls this function for each value being displayed.

Syntax

Module_Product_Facet_Value_Prompt(module var, facet var, value var)

Parameters

module	The **Module** record of the current module
facet	The facet to which the value belongs
value	The facet value which is to be modified

Return Value

The modified/formatted facet value. If the module does not need to make any modifications, it should just return "value" directly.

Module_Product_Facet_Value_Prompt_JavaScript

Output in the head tag, this function allows a module to output any JavaScript necessary for prompt or value formatting of a facet. For example, the **stdfacets** module uses this function to format all price facet values using the currency formatter. Another module could do something similar or it could use this function to map nonsensical data to human readable data, etc.

Syntax

Module_Product_Facet_Value_Prompt_JavaScript(module var)

Parameters

module	The **Module** record of the current module

Return Value

None

Module_Product_Facet_Value_Selected

This function is called when displaying facet values in runtime to determine whether a given value should be displayed in a selected state.

Syntax

Module_Product_Facet_Value_Selected(module var, facet var, value var, input_values var, input_value_count)

Parameters

module	The **Module** record of the current module
facet	The facet to which the value belongs
value	The value in question
input_values	An array of selected values
input_value_count	The number of elements in **input_values**

Return Value

1 if the value specified by **value** should be displayed as selected

0 if it should not

Module_Product_Facet_Values_Query

This function returns all of the values for the specified facet that are assigned to one of the products in the query.

Syntax

Module_Product_Facet_Values_Query(module var, query var, facet_code, values var)

Parameters

module	The **Module** record of the current module
query	Modules may modify the query as needed (it will be discarded after the call).
facet_code	The code of the facet for which values should be returned
values	The output structure is populated with the following members: **:value** – The facet value **:count** – The number of values (optional)

Return Value

The number of facet values put into the **values** parameter

Module Provisioning Feature (provision_store)

Modules that implement the **Module Provisioning** feature (**provision_store**) provide functionality available through the provisioning system. Provisioning allows automatic instructions for the manipulation of data in the store. The upgrade system makes use of the provisioning system to transfer data from one store to another.

The **provision_store** feature includes the following function:

- **Module_Provision_Store**

Module_Provision_Store

This function allows a module to extend the Miva Merchant XML provisioning language to support functionality specific to the module. Module specific provisioning functions are triggered by the **<Module>** provisioning tag.

Example

```
<Module code="example_module" feature="util">
    <example_module_tag>content</example_module_tag>
</Module>
```

Syntax

Module_Provision_Store (module var, provide_xml var)

Parameters

module	The **Module** record of the current module
provide_xml	The module-specific XML content between the start and end module tags (see the example above). This data is passed in the format returned by the **xml_parse()** Miva Script function.

Return Value

Ignored. The module should use the **PRV_LogMessage** or **PRV_LogError** functions in **features/prv/prv_ad.mvc** to report errors.

Report Feature (report)

The report subsystem provides a mechanism for the generation, display, and export of reports. The report subsystem implements much of common functionality allowing report module developers to focus on the data itself.

Items that the report subsystem implements include:

- A common configuration interface for reports;
- Handling of date ranges and date intervals for the report module, including proper handling of daylight savings time, leap years, and leap seconds;
- A common data store (the **sNN_ReportData** table) for reports to store numeric data;
- SVG based line charts;
- **libgd** generated PNG line charts;
- **libgd** generated pie charts;
- CSV and XLS export;
- Periodic refresh of report data.

Report module developers may choose to use some or all of these features.

The **report** feature includes the following functions:

- **ReportModule_Calculate_DateRange_All**
- **ReportModule_Capabilities**
- **ReportModule_Chart_Type**
- **ReportModule_Delete**
- **ReportModule_Display**
- **ReportModule_Export**
- **ReportModule_Field**
- **ReportModule_Fields**
- **ReportModule_Format_Vertical_Label**
- **ReportModule_HTML_Chart**
- **ReportModule_Invalid**
- **ReportModule_Prompt**
- **ReportModule_Provision_Settings**
- **ReportModule_Run**
- **ReportModule_Run_DateRange**
- **ReportModule_Run_Intervals**
- **ReportModule_SVG_Line_Chart_Definition**
- **ReportModule_Tabular_Definition**
- **ReportModule_Update**
- **ReportModule_Validate**

ReportModule_Calculate_DateRange_All

When the module implements the date_range capability, this function is called when the user has configured a report with a date range of "All Dates" to determine the earliest and latest date that will be present in the generated report.

Supported API Version

5.70 and higher

Syntax

ReportModule_Calculate_DateRange_All(module var, report var, time_t_start, time_t_end)

Parameters

module	The **Module** record of the current module
report	The **Report** record being run
time_t_start	The module should set this parameter to the earliest date from its data set. The value is in UNIX **time_t** format.
time_t_end	The module should set this parameter to the latest date from its data set. The value is in UNIX **time_t** format.

Return Value

1 on success

0 on error

ReportModule_Capabilities

This function describes the functionality that the report module provides to the report subsystem.

Supported API Version

5.70 and higher

Syntax

ReportModule_Capabilities(module var, capabilities var)

Parameters

module	The **Module** record of the current module
capabilities	An output structure describing the functionality of the report module API that the module implements. The structure should be populated with the following members: **date_range** – (Boolean) Determines whether the report can be limited to a particular range of dates. If false, the module must implement the **ReportModule_Run** function. If true, the module must implement the **ReportModule_Calculate_DateRange_All** function and either **ReportModule_Run_DateRange** (if **date_interval** is false) or **ReportModule_Run_Intervals** (if **date_interval** is true). **date_interval** – (Boolean) Determines whether the report data can be grouped by a date interval (hour, day, week, month, year). Only meaningful if **date_range** is also true. If true, the module must implement the **ReportModuel_Run_Intervals** function.

display – (Boolean) Indicates whether the module supports main-screen display or not. If true, the module must implement the **ReportModule_Display** function.

output_chart – (Boolean) Indicates whether the module supports some form of chart output or other visualization. If true, the module must implement the **ReportModule_Chart_Type** function and whatever additional functions are dictated by the return value from **ReportModule_Chart_Type**.

output_tabular – (Boolean) If true, the module supports tabular (CSV or XLS) output and must implement the **ReportModule_Tabular_Definition** function.

output_custom – (Boolean) True if the module supports a non-tabular or other custom output format (generally some sort of data export). If true, the module must populate the **output_custom_name** member and implement the **ReportModule_Export** function.

output_custom_name – (Text) If **output_custom** is true, this value should be populated with text describing the output format that will be displayed to the user.

provision_settings – (Boolean) True if the module supports provisioning of report settings (through the **Report_Add** provisioning tag). If true, the module must implement the **ReportModule_Provision_Settings** function.

Return Value

None. The module must not return a value.

ReportModule_Chart_Type

This function is only required if the module implements the **output_chart** capability. It is called to determine the type of chart that the report module will output.

Supported API Version

5.70 and higher

Syntax

ReportModule_Chart_Type(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being run

Return Value

One of the following values:

"html" Indicates that the module will output its own HTML formatted chart. When this value is returned, the module must implement the **ReportModule_HTML_Chart** function and it will be called to do the actual chart display.

"svg_line" Uses the report subsystem to render an SVG line chart. When this value is returned, the module must implement the **ReportModule_SVG_Line_Chart_Definition** and **ReportModule_Format_Vertical_Label** functions.

ReportModule_Delete

This function notifies the module that a report is being deleted. When a report is deleted, the database layer calls this function in order to give the module an opportunity to delete any records associated with the report.

Supported API Version

9.06 and higher

Syntax

ReportModule_Delete(module var, report var)

Parameters

module The **Module** record of the current module

report The **Report** record being deleted

Return Value

1 on success

0 on error

ReportModule_Display

This function is called only when the module implements the **display** capability and the **Report** record in question has been configured to be displayed on the Main page. It is called to output a Main-page display of the chart data. Typically, this is a summary of the full report data. For example, the standard report modules in PR8 generate summary displays using sparklines or smaller displays of a subset of the data. The module can output any HTML that is normally valid in the administrative interface from this function.

Supported API Version

5.70 and higher

Syntax

ReportModule_Display(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being run

Return Value

Ignored

ReportModule_Export

This function is called to output a custom export format when the module implements the **output_custom** capability. The module is responsible for all output (including a starting **<HTML>** tag if the output format is HTML) and can control the output content-type or other response headers using **miva_output_header**.

Supported API Version

5.70 and higher

Syntax

ReportModule_Export(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being exported

Return Value

1 on success

0 on error

ReportModule_Field

This function is called to render the HTML input elements for a single configuration field from the list returned by **ReportModule_Fields**.

Supported API Version

5.70 and higher

Syntax

ReportModule_Field(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The **field_id** of the field to render. This value comes from the list of **field_ids** returned by **ReportModule_Fields**

Return Value

Ignored

ReportModule_Fields

Returns a comma separated list of identifiers, one for each configuration field that should be displayed when adding or editing a report that uses this module.

Supported API Version

5.70 and higher

Syntax

ReportModule_Fields(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being configured, or NULL if a report is being added

Return Value

A comma separated list of field identifiers

ReportModule_Format_Vertical_Label

This function is presently only required if the module implements the **output_chart** feature and specifies a chart type of "svg_line" as the return value from **ReportModule_Chart_Type**. It may also be required for new chart types in the future. This function adds whatever formatting is required for proper display of a vertical label in the output chart. For example, it could format dollar amounts using the store's currency formatting module, or add commas to non-currency numeric values.

Supported API Version

5.70 and higher

Syntax

ReportModule_Format_Vertical_Label(module var, report var, set_id, data)

Parameters

module	The **Module** record of the current module
report	The **Report** record being charted
set_id	The **set_id** identifying a group of records from the **ReportData** table for this data set
data	The value that should be formatted

Return Value

The formatted value of **data**

Note: For SVG line charts, HTML is permitted, however future chart types may not properly handle HTML.

ReportModule_HTML_Chart

When **ReportModule_Chart_Type** returns "html", this function is called to allow the module to output whatever content is required to visualize or chart the report data. The module is responsible for all output.

Supported API Version

5.70 and higher

Syntax

ReportModule_HTML_Chart(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being charted

Return Value

1 on success

0 on error

ReportModule_Invalid

When **ReportModule_Validate** returns **0**, this function is called for each **field_id** from the list returned by **ReportModule_Fields** to determine if the field's prompt should be shown in the invalid state.

Supported API Version

5.70 and higher

Syntax

ReportModule_Invalid(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The **field_id** being queried

Return Value

1 if the field is invalid

0 if the field is valid

ReportModule_Prompt

This function is called for each field in the list returned by **ReportModule_Fields** to get a prompt that is displayed to the user.

Supported API Version

5.70 and higher

Syntax

ReportModule_Prompt(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The **field_id** for which to return the prompt

Return Value

The textual prompt for the requested field. HTML is permitted.

ReportModule_Provision_Settings

This function is called only if the module implements the **provision_settings** capability. This function is called when a report is being added using the **<Report_Add>** provisioning tag to provision module-specific configuration values for the report.

Supported API Version

5.70 and higher

Syntax

ReportModule_Provision_Settings(module var, report var, provide_xml var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being provisioned
provide_xml	The contents of the **<Settings>** subtag of **<Report_Add>**, in parsed provisioning format

Return Value

1 on success

0 on error

If the module returns **0**, the provisioned report is not added

Note: Modules should report provisioning errors/warnings with **PRV_LogMessage** or **PRV_LogError**

ReportModule_Run

This function is called to execute the report when the module does not implement the **date_range** capability. When called, the module should do whatever operations are required to gather data for the report. Generally, this will involve making database queries and storing records in the **ReportData** table.

Supported API Version

5.70 and higher

Syntax

ReportModule_Run(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being executed

Return Value

1 on success

0 on error

ReportModule_Run_DateRange

This function is called when the module implements the **date_range** capability and does not implement the **date_interval** capability. When called, the module should execute whatever queries are required to generate the report data and store the data in the **ReportData** table or in a module-specific format.

Supported API Version

5.70 and higher

Syntax

ReportModule_Run_DateRange(module var, report var, time_t_start, time_t_end)

Parameters

module	The **Module** record of the current module
report	The **Report** record being executed
time_t_start	The starting date/time for the report execution in UNIX **time_t** format
time_t_end	The ending date/time in UNIX **time_t** format

Important: The ending **time_t** is not inclusive and the report should include data that is < **time_t_end**, *not* ≤ **time_t_end**.

Return Value

1 on success

0 on error

ReportModule_Run_Intervals

This function is called to run a report when the module implements both the **date_range** and **date_interval** capabilities.

Supported API Version

5.70 and higher

Syntax

ReportModule_Run_Intervals(module var, report var, time_t_start, time_t_end, intervals var, interval_count)

Parameters

module	The **Module** record of the current module
report	The **Report** record being executed
time_t_start	The starting UNIX **time_t** of the overall report execution (i.e., the **time_t_start** of the first interval)
time_t_end	The ending UNIX **time_t** of the overall report execution (i.e., the **time_t_end** of the last interval)
intervals	An array containing one entry per date interval between **time_t_start** and **time_t_end**. Each entry has the following members: **time_t_start** – The starting **time_t** of this interval **time_t_end** – The ending **time_t** of this interval
interval_count	The number of entries in the **intervals** array

Important: The ending **time_t** is not inclusive and the report should include data that is < **time_t_end**, *not* ≤ **time_t_end**.

Return Value

1 on success

0 on error

ReportModule_SVG_Line_Chart_Definition

When the module implements the **output_chart** capability, and **ReportModule_Chart_Type** returns "svg_line", this function is called by the report subsystem to determine what data sets should be included in the SVG line chart and other controls over the chart display.

Supported API Version

5.70 and higher

Syntax

ReportModule_SVG_Line_Chart_Definition(module var, report var, chart var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being charted

chart An output structure that controls the resulting SVG line chart. The module should populate the following members:

dataset_count – The number of entries in the **datasets** array

datasets[] – An array of structures defining the individual lines in the chart. Each entry should have the following members:

set_id – A reference to a **set_id** from a set of **ReportData** table entries that comprise the individual data points for this data set

color – The color in which to draw this data set, in #RRGGBB format. A pre-defined color palette is available through the **rpt_ut.mv** function **Chart_Color_Palette**.

name – The name of this data set as it should be displayed to the user

visible – (Boolean) True if the data set should be drawn by default. If false, the data set will be present but its checkbox will be unchecked and its line will not be plotted.

Return Value

None. The function must not return a value.

ReportModule_Tabular_Definition

This function is called when a report using a module that implements the **output_tabular** capability is exported in either CSV or XLS format. The function fills out the **definition** parameter to define the rows and columns that are then exported into the requested tabular format by the report subsystem.

Supported API Version

5.70 and higher

Syntax

ReportModule_Tabular_Definition(module var, report var, definition var)

Parameters

module The **Module** record of the current module

report The **Report** record being exported

definition Output. A structure with the following members:

rows[] – An array of structures, with the members of each entry in the array defining a single row. The members are as follows:

start – The beginning row at which the data defined by this entry is drawn

type – One of the following: “values,” “reportdata” or “reportdata_date”. Depending on the type specified, differing additional fields are required (see below).

columns[] – An array of structures, with the members of each entry in the array defining a single column. The members are as follows:

start – The beginning column at which the data defined by this entry is drawn

type – One of the following: "values," "reportdata" or "reportdata_date". Depending on the type specified, differing additional fields are required (see below).

For **type** "values", the following additional field is required:

- **values[]** – An array of literal values that are included in the tabular output beginning at the position specified by "start", and extending horizontally (for a row) or vertically (for a column) for each entry in the **values** array.

For **type** "reportdata", the following additional field is required:

- **set_id** – A **set_id** defining a set of data points from the **ReportData** table. Each data point is included in the tabular output beginning at the position specified by "start", and extending horizontally (for a row) or vertically (for a column) for each entry in the **values** array.

For **type** "reportdata_date", the following additional field is required:

- **set_id** – A **set_id** defining a set of data points from the **ReportData** table. The **dt_start** member of the **ReportData** record for each data point is formatted using the output date/time format extending horizontally (for a row) or vertically (for a column) for each entry in the **values** array.

Rows are output first in array element order, then columns are output in array element order. Overlapping is allowed, and the output file will include the last generated value for each cell.

Return Value

None. This function must not return a value.

ReportModule_Update

This function is called to store the configuration of a report when the report is added or updated in the administrative interface.

Supported API Version

5.70 and higher

Syntax

ReportModule_Update(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being added or updated. The module should store its configuration in the structure **report:config**.

Return Value

1 on success

0 on error

ReportModule_Validate

This function is called to validate the HTML input controls for the fields defined by **ReportModule_Fields** when a report is being added or updated.

Supported API Version

5.70 and higher

Syntax

ReportModule_Validate(module var, report var)

Parameters

module	The **Module** record of the current module
report	The **Report** record being updated or NULL if a report is being added

Return Value

1 if all input fields are valid

0 if any input field is invalid

Note: Modules can report invalid fields through **ReportModule_Invalid** or by calling the **FieldError** function.

Scheduled Task Feature (scheduledtask)

The **scheduledtask** subsystem provides a mechanism for modules to execute tasks on a scheduled basis.

The **scheduledtask** feature includes the following functions:

- **ScheduledTaskModule_Capabilities**
- **ScheduledTaskModule_Delete**
- **ScheduledTaskModule_Execute**
- **ScheduledTaskModule_Field**
- **ScheduledTaskModule_Fields**
- **ScheduledTaskModule_Invalid**

- **ScheduledTaskModule_Operations**
- **ScheduledTaskModule_Prompt**
- **ScheduledTaskModule_Provision_Settings**
- **ScheduledTaskModule_Update**
- **ScheduledTaskModule_Validate**

ScheduledTaskModule_Capabilities

This function describes the capabilities that the scheduled task module provides to the **scheduledtask** subsystem.

Syntax

ScheduledTaskModule_Capabilities(module var, capabilities var)

Parameters

module The **Module** record of the current module

capabilities An output structure containing information about the functionality of the scheduled task module API that the module implements. The output structure is populated with the following members:

:provision – (Boolean) If true, the module supports creation of scheduled tasks through provisioning and does not require any additional settings
:provision_settings – (Boolean) If true, the module supports configuration of its settings through provisioning. The module must implement the **ScheduledTaskModule_Provision_Settings** function.

Return Value

Ignored

ScheduledTaskModule_Delete

This function notifies the module that a scheduled task is being deleted. When a scheduled task is deleted, the database layer calls this function in order to give the module an opportunity to delete any records associated with the scheduled task.

Syntax

ScheduledTaskModule_Delete(module var, task var)

Parameters

module The **Module** record of the current module

task The scheduled task record being deleted

Return Value

1 on success

0 on errror

ScheduledTaskModule_Execute

This function is called when a module specific operation needs to be run. The function should check for the **task:operation** value, which should be one of the codes defined in **ScheduledTaskModule_Operations**.

Syntax

ScheduledTaskModule_Execute(module var, task var)

Parameters

module	The **Module** record of the current module
task	The scheduled task record being executed

Return Value

1 on success

0 on errror

ScheduledTaskModule_Field

This function draws the HTML input element(s) required for configuration of a single field from the list returned by **ScheduledTaskModule_Fields**. The module outputs the required HTML directly.

Syntax

ScheduledTaskModule_Field(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **ScheduledTaskModule_Fields**.

Return Value

Ignored

ScheduledTaskModule_Fields

This function returns a comma separated list of field identifiers for module-specific fields that should be present for a given scheduled task.

Syntax

ScheduledTaskModule_Fields(module var, task var)

Parameters

module	The **Module** record of the current module
task	The scheduled task record being executed

Return Value

A comma separated list of field identifiers

ScheduledTaskModule_Invalid

When **ScheduledTaskModule_Validate** returns **0**, this function is called for each field to determine which field(s) should be displayed in the invalid state.

Syntax

ScheduledTaskModule_Invalid(module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	The ID of the field to output. This ID comes from the comma separated list returned by **ScheduledTaskModule_Fields**.

Return Value

1 if the specified field is invalid

0 if the specified field is valid

ScheduledTaskModule_Operations

This function populates the supported operations of the scheduled task module.

Syntax

ScheduledTaskModule_Operations(module var, operations var)

Parameters

module The **Module** record of the current module

operations An array of elements stating the supported operations. For example:

<MvASSIGN NAME = "l.operations" INDEX = 1 MEMBER = "code" VALUE = "some_unique_code">

<MvASSIGN NAME = "l.operations" INDEX = 1 MEMBER = "descrip" VALUE = "A brief description displayed by the scheduled task subsystem to the end-user">

<MvASSIGN NAME = "l.operations" INDEX = 2 MEMBER = "code" VALUE = "another_operation">

<MvASSIGN NAME = "l.operations" INDEX = 2 MEMBER = "descrip" VALUE = "Another brief description">

Return Value

The number of operations array elements

ScheduledTaskModule_Prompt

This function is called for each field returned by **ScheduledTaskModule_Fields** to obtain the prompt (the descriptive text displayed to the left of the field).

Syntax

ScheduledTaskModule_Prompt(module var, field_id)

Parameters

module The **Module** record of the current module

field_id The ID of the field to output. This ID comes from the comma separated list returned by **ScheduledTaskModule_Fields**.

Return Value

Textual prompt. HTML permitted.

ScheduledTaskModule_Provision_Settings

This function is called when a scheduled task is created using the **ScheduledTask_Add/Update** provisioning tag and the **<Settings>** tag is present. The module is expected to implement provisioning for whatever settings are normally maintained on a scheduled task by scheduled task basis.

Syntax

ScheduledTaskModule_Provision_Settings(module var, task var, proxide_xml var)

Parameters

module	The **Module** record of the current module
task	The scheduled task record being executed, or NULL if a scheduled task is being added
provide_xml	Provisioning-parsed XML from the tag of a **ScheduledTask_Add**/**Update** tag

Return Value

1 on success

0 on errror

ScheduledTaskModule_Update

This function is called when a scheduled task is modified so that the module can store any changes to the module controlled scheduled task fields.

Syntax

ScheduledTaskModule_Update(module var, task var)

Parameters

module	The **Module** record of the current module
task	The scheduled task record being edited, or NULL if a scheduled task is being added

Return Value

1 on success

0 on errror

ScheduledTaskModule_Validate

This function is called to validate module-provided scheduled task fields when creating or updating a scheduled task record from the administrative interface.

Syntax

ScheduledTaskModule_Validate(module var, task var)

Parameters

module	The **Module** record of the current module
task	The scheduled task record being validated, or NULL if a scheduled task is being added

Return Value

1 on success

0 on errror

Shipping Calculation Feature (shipping)

Modules that implement the **Shipping Calculation** feature (**shipping**) provide one or more shipping methods in the shopping interface (e.g., USPS, FedEx, UPS, etc.)

The **shipping** feature includes the following functions:

- **ShippingModule_Basket_Methods**
- **ShippingModule_Calculate_Basket**
- **ShippingModule_Description**
- **ShippingModule_Enabled_Methods**
- **ShippingModule_Order_Content**
- **ShippingModule_Order_Delete**
- **ShippingModule_Order_Head**
- **ShippingModule_Order_Tabs**
- **ShippingModule_Order_Update**
- **ShippingModule_Order_Validate**
- **ShippingModule_Report_Fields**
- **ShippingModule_Report_Label**
- **ShippingModule_Report_Value**
- **ShippingModule_Shipping_Methods** (deprecated)

ShippingModule_Basket_Methods

This function replaces **ShippingModule_Shipping_Methods** in version 5.71 API or higher modules. Its purpose is to generate a list of shipping methods, complete with rate information, that will be presented to the shopper at checkout. The current shopping basket is always stored in the global variable **g.Basket** (as with **ShippingModule_Shipping_Methods**).

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Basket_Methods(module var, packages var, package_count, methods var)

Parameters

module	The **Module** record of the current module.
packages	An array of packages with one entry for each box required to ship the basket in **g.Basket**. Each entry in the array has the following members: **weight –** The weight of the package in the store's configured weight units

width – The width of the package in the store's configured dimension units

length – The length of the package in the store's configured dimension units

height – The height of the package in the store's configured dimension units

items – An array containing records that describe the contents of this package. Each entry has the following members:

product – A product record (present only if the item maps to a product)

width – The width of the item in the store's configured dimension units

length – The length of the item in the store's configured dimension units

height – The height of the item in the store's configured dimension units

weight – The weight of this particular item in the store's configured weight units

item_count – The number of entries in the **items** array

product_ids – An array of the unique numeric IDs of products in this package

product_count – The number of entries in the **product_ids** array

package_count The number of package records in the **packages** array parameter

methods An output array that the module populates to describe the shipping methods that are valid for the basket. Each entry in the array has the following members:

code – A code that the module uses to identify this shipping method. Can be anything but for historical reasons should not include a colon (:)

name – A description of the shipping method that will be displayed to the shopper

price – The price of this shipping method

Return Value

The number of shipping methods populated into the methods array

– OR –

0 on error

Note: Depending on the caller, error messages are usually not reported.

ShippingModule_Calculate_Basket

This function is called to apply a shipping method provided by the module to the current basket. The module should recalculate or retrieve the previously calculated shipping rate for the indicated method from the **BasketInfo** mechanism, then create the appropriate **BasketCharge** database table entries to reflect the calculated shipping rate.

Syntax

ShippingModule_Calculate_Basket (module var, data)

Parameters

module The **Module** record of the current module

data The code of the selected shipping method. This code will match one of the values returned in the **code** member of the output array from **ShippingModule_Shipping_-Methods** or **ShippingModule_Basket_Methods**.

Return Value

1 on success

0 on error

ShippingModule_Description

This function is called to retrieve a user-friendly description of a shipping method.

Syntax

ShippingModule_Description (module var, data)

Parameters

module The **Module** record of the current module

data The code of the shipping method for which a description is being queried. The code matches one of the values in the array returned by **ShippingModule_Basket_Methods** (new API) or **ShippingModule_Shipping_Methods** (old API).

Return Value

A textual description of the shipping method or an empty string if the shipping method code is not recognized

ShippingModule_Enabled_Methods

This function loads a list of enabled shipping methods that the module can return regardless of basket contents.

Supported API Version

5.61 and higher

Syntax

ShippingModule_Enabled_Methods (module var, methods var)

Parameters

module The **Module** record of the current module

methods An output array that the module populates to describe the available shipping methods. Each entry in the array has the following members:

code – A code that the module uses to identify this shipping method. Can be anything but for historical reasons should not include a colon (:)

name – A description of the shipping method that will be displayed to the shopper

Return Value

The number of shipping methods populated into the methods array

– OR –

0 on error

ShippingModule_Order_Content

This function is called to display the content of an Order tab from the list returned by **ShippingModule_Order_Tabs**. When the tab parameter matches one of the tabs drawn by this module, it draws the controls (using HTML) meant to be visible on the tab. When the tab parameter does not match, the module hides any input values in HTML hidden **<input>** tags.

Syntax

ShippingModule_Order_Content (module var, tab, load_fields)

Parameters

module The **Module** record of the current module

tab The code of the currently visible tab. It can be one of the module's tabs or one of the system generated tabs.

load_fields This value is set to **1** when the page is initialized so the module can load default values for any input fields. On subsequent calls, the value is **0** and the module is expected to retain the initially loaded values using hidden input fields, etc.

Return Value

1 on success

0 on error

ShippingModule_Order_Delete

This function is called when an order is deleted for the shipping module associated with that order (selected during the checkout process), giving the shipping module a chance to delete any records it may have stored related to that order.

Note: Because the order being deleted is not passed to the module as a parameter, the order must be inferred by examining global variables. Refer to *Appendix D: Deleting Orders* on page 383 for further details.

Syntax

ShippingModule_Order_Delete (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

ShippingModule_Order_Head

This function allows the module to output content in the HTML **<head>** tag of the Legacy Order Processing **Edit Order** screen and module-specific order tabs in the new Order Management interface. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

ShippingModule_Order_Head(module var, tab, order var)

Parameters

module The **Module** record of the current module

tab The code of the currently visible tab

order The **Order** record of the order being displayed

Return Value

1 on success

0 on error

ShippingModule_Order_Tabs

This function is called for the shipping module associated with an order to determine what (if any) additional order tabs should be visible when reviewing the order. **ShippingModule_Order_Tabs**

returns a string that identifies the tabs to be added to the default list of system-generated tabs. The string takes the following form:

```
<code>:<Description>,<code2>:<Description2>
```

The module may specify as many `<code>:<Description>` pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Syntax

ShippingModule_Order_Tabs (module var)

Parameters

module The **Module** record of the current module

Return Value

A string describing the tabs to be added (see description above)

ShippingModule_Order_Update

Admin calls this function when **Update** is selected on the **Edit Order** configuration screen.

Syntax

ShippingModule_Order_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

ShippingModule_Order_Validate

Admin calls this function to validate fields when a user selects **Update** on the **Edit Order** configuration screen.

Syntax

ShippingModule_Order_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if all fields are valid

0 if any field is invalid

ShippingModule_Report_Fields

Admin calls this function when displaying a standard batch report. It returns a comma separated list of field identifiers for fields displayed in the batch report.

Syntax

ShippingModule_Report_Fields (module var)

Parameters

module The **Module** record of the current module

Return Value

A comma separated list of field identifiers

ShippingModule_Report_Label

Admin calls this function when displaying a standard batch report. It returns a string for use as visible text for display beside the field identified by **field_id**.

Syntax

ShippingModule_Report_Label (module var, field_id)

Parameters

module The **Module** record of the current module

field_id A field identifier from the comma separated list of fields returned by **ShippingModule_Report_Fields**

Return Value

A textual description of the field specified by **field_id**

ShippingModule_Report_Value

Admin calls this function when displaying a standard batch report. It returns the value to display in the field identified by **field_id** for the method identified in **data**.

Syntax

ShippingModule_Report_Value (module var, field_id, data)

Parameters

module The **Module** record of the current module

field_id A field identifier from the comma separated list of fields returned by **ShippingModule_Report_Fields**

data The shipping method code for which report field values are being queried. This will be one of the shipping method codes from the output array returned by **ShippingModule_Basket_Methods** or **ShippingModule_Shipping_Methods**

Return Value

The textual value of the report field identified by **field_id** for the indicated shipping method

ShippingModule_Shipping_Methods

This function generates a list of shipping methods, complete with rate information, that will be presented to the shopper at checkout. The current shopping basket is always stored in the global variable **g.Basket** (as with **ShippingModule_Basket_Methods**).

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Basket_Methods**.

Supported API Version

Supported in versions less than 5.71

Syntax

ShippingModule_Shipping_Methods (module var, methods var)

Parameters

module The **Module** record of the current module

methods An output array that the module populates to describe the shipping methods that are valid for the basket. Each entry in the array has the following members:

code – A code that the module uses to identify this shipping method. Can be anything but for historical reasons should not include a colon (:)

name – A description of the shipping method that will be displayed to the shopper

price – The price of this shipping method

Return Value

The number of shipping methods populated into the methods array

– OR –

0 on error

Shipping Label Generation Feature (shipping_label)

Modules that implement the **Shipping Label Generation** feature (**shipping_label**) provide user interface elements and operational code to generate shipping labels.

The **shipping_label** feature includes the following functions:

- **ShippingModule_Label_Boxes**
- **ShippingModule_Label_Field** (deprecated)
- **ShippingModule_Label_Fields** (deprecated)
- **ShippingModule_Label_Generate** (deprecated)
- **ShippingModule_Label_Invalid** (deprecated)
- **ShippingModule_Label_Methods**
- **ShippingModule_Label_Package_Field**
- **ShippingModule_Label_Package_Fields**
- **ShippingModule_Label_Package_Invalid**
- **ShippingModule_Label_Package_Prompt**
- **ShippingModule_Label_Package_Validate** (deprecated)
- **ShippingModule_Label_Package_Validate_Package**
- **ShippingModule_Label_Prompt** (deprecated)
- **ShippingModule_Label_Render**
- **ShippingModule_Label_Shipment_Field**
- **ShippingModule_Label_Shipment_Fields**
- **ShippingModule_Label_Shipment_Invalid**
- **ShippingModule_Label_Shipment_Prompt**
- **ShippingModule_Label_Shipment_Validate**
- **ShippingModule_Label_Validate** (deprecated)
- **ShippingModule_Label_Void_Shipment**
- **ShippingModule_Labels_Generate**

ShippingModule_Label_Boxes

This function allows a module to return a list of shipper-specific boxes that can be used when generating labels. These boxes are displayed alongside the boxes configured by the merchant in the label generation dialog.

Examples of shipper-specific boxes:

- USPS® Large Flat Rate Box
- FedEx® Express Envelope

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Boxes(module var, ordershipment var, boxes var)

Parameters

module	The **Module** record of the current module
ordershipment	The **OrderShipment** record for which label(s) are being generated
boxes	An output array describing the boxes supported by the module. Each element has the following members: **code** – A unique code that identifies the box **name** – A descriptive name of the box that will be displayed to the user

Return Value

The number of boxes placed into the **boxes** array

ShippingModule_Label_Field

This function draws a single input field for label generation. HTML is allowed.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Label_Shipment_Field**.

Note: In Miva Merchant PR7 through PR8 Update 3, there is a single package per shipment. Only a single set of the fields returned by **ShippingModule_Label_Fields** and drawn by the other **ShippingModule_Label_*XXX*** fields will be displayed at once. In PR8 Update 4 and newer, the fields defined by these deprecated functions will be drawn once per package within a shipment. The module will be called multiple times to generate multiple labels for the shipment when there is more than one package.

Supported API Version

Supported in versions less than 5.70 and greater than or equal to 5.60

Syntax

ShippingModule_Label_Field (module var, method, field_id)

Parameters

module	The **Module** record of the current module
method	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried (from the list of field IDs returned by **ShippingModule_Label_Fields**)

Return Value

Ignored

ShippingModule_Label_Fields

This function returns a comma separated list of field identifiers that will be displayed for the label.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Label_Shipment_Fields**.

Note: In Miva Merchant PR7 through PR8 Update 3, there is a single package per shipment. Only a single set of the fields returned by **ShippingModule_Label_Fields** and drawn by the other **ShippingModule_Label_*XXX*** fields will be displayed at once. In PR8 Update 4 and newer, the fields defined by these deprecated functions will be drawn once per package within a shipment. The module will be called multiple times to generate multiple labels for the shipment when there is more than one package.

Supported API Version

Supported in versions less than 5.71

Syntax

ShippingModule_Label_Fields (module var, method)

Parameters

module	The **Module** record of the current module
method	The code of the selected shipping method (from **ShippingModule_Label_Methods**)

Return Value

A comma separated list of field identifiers

ShippingModule_Label_Generate

This function is called to generate a single shipping label. If a shipment contains multiple packages in PR8 Update 4 or newer, this function will be called once per package.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Labels_Generate**.

Note: In Miva Merchant PR7 through PR8 Update 3, there is a single package per shipment. Only a single set of the fields returned by **ShippingModule_Label_Fields** and drawn by the other **ShippingModule_Label_*XXX*** fields will be displayed at once. In PR8 Update 4 and newer, the fields defined by these deprecated functions will be drawn once per package within a shipment. The module will be called multiple times to generate multiple labels for the shipment when there is more than one package.

Supported API Version

Supported in versions less than 5.71

Syntax

ShippingModule_Label_Generate (module var, method, source_address var, dest_address var, weight, ordershipment var)

Parameters

module	The **Module** record of the current module
method	The code of the selected shipping method
source_address	A structure containing the "From" or origin address of the shipment. Contains the following members: **name** – Shipper name **email** – Shipper email address **phone** – Shipper phone number **fax** – Shipper FAX number **company** – Company name **addr1** – Street address, line 1 **addr2** – Street address, line 2 **city** – City **state** – State

zip – Zip/postal code

cntry – ISO 3166-1 alpha-2 country code (two-letter country code)

dest_address A structure containing the "To" or destination address of the shipment. Contains the following members:

name – Recipient name

email – Recipient email address

phone – Recipient phone number

fax – Recipient FAX number

company – Company name

addr1 – Street address, line 1

addr2 – Street address, line 2

city – City

state – State

zip – Zip/postal code

cntry – ISO 3166-1 alpha-2 country code (two-letter country code)

weight The weight of the package being shipped (in store weight units)

ordershipment The **OrderShipment** record of the shipment for which the label is being generated

After generating the label, the module should populate the following fields in the passed-in **ordershipment** parameter:

:tracknum Receives the tracking number for the generated label (if applicable)

:tracktype Receives the appropriate tracking link code for the tracking number

:cost Receives the actual shipping cost (if available) after the label was generated

:label_type Receives the MIME type of the label content (for example, application/pdf)

:label_data Receives the contents of the label itself, Base64 encoded

Return Value

1 on success

0 on error

ShippingModule_Label_Invalid

If **ShippingModule_Label_Validate** returns **0**, **ShippingModule_Label_Invalid** is called for each field identifier to determine if a particular field failed validation. If **ShippingModule_Label_Invalid** returns **1** for a particular field, that field's prompt is displayed in red in the user interface.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Label_Shipment_Invalid**.

Note: In Miva Merchant PR7 through PR8 Update 3, there is a single package per shipment. Only a single set of the fields returned by **ShippingModule_Label_Fields** and drawn by the other **ShippingModule_Label_*XXX*** fields will be displayed at once. In PR8 Update 4 and newer, the fields defined by these deprecated functions will be drawn once per package within a shipment. The module will be called multiple times to generate multiple labels for the shipment when there is more than one package.

Supported API Version

Supported in versions less than 5.71

Syntax

ShippingModule_Label_Invalid (module var, method, field_id)

Parameters

module	The **Module** record of the current module
method	The code of the selected shipping method
field_id	The ID of the field being queried

Return Value

1 if the field is invalid

0 if the field is valid

ShippingModule_Label_Methods

This function returns a list of shipping methods that the module supports for label generation.

Note: The existing **ShippingModule_Label_Methods** function was modified to add a new **ordershipment** parameter in API version 5.71.

Supported API Version

New **ordershipment** parameter in 5.71 (PR8 Update 4)

Important: If the API version is 5.71 or higher, the implementation of the function must include the **ordershipment** parameter. If the API version is 5.70 or lower, the implementation of the function must *not* include the **ordershipment** parameter.

Syntax

ShippingModule_Label_Methods(module var, methods var, ordershipment var)

Parameters

module	The **Module** record of the current module
methods	An output array that describes the shipping methods for which labels can be generated. Each element in the array contains the following members: **code** – A unique code describing the shipping method **name** – The name of the method as it should be displayed to the user
ordershipment	The **OrderShipment** record for which label(s) are being generated

Return Value

The number of shipping methods placed into the **methods** array.

ShippingModule_Label_Package_Field

This function draws a single package-level input field.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Package_Field(module var, method_code, field_id, field_prefix, fields var, product_ids var, product_count)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried
field_prefix	The package-specific field prefix for this particular package. When the module constructs HTML input elements for a package field, it should include the field prefix in the **NAME** attribute of the HTML element. ***Example***: `<input type="text" name="{ l.field_prefix $ 'example_field' }" value="{ encodeentities( l.fields:example_field ) }">`
fields	A structure containing the fields for this package. There will be one member for each HTML input element. Using the example above, the **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.
product_ids	An array containing the IDs of the products within the package. Each unique ID is included only once, even if there are more than one of that particular product included.
product_count	The number of entries in the **product_ids** array.

Return Value

Ignored

ShippingModule_Label_Package_Fields

This function returns a comma separated list of field identifiers that will be displayed for each package.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Package_Fields(module var, method_code, field_prefix, fields var, ordershipment var, product_ids var, product_count)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_prefix	The package-specific field prefix for this particular package. When the module constructs HTML input elements for a package field, it should include the field prefix in the **NAME** attribute of the HTML element. ***Example***: `<input type="text" name="{ l.field_prefix $ 'example_field' }" value="{ encodeentities( l.fields:example_field ) }">`
fields	A structure containing the fields for this package. There will be one member for each HTML input element. Using the example above, the **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.
ordershipment	The **OrderShipment** record for which label(s) are being generated
product_ids	An array containing the IDs of the products within the package. Each unique ID is included only once, even if there are more than one of that particular product included.
product_count	The number of entries in the **product_ids** array.

Return Value

A comma separated list of field identifiers that is displayed for each package. This list is broken into individual values, with each value passed as the **field_id** parameter to the other **ShippingModule_Label_Package_*XXX*** functions.

ShippingModule_Label_Package_Invalid

This function indicates whether a package-level field failed validation and should be displayed in the invalid state.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Package_Invalid(module var, method_code, field_id, field_prefix, fields var, product_ids var, product_count)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried
field_prefix	The package-specific field prefix for this particular package. When the module constructs HTML input elements for a package field, it should include the field prefix in the **NAME** attribute of the HTML element. ***Example***: `<input type="text" name="{ l.field_prefix $ 'example_field' }" value="{ encodeentities( l.fields:example_field ) }">`
fields	A structure containing the fields for this package. There will be one member for each HTML input element. Using the example above, the **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.
product_ids	An array containing the IDs of the products within the package. Each unique ID is included only once, even if there are more than one of that particular product included.
product_count	The number of entries in the **product_ids** array.

Return Value

1 if the field identified by **field_id** is invalid

0 if the field identified by **field_id** is valid

ShippingModule_Label_Package_Prompt

This function returns the prompt (descriptive text displayed to the left of a field) for the field identified by **field_id**.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Package_Prompt(module var, method_code, field_id, field_prefix, fields var, product_ids var, product_count)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried
field_prefix	The package-specific field prefix for this particular package. When the module constructs HTML input elements for a package field, it should include the field prefix in the **NAME** attribute of the HTML element. ***Example***: `<input type="text" name="{ l.field_prefix $ 'example_field' }" value="{ encodeentities( l.fields:example_field ) }">`
fields	A structure containing the fields for this package. There will be one member for each HTML input element. Using the example above, the **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.
product_ids	An array containing the IDs of the products within the package. Each unique ID is included only once, even if there are more than one of that particular product included.
product_count	The number of entries in the **product_ids** array.

Return Value

Textual prompt. HTML is permitted.

ShippingModule_Label_Package_Validate

This function is called for each package for which labels will be generated. It allows the module to validate that the input fields were filled out correctly.

Note: This function was replaced with **ShippingModule_Label_Package_Validate_Package** in 5.72 API modules.

Supported API Version

5.71 (PR8 Update 4), deprecated in 5.72 (PR8 Update 5)

Syntax

ShippingModule_Label_Package_Validate(module var, method_code, field_prefix, fields var, product_ids var, product_count)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)

field_prefix	The package-specific field prefix for this particular package. When the module constructs HTML input elements for a package field, it should include the field prefix in the **NAME** attribute of the HTML element. ***Example***: `<input type="text" name="{ l.field_prefix $ 'example_field' }" value="{ encodeentities( l.fields:example_field ) }">`
fields	A structure containing the fields for this package. There will be one member for each HTML input element. Using the example above, the **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.
product_ids	An array containing the IDs of the products within the package. Each unique ID is included only once, even if there are more than one of that particular product included.
product_count	The number of entries in the **product_ids** array.

Return Value

1 if all fields are valid

0 if any field is invalid

Note: Modules can report validation errors either through **ShippingModule_Label_Package_Invalid** or by calling the **FieldError** function.

ShippingModule_Label_Package_Validate_Package

This function is called for each package for which labels will be generated. It allows the module to validate that the input fields were filled out correctly.

Note: For 5.72 API modules, this function replaces **ShippingModule_Label_Package_Validate**. **ShippingModule_Label_Package_Validate_Package** is functionally identical to **ShippingModule_Label_Package_Validate** but the **ordershipment** record and the entire **package** record are passed in to provide the module access to additional fields that may be required for validation.

Supported API Version

5.72 and higher (new in PR8 Update 5)

Syntax

ShippingModule_Label_Package_Validate_Package (module var, method_code, ordershipment var, package var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)

ordershipment The ordershipment record for which labels are being generated

package A structure containing details about the package being validated. It contains the following members:

weight – The weight (in store weight units) of the package

modulebox_code – If a module-specified box (from **ShippingModule_Label_Boxes**) was selected for this package, this member will be present with one of the codes from that function and width/length/height will be empty

– OR –

width – Width in store dimension units

length – Length in store dimension units

height – Height in store dimension units

product_ids – An array of unique **product_ids** contained in this package

product_count – The number of elements in **product_ids**

fields – A structure containing the module's package-level fields for this package. There will be one member for each HTML input element.

Example:
```
<input type="text" name="{ l.field_prefix $
'example_field' }" value="{ encodeentities
( l.fields:example_field ) }">
```

The **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.

Return Value

1 if all fields are valid

0 if any field is invalid

Note: Modules can report validation errors either through **ShippingModule_Label_Package_Invalid** or by calling the **FieldError** function.

ShippingModule_Label_Prompt

This function returns the prompt for a single label generation field.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Label_Shipment_Prompt**.

Note: In Miva Merchant PR7 through PR8 Update 3, there is a single package per shipment. Only a single set of the fields returned by **ShippingModule_Label_Fields** and drawn by the other **ShippingModule_Label_*XXX*** fields will be displayed at once. In PR8 Update 4 and newer, the fields defined by these deprecated functions will be drawn once per package within a shipment. The module will be called multiple times to generate multiple labels for the shipment when there is more than one package.

Supported API Version

Supported in versions less than 5.71

Syntax

ShippingModule_Label_Prompt (module var, method, field_id)

Parameters

module	The **Module** record of the current module
method	The code of the selected shipping method
field_id	The ID of the field being queried

Return Value

A prompt for the field. HTML is allowed. The user interface displays the prompts to the left of the field.

ShippingModule_Label_Render

If an OrderShipmentLabel record is created with a "label_type" of "module", this function will be called (in the module identified by the OrderShipmentLabel's **module_id** field) to draw the shipping label.

This allows a module to handle complex or compound label displays that cannot be displayed simply by setting the Content-Type header. For example, the UPS Ready® Tools module labels consist of two parts—HTML and GIF—and the module uses this mechanism to enable the display of both parts.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Render(module var, label var)

Parameters

module	The **Module** record of the current module
label	The **OrderShipmentLabel** record being displayed

Return Value

Ignored

ShippingModule_Label_Shipment_Field

The **ShippingModule_Label_Shipment_*XXX*** functions in API version 5.71 or higher replace the **ShippingModule_Label_*XXX*** functions (**ShippingModule_Label_Field, ShippingModule_Label_Fields, ShippingModule_Label_Generate, ShippingModule_Label_Invalid, ShippingModule_Label_Prompt, ShippingModule_Label_Validate**) from older API versions and allow the module to control fields that are displayed at the shipment level when generating labels.

This function outputs the HTML required to draw a single shipment-level field.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Shipment_Field(module var, method_code, field_id, ordershipment var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried
ordershipment	The **OrderShipment** record for which label(s) are being generated

Return Value

Ignored

ShippingModule_Label_Shipment_Fields

This function returns a list of shipment-level field identifiers.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Shipment_Fields(module var, method_code, ordershipment var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
ordershipment	The **OrderShipment** record for which label(s) are being generated

Return Value

A comma separated list of field identifiers. The list is split into individual components and the individual values are passed as the **field_id** parameter to the other **ShippingModule_Label_Shipment_*XXX*** functions (**ShippingModule_Label_Shipment_Field, ShippingModule_Label_Shipment_Fields, ShippingModule_Label_Shipment_Invalid, ShippingModule_Label_Shipment_Prompt, ShippingModule_Label_Shipment_Validate**).

ShippingModule_Label_Shipment_Invalid

This function is called when **ShippingModule_Label_Shipment_Validate** returns **0** for each shipment-level field to determine whether the field's prompt should be displayed in the invalid state.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Shipment_Invalid(module var, method_code, field_id, ordershipment var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried
ordershipment	The **OrderShipment** record for which label(s) are being generated

Return Value

1 if the field specified by **field_id** is invalid

0 if the field specified by **field_id** is valid

ShippingModule_Label_Shipment_Prompt

This function returns the prompt for a single shipment-level input field.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Shipment_Prompt(module var, method_code, field_id, ordershipment var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
field_id	The ID of the field being queried
ordershipment	The **OrderShipment** record for which label(s) are being generated

Return Value

A prompt for the field. HTML is allowed. The user interface displays the prompts to the left of the field.

ShippingModule_Label_Shipment_Validate

This function is called to validate the module's shipment-level fields. Separate calls to **ShippingModule_Label_Package_Validate** are made for each package in the shipment to validate the packages.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Label_Shipment_Validate(module var, method_code, ordershipment var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method (from **ShippingModule_Label_Methods**)
ordershipment	The **OrderShipment** record for which label(s) are being generated

Return Value

1 if all shipment-level fields are valid

0 if any shipment-level field is invalid

Note: Modules can report invalid fields through **ShippingModule_Label_Shipment_Invalid** or by calling the **FieldError** function.

ShippingModule_Label_Validate

This function is called to validate the user's input prior to generating a label.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **ShippingModule_Label_Shipment_Validate**.

Note: In Miva Merchant PR7 through PR8 Update 3, there is a single package per shipment. Only a single set of the fields returned by **ShippingModule_Label_Fields** and drawn by the other **ShippingModule_Label_*XXX*** fields will be displayed at once. In PR8 Update 4 and newer, the fields defined by these deprecated functions will be drawn once per package within a shipment. The module will be called multiple times to generate multiple labels for the shipment when there is more than one package.

Supported API Version

Supported in versions less than 5.71

Syntax

ShippingModule_Label_Validate (module var, method)

Parameters

module	The **Module** record of the current module
method	The code of the selected shipping method

Return Value

1 if all fields are valid

0 if any field is invalid

ShippingModule_Label_Void_Shipment

This function facilitates the voiding of shipping labels. When called, the module should perform whatever actions are required to invalidate all labels generated for the specified shipment. It is the module's responsibility to delete any **OrderShipmentLabel** records that have been voided — if the void of one or more labels fails, the module should delete the voided labels and leave the unvoided labels alone.

Note: If a module has an API version lower than 5.72, the VOID functionality is automatically disabled in the user interface when dealing with labels from that module.

Supported API Version

5.72 and higher (new in PR8 Update 5)

Syntax

ShippingModule_Label_Void_Shipment (module var, ordershipment var)

Parameters

module	The **Module** record of the current module
ordershipment	The **OrderShipment** for which label(s) are to be voided

Return Value

1 on success

0 on error

ShippingModule_Labels_Generate

This function replaces **ShippingModule_Label_Generate** and is called to generate one or more shipping labels for an OrderShipment record. When called, the module should make whatever API calls are required to generate the label(s). The module is responsible for inserting one or more OrderShipmentLabel records containing the label data.

Supported API Version

5.71 and higher (new in PR8 Update 4)

Syntax

ShippingModule_Labels_Generate(module var, method_code, source_address var, dest_address var, ordershipment var, package_list var, package_count var)

Parameters

module	The **Module** record of the current module
method_code	The code of the selected shipping method
source_address	A structure containing the "From" or origin address of the shipment. Contains the following members: **name** – Shipper name **email** – Shipper email address **phone** – Shipper phone number **fax** – Shipper FAX number **company** – Company name **addr1** – Street address, line 1

addr2 – Street address, line 2

city – City

state – State

zip – Zip/postal code

cntry – ISO 3166-1 alpha-2 country code (two-letter country code)

dest_address A structure containing the "To" or destination address of the shipment. Contains the following members:

name – Recipient name

email – Recipient email address

phone – Recipient phone number

fax – Recipient FAX number

company – Company name

addr1 – Street address, line 1

addr2 – Street address, line 2

city – City

state – State

zip – Zip/postal code

cntry – ISO 3166-1 alpha-2 country code (two-letter country code)

ordershipment The **OrderShipment** record for which labels are being generated

package_list An array of packages within this shipment. The module is guaranteed to receive at least one package. Each element in the array contains the following members:

weight – The weight (in store weight units) of the package

modulebox_code – If a module-specified box (from **ShippingModule_Label_Boxes**) was selected for this package, this
member will be present with one of the codes from that function and width/length/height will be empty

– OR –

width – Width in store dimension units

length – Length in store dimension units

height – Height in store dimension units

product_ids – An array of unique **product_ids** contained in this package

product_count – The number of elements in **product_ids**

fields – A structure containing the module's package-level fields for this package. There will be one member for each HTML input element.

Example:
```
<input type="text" name="{ l.field_prefix $
        'example_field' }" value="{ encodeentities
        ( l.fields:example_field ) }">
```

The **NAME** attribute of the input field is constructed using **field_prefix** (the package specific prefix) and a field-specific component (the text `'example_field'`). In this case, the value of the field will be available to the module as the member "example_field" in **l.fields**, as you can see in the **VALUE** attribute above.

package_count The number of packages in **package_list**

Return Value

1 on success

0 on error

Framework Support Feature (skins)

Modules that implement the **Framework Support** feature (**skins**) support being saved within a Framework.

The **skins** feature includes the following functions:

- **SkinsComponentModule_Description**
- **SkinsComponentModule_Export** (deprecated)
- **SkinsComponentModule_Export_Item**

SkinsComponentModule_Description

This function returns a textual description of the component module's functionality for a single item. In the current software, this text is displayed as a tooltip when hovering over the **Help** link next to a framework component when saving a framework.

Syntax

SkinsComponentModule_Description (module var, item var)

Parameters

module The **Module** record of the current module

item The **item** record of the item for which to return a description. Some modules will use this parameter to determine what pages reference the item and include a list of pages in the description.

Return Value

A textual description. HTML should not be used.

SkinsComponentModule_Export

This function is called when saving a framework to allow the module to export the settings for all items and pages that reference the component. The output should be in Miva Merchant XML provisioning format and be placed into the **output** parameter.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should implement **SkinsComponentModule_Export_Item**.

Supported API Version

Supported in versions less than 5.60

Syntax

SkinsComponentModule_Export (module var, output var)

Parameters

module The **Module** record of the current module

output A variable that will receive the provisioning XML

Return Value

Ignored

SkinsComponentModule_Export_Item

When saving a framework, the system calls this function once for each item registered to a component. When called, the module should output the XML provisioning code required to replicate the settings for the specified item on every page to which it is assigned. Typically, a module would loop through the list of pages to which the item is assigned, exporting the settings for each page separately.

Example

```
<MvFOREACH ITERATOR = "l.page" ARRAY = "l.pages" COUNT = "{ [
 g.Module_Feature_TUI_DB ].PageList_Load_Item_Runtime( l.item:id, l.page ) }">
    …
    generate provisioning code for l.item on page l.page
    …
</MvFOREACH>
```

Note: The provisioning code should be appended to the **output** parameter.

Supported API Version

5.60 and higher

Syntax

SkinsComponentModule_Export_Item (module var, item var, output var)

Parameters

module	The **Module** record of the current module
item	An **Item** record identifying the item for which to generate provisioning code
output	An output variable which receives the generated provisioning code

Return Value

Ignored

Store Selection Feature (storeselui)

Modules that implement the **Store Selection** feature (**storeselui**) provide a user interface that allows shoppers to select a store when a domain contains multiple stores.

Note: The **storeselui** and **storeui** features are restricted. The software will prevent third party modules implementing these features from being installed.

The **storeselui** feature includes the following functions:

- **UIModule_StoreSelection_Content**
- **UIModule_StoreSelection_Render**
- **UIModule_StoreSelection_Tabs**
- **UIModule_StoreSelection_Thumbnail**
- **UIModule_StoreSelection_Update**
- **UIModule_StoreSelection_Validate**

UIModule_StoreSelection_Content

Miva Merchant calls this function from the selected **storeselui** module when the Admin displays the **Domain Settings** screen.

Syntax

UIModule_StoreSelection_Content (tab, load_fields)

Parameters

tab	The code of the currently displayed tab
load_fields	**1** if the module should initialize the page state (global variables), **0** if the page state has already been initialized

Return Value

1 on success

0 on error

UIModule_StoreSelection_Render

This function is called to render a store selection UI when there is more than one store in the domain and the parameters for a page hit do not indicate a specific store.

Syntax

UIModule_StoreSelection_Render ()

Return Value

Ignored

UIModule_StoreSelection_Tabs

Admin calls this function when displaying the **Domain Settings** screen. It returns a tab list using the following format:

```
<code>:<Description>,<code2>:<Description2>
```

Codes must be unique across modules. Miva Merchant recommends that the codes include the module code. The module may specify as many *<code>:<Description>* pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Syntax

UIModule_StoreSelection_Tabs ()

Return Value

A tab list (as described above)

UIModule_StoreSelection_Thumbnail

This function returns a URI or URL to a graphic that contains a thumbnail example of what the store selection layout looks like.

Note: This function is not called in the current version of the software as alternate store selection layout has been disabled since PR5.

Syntax

UIModule_StoreSelection_Thumbnail ()

Return Value

A URI relative to the base URL for graphics or URL. The default store selection layout thumbnail is 200x150 pixels.

UIModule_StoreSelection_Update

This function is called when the user presses the **Update** button to update the fields on the tab(s) drawn by **UIModule_StoreSelection_Content**.

Syntax

UIModule_StoreSelection_Update ()

Return Value

1 on success

0 on error

UIModule_StoreSelection_Validate

This function is called to validate the fields on the tab(s) drawn by **UIModule_StoreSelection_Content**, prior to calling **UIModule_StoreSelection_Update**.

Syntax

UIModule_StoreSelection_Validate ()

Return Value

1 on success

0 on error

Shopping UI Feature (storeui)

Modules that implement the **Shopping UI** feature (**storeui**) provide a basic structure for the shopping interface and are responsible for creating an initial set of pages and items when a store is created.

Note: The **storeselui** and **storeui** features are restricted. The software will prevent third party modules implementing these features from being installed.

The **storeui** feature includes the following functions:

- **StoreUIModule_Create_Frameworks**
- **StoreUIModule_Create_Items**
- **StoreUIModule_Create_Pages**
- **StoreUIModule_Create_URIs**
- **StoreUIModule_Default_Framework**
- **StoreUIModule_Dispatch**
- **StoreUIModule_Exception**
- **StoreUIModule_Screen_Secure**
- **StoreUIModule_Thumbnail**

StoreUIModule_Create_Frameworks

When a new store is created, this function is called in the UI module that the user selected for the new store so the module can register any default frameworks that should be present in a newly created store. Frameworks are registered by calling the **Skin_Directory_Verify** and **SKIN_Sample_Retrieve** functions in **features/tui/tui_ut.mv**.

Supported API Version

5.60 and higher

Syntax

StoreUIModule_Create_Frameworks (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Example

```
<MvFUNCTION NAME = "StoreUIModule_Create_Frameworks" PARAMETERS = "module var"
 STANDARDOUTPUTLEVEL = "">
    <MvASSIGN NAME = "l.skin_code"  VALUE = "default_fw">
    <MvIF EXPR = "{ NOT [ g.Module_Feature_TUI_UT ].SKIN_Directory_Verify(
     l.skin_code, l.skin_dir, l.skin_file ) OR
                   NOT [ g.Module_Feature_TUI_UT ].SKIN_Sample_Retrieve(
                  l.skin_code ) }">
        <MvFUNCTIONRETURN VALUE = 1>
    </MvIF>

    <MvASSIGN NAME = "l.skin_code"  VALUE = "css_fw">
    <MvIF EXPR = "{ NOT [ g.Module_Feature_TUI_UT ].SKIN_Directory_Verify(
     l.skin_code, l.skin_dir, l.skin_file ) OR
                   NOT [ g.Module_Feature_TUI_UT ].SKIN_Sample_Retrieve(
                  l.skin_code ) }">
        <MvFUNCTIONRETURN VALUE = 1>
    </MvIF>

    <MvFUNCTIONRETURN VALUE = 1>
</MvFUNCTION>
```

StoreUIModule_Create_Items

When a new store is being created, this function is called to allow the selected UI module to create the default list of items for the new store. Items are typically registered by calling the **TemplateManager_Create_Item** function in **features/tui/tui_mgr.mv**. This function is called before **StoreUIModule_Create_Pages** so that the items will be registered with the template manager before the default set of pages is created.

Syntax

StoreUIModule_Create_Items (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

StoreUIModule_Create_Pages

When a new store is being created, this function is called to allow the selected UI module to create the default set of pages for the new store. The pages are created by calling the **TemplateManager_Create_Page** function in **features/tui/tui_mgr.mv**. UI Modules should pass their

own module ID as the **ui_id** parameter to that function, which will prevent the user from deleting the page manually.

Syntax

StoreUIModule_Create_Pages (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

StoreUIModule_Create_URIs

This function creates URI entries for the URI management subsystem on store creation. For example, **cssui.mv** creates URI entries for the **Checkout**, **Customer**, **Affiliate**, and **Checkout: Special Offer(s)** screens.

Supported API Version

9.04 and higher

Syntax

StoreUIModule_Create_URIs(module var, pg_regen)

Parameters

module The **Module** record of the current module

pg_regen Either NULL or ***action:status***, where ***status*** is the URL redirect code (**200** by default) and ***action*** is one of the following:

"new" – Add a new non-canonical URI if one does not already exist;
"replace" – Add or create a new URI, making it canonical, and making any existing URI for that screen or page non-canonical;
"sole" – Create a new canonical URI if a URI does not exist for that screen or page.

Return Value

1 on success

0 on error

StoreUIModule_Default_Framework

This function returns the code for the default framework for the **storeui** module. For example, **cssui.mv** returns "cssui_default_fw".

Supported API Version

9.04 and higher

Syntax

StoreUIModule_Default_Framework(module var)

Parameters

module The **Module** record of the current module

Return Value

The default framework code as specified by the **storeui** module

StoreUIModule_Dispatch

This function is called prior to rendering a page to allow the UI module to intercept the page render request. For example, MMUI and CSSUI use this function to turn the dynamic page code ACNT into either ACED or ACAD depending on whether a shopper is logged into their customer account. Other uses include preventing display of certain pages unless certain criteria are met (such as minimum order requirements) or performing session verification to make sure that a secondary access token is present before displaying sensitive information

Syntax

StoreUIModule_Dispatch (module var, page var)

Parameters

module The **Module** record of the current module

page (Input and output) On input, the original requested page code. The module may change this value to cause a different page to be rendered.

Return Value

1 if the page render should proceed as normal

0 if the module wants to prevent the default page render

-1 on error

StoreUIModule_Exception

This function is called when a UI exception occurs. UI exceptions can occur during input validation prior to an action, or by a component module using the **TemplateManager_Throw_Exception** function. This allows the UI module to take whatever action is required to handle the exception, for example, setting an error message and/or redirecting the user to an error page.

Following is a list of the system UI exception codes as of PR8 Update 5. Modules may also throw other exception codes.

- **inventory_limited**
- **inventory_out**
- **basket_changed**
- **order_invalid_info**
- **order_invalid_payment**
- **order_authorizationfailure**
- **order_alreadyprocessed**
- **order_basketchanged**
- **order_invoice**
- **page_not_found**
- **product_not_found**
- **category_not_found**
- **product_attributes**
- **invalid_variant**
- **upsell_attributes**
- **upsell_attributes_multiple**
- **upsell_toomanyselected**
- **customer_invalid_login**
- **customer_email_error**
- **customer_email_sent**
- **customer_invalid_addinfo**
- **customer_invalid_editinfo**
- **affiliate_invalid_login**
- **affiliate_email_error**
- **affiliate_email_sent**
- **affiliate_invalid_addinfo**
- **affiliate_invalid_editinfo**
- **store_offline**
- **no_payment_methods**
- **no_shipping_methods**
- **customer_invalid_session**
- **affiliate_invalid_session**

- **checkout_invalid_session**
- **invoice_invalid_session**

Syntax

StoreUIModule_Exception (module var, code)

Parameters

module	The **Module** record of the current module
code	The code of the UI exception

Return Value

1 on success

0 on error

StoreUIModule_Screen_Secure

This function returns a boolean specifying that the given screen should be rendered securely (using 'https://' instead of 'http://'). For example, **cssui.mv** returns true for screen codes "OINF", "ACNT", "AFAE", and "OUSL", and false for any other screen codes.

Supported API Version

9.04 and higher

Syntax

StoreUIModule_Screen_Secure(module var, screen)

Parameters

module	The **Module** record of the current module
screen	Screen code to determine if a screen is to be rendered securely

Return Value

Boolean

StoreUIModule_Thumbnail

This function returns a URI or URL to a graphic that contains a thumbnail example of what the UI module's shopping interface looks like.

Note: This function is not called in the current version of the software.

Syntax

StoreUIModule_Thumbnail (module var)

Parameters

module The **Module** record of the current module

Return Value

A URI relative to the base URL for graphics or URL. The MMUI thumbnail is 200x150 pixels.

Store Wizards Feature (storewizard)

Modules that implement the **Store Wizards** feature (**storewizard**) provide top-level wizard functionality to accomplish common tasks in the Administrative Interface at the store level for a store.

The **storewizard** feature includes the following functions:

- **StoreWizardModule_Action**
- **StoreWizardModule_Content**
- **StoreWizardModule_Icon**
- **StoreWizardModule_Logo**
- **StoreWizardModule_Privileges**
- **StoreWizardModule_Title**
- **StoreWizardModule_Validate**
- **StoreWizardModule_Validate_Step**

StoreWizardModule_Action

Miva Merchant calls this function after the user completes all steps in the wizard and after a successful return from **StoreWizardModule_Privileges** and **StoreWizardModule_Validate**. It can be used to deliver instructions about what to do with or in reaction to the input submitted by the user.

Syntax

StoreWizardModule_Action (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

StoreWizardModule_Content

Miva Merchant calls this function each time the wizard displays a new screen. It provides the instructions for the main portion of the wizard screen.

Syntax

StoreWizardModule_Content (module var, step, load_fields)

Parameters

module	The **Module** record of the current module
step	The current step. Used to display only certain information on a given screen.
load_fields	Used to load initial data

Return Value

1 on success

0 on error

StoreWizardModule_Icon

This function is used to define the path and name of the icon graphic that is displayed on the **admin.mv** front page under the store section.

Note: Icons have not been used in wizard modules since PR5.

Supported API Version

Supported in all versions but not called

Syntax

StoreWizardModule_Icon (module var)

Parameters

module	The **Module** record of the current module

Return Value

A string showing the path to the icon image

StoreWizardModule_Logo

This function is used to define the path and name of the logo graphic that is displayed in the upper left corner of the wizard. It is called each time the wizard displays a new screen.

Syntax

StoreWizardModule_Logo (module var)

Parameters

module The **Module** record of the current module

Return Value

A string showing the path to the logo image

StoreWizardModule_Privileges

Miva Merchant calls this function when the Admin expands the store portion of its left menu.

Syntax

StoreWizardModule_Privileges (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if the current user has privileges to run the wizard

0 if the current user does not have privileges to run the wizard

StoreWizardModule_Title

Miva Merchant calls this function each time the Admin displays a new wizard screen.

Syntax

StoreWizardModule_Title (module var, step)

Parameters

module The **Module** record of the current module

step The current step. Used to display different titles for different steps.

Return Value

A string showing the title to display

StoreWizardModule_Validate

Miva Merchant calls this function after the user completes all the steps in the wizard. Instructions should be included in the function on an acceptable format for information the Administrator submits.

Syntax

StoreWizardModule_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

StoreWizardModule_Validate_Step

Miva Merchant calls this function each time the Admin receives input from the submission of a wizard screen (one screen per step). In case of failure, the Admin will not allow the wizard to move onto the next screen. Instructions should be included in the function on an acceptable format for information the Administrator submits. Flags can be set for use by the UI to indicate an invalid state.

Syntax

StoreWizardModule_Validate_Step (module var, step)

Parameters

module The **Module** record of the current module

step The numeric identifier (**1** through **N**) of the step being validated

Return Value

1 on success

0 on error

System Extensions Feature (system)

Modules that implement the **System Extensions** feature (**system**) hook into the standard actions in **merchant.mv** allowing the module to augment, modify or replace the functionality of the standard action handler, or to implement new screens or actions.

The **system** feature includes the following functions:

- **SystemModule_Action**
- **SystemModule_Screen**
- **SystemModule_UIException**

SystemModule_Action

Miva Merchant calls this function for every form submit that passes an action value or list of values. With this method, every system module is called for each action.

Syntax

SystemModule_Action (module var, action)

Parameters

module The **Module** record of the current module

action The current action passed by merchant from the list of actions passed by the form submit

Return Value

1 on success

0 on error

-1 exits **SystemModule_Action** without performing default behavior

SystemModule_Screen

Miva Merchant calls this function when displaying a screen.

Syntax

SystemModule_Screen (module var, screen)

Parameters

module The **Module** record of the current module

screen The global screen value passed through a form or as part of the URL (e.g., **Screen=CTGY**)

Return Value

1 on success

0 on error

-1 exits **SystemModule_Screen** without performing default behavior

SystemModule_UIException

Miva Merchant calls this function during a UI Exception.

Syntax

SystemModule_UIException (module var, exception)

Parameters

module	The **Module** record of the current module
exception	The code value of the exception

Return Value

1 on success

0 on error

-1 exits **SystemModule_UIException** without performing default behavior

Sales Tax Calculation Feature (tax)

Modules that implement the **Sales Tax Calculation** feature (**tax**) provide sales tax calculation for a store.

The **tax** feature includes the following functions:

- **TaxModule_Calculate_Basket**
- **TaxModule_Order_Field**
- **TaxModule_Order_Fields**
- **TaxModule_Order_Hide_Fields**
- **TaxModule_Order_Invalid**
- **TaxModule_Order_Prompt**
- **TaxModule_Order_Required**
- **TaxModule_Order_Validate**
- **TaxModule_ProcessOrder**

TaxModule_Calculate_Basket

Miva Merchant calls this function when responding to a form submission from the shopper containing **Action = CTAX**. This function should contain the instructions necessary to insert a 'TAX' basket charge in the basket.

Syntax

TaxModule_Calculate_Basket (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

TaxModule_Order_Field

This function creates the HTML code needed to display the tax field identified by the **field_id** parameter (set in **TaxModule_Order_Fields**).

Syntax

TaxModule_Order_Field (module var, field_id)

Parameters

module The **Module** record of the current module

field_id The code set in **TaxModule_Order_Fields**

Return Value

1 if HTML is output

0 if nothing is set

Note: The return value is ignored by Miva Merchant.

TaxModule_Order_Fields

The Miva Merchant UI calls this function at each checkout screen to determine which, if any, tax fields to display.

Syntax

TaxModule_Order_Fields (module var)

Parameters

module The **Module** record of the current module

Return Value

A comma separated list of field identifiers of the form:

id[,id,id,id-]

TaxModule_Order_Hide_Fields

The Miva Merchant UI calls this function when displaying the OSEL screen. The purpose of this function is to insert hidden fields on checkout screens to carry data through, as form variables, from earlier screens (such as OINF) through later screens (such as OSEL) until the shopper is able to submit a form with **Action = CTAX**.

Syntax

TaxModule_Order_Hide_Fields (module var)

Parameters

module The **Module** record of the current module

Return Value

Ignored

TaxModule_Order_Invalid

The Miva Merchant UI consults this function to determine whether to highlight a given field to indicate that the last information the shopper entered in that field was invalid. The function itself may make that determination based on an invalidity flag set in **TaxModule_Order_Validate**. The strategy is the same as used for **PaymentModule_Payment_Invalid** and **PaymentModule_Payment_Validate**.

Syntax

TaxModule_Order_Invalid (module var, field_id)

Parameters

module The **Module** record of the current module

field_id The code set in **TaxModule_Order_Fields**

Return Value

1 if the field submission is invalid

0 if the field submission is valid

TaxModule_Order_Prompt

This function returns a text string for display beside the tax field identified by the **field_id** parameter (set in **TaxModule_Order_Fields**).

Syntax

TaxModule_Order_Prompt (module var, field_id)

Parameters

module The **Module** record of the current module

field_id The code set in **TaxModule_Order_Fields**

Return Value

A string showing the tax field prompt text

TaxModule_Order_Required

The Miva Merchant UI consults this function to determine whether or not a field needs validation, such as would be necessary to make sure that the shopper made a selection on a field.

Syntax

TaxModule_Order_Required (module var, field_id)

Parameters

module The **Module** record of the current module

field_id The code set in **TaxModule_Order_Fields**

Return Value

1 if the field needs validation

0 if the field is not required

TaxModule_Order_Validate

Miva Merchant calls **Action_Save_OrderInformation** in response to a form submission containing Action=ORDR, as is the case with the form shoppers submit from the OINF page. The function provides an opportunity to check if the tax information is acceptable. One use is to set invalidity flags for consultation by **TaxModule_Order_Invalid**. **TaxModule_Order_Invalid** can check the status of a given flag (for example, an ID number) and decide whether to declare the submission invalid on that basis. (For another implementation of the same strategy, see **PaymentModule_Payment_Validate** and **PaymentModule_Payment_Invalid**.) If the function returns 0, the UI will return to the previous page. The UI will know from the invalidity flags that the previous submissions were invalid and it will highlight the relevant input fields appropriately.

Syntax

TaxModule_Order_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

TaxModule_ProcessOrder

Miva Merchant calls this function after creating an order but before calling **FulfillmentModule_Process_Order**. The function provides an opportunity to perform operations at the time the shopper places the order, such as calculating taxes in real time.

Syntax

TaxModule_ProcessOrder (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Store Utilities Feature (util)

Modules that implement the **Store Utilities** feature (**util**) provide user interface elements and operations available under the **Utilities** link in the left navigation of the Administrative Interface.

The **util** feature includes the following functions:

- **StoreUtilityModule_Action**
- **StoreUtilityModule_Action_Privileges**
- **StoreUtilityModule_LeftNavigation**
- **StoreUtilityModule_Screen**
- **StoreUtilityModule_Screen_Privileges**
- **StoreUtilityModule_Validate**

StoreUtilityModule_Action

The Miva Merchant Admin calls this function if **StoreUtilityModule_Validate** returns a success.

Syntax

StoreUtilityModule_Action (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

StoreUtilityModule_Action_Privileges

This function allows modules to define custom action privileges without requiring users to have the **Store Utility Settings** privileges assigned.

Supported API Version

9.05 and higher

Syntax

StoreUtilityModule_Action_Privileges (module var)

Parameters

module The **Module** record of the current module

Return Value

1 user has sufficient privileges as defined by the module

0 user does not have sufficient privileges

-1 perform default privilege checks (as if the module were an older API version)

StoreUtilityModule_LeftNavigation

The Miva Merchant Admin calls this function to determine what to place in its left menu. The **indent** parameter tells Admin at what depth of indentation to display the link. For example, an **indent** value of '2' would represent the second indent level. The variable Screen=SUTL can be placed in the link, which will cause Admin to call **StoreUtilityModule_Screen**.

Syntax

StoreUtilityModule_LeftNavigation (module var, indent)

Parameters

module The **Module** record of the current module

indent The indentation level for the link position in the Admin

Return Value

1 on success

0 on error

StoreUtilityModule_Screen

This function tells the Miva Merchant Admin what to display as main content when rendering the SUTL screen.

Syntax

StoreUtilityModule_Screen (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

StoreUtilityModule_Screen_Privileges

This function allows modules to define custom screen privileges without requiring users to have the **Store Utility Settings** privileges assigned.

Supported API Version

9.05 and higher

Syntax

StoreUtilityModule_Screen_Privileges (module var)

Parameters

module The **Module** record of the current module

Return Value

1 user has sufficient privileges as defined by the module

0 user does not have sufficient privileges

-1 perform default privilege checks (as if the module were an older API version)

StoreUtilityModule_Validate

Miva Merchant Admin calls this function when processing a form submission that includes Action=SUTL.

Syntax

StoreUtilityModule_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Affiliate Add/Edit Screen Feature (vis_affil)

Modules that implement the **Affiliate Add/Edit Screen** feature (**vis_affil**) can add tabs to the **Affiliate Add/Edit** screen, and are notified when Affiliates are created, updated, or deleted from the **Affiliate Add/Edit** screen.

The **vis_affil** feature includes the following functions:

- **Module_Affiliate_Content**
- **Module_Affiliate_Delete**
- **Module_Affiliate_Head**
- **Module_Affiliate_Insert**
- **Module_Affiliate_Tabs**
- **Module_Affiliate_Update**
- **Module_Affiliate_Validate**

Module_Affiliate_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **affiliate:id** is **0**, the system is requesting tabs for the **Add Affiliate** screen. Otherwise, tabs are being

requested for the **Edit Affiliate** screen and **affiliate** will contain the record being edited. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for the Affiliate being edited.

Syntax

Module_Affiliate_Content (module var, tab, load_fields, affiliate var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present
affiliate	The **Affiliate** record that is being edited or NULL if adding a record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Affiliate_Delete

This function performs module-specific actions when an Affiliate is deleted. It is called prior to the Affiliate record being deleted from the database.

Syntax

Module_Affiliate_Delete (module var, affiliate var)

Parameters

module	The **Module** record of the current module
affiliate	The **Affiliate** record that will be deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Affiliate_Head

This function allows the module to output content in the HTML **<head>** tag of the **Add/Edit Affiliate** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Affiliate_Head(module var, tab, affiliate var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
affiliate	The record of the affiliate being edited or NULL if adding an affiliate

Return Value

1 on success
0 on error

Module_Affiliate_Insert

This function performs module-specific actions when an affiliate is created. It is called after the **Affiliate** record has been inserted into the database.

Syntax

Module_Affiliate_Insert (module var, affiliate var)

Parameters

module	The **Module** record of the current module
affiliate	The **Affiliate** record that will be deleted

Return Value

1 on success
0 on error

Note: Error messages should be reported via the **Error** function.

Module_Affiliate_Tabs

This function returns a list of tab codes and titles that are provided by the module. If **affiliate:id** is **0**, the system is requesting tabs for the **Add Affiliate** screen. Otherwise, tabs are being requested for the **Edit Affiliate** screen and the affiliate will contain the record being edited.

Syntax

Module_Affiliate_Tabs (module var, affiliate var)

Parameters

module	The **Module** record of the current module
affiliate	The **Affiliate** record that is being edited or NULL if adding a record

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Affiliate_Update

This function performs module-specific actions when an affiliate is updated. It is called after the **Affiliate** record has been updated in the database.

Syntax

Module_Affiliate_Update (module var, affiliate var)

Parameters

module	The **Module** record of the current module
affiliate	The **Affiliate** record that will be deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Affiliate_Validate

This function performs validation of additional tab content when an affiliate is added or updated. If the global variable **Edit_Affiliate** is empty, the validation is being performed for an addition. Otherwise, **Edit_Affiliate** will contain the code of the affiliate being updated.

Syntax

Module_Affiliate_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is unsuccessful

Note: Modules can call the **FieldError** function to report invalid input fields.

Affiliate Batch Edit Screen Feature (vis_affilbe)

Modules that implement the **Affiliate Batch Edit Screen** feature (**vis_affilbe**) are notified of actions that occur on the **Affiliate Batch Edit** screen.

The **vis_affilbe** feature includes the following functions:

- **Module_Affiliate_BatchEdit_Content**
- **Module_Affiliate_BatchEdit_Delete**
- **Module_Affiliate_BatchEdit_Head**
- **Module_Affiliate_BatchEdit_Tabs**
- **Module_Affiliate_BatchEdit_Update**
- **Module_Affiliate_BatchEdit_Validate**

Module_Affiliate_BatchEdit_Content

This function is called to render the content of an **Affiliate Batch Edit** screen tab. When the **tab** parameter matches one of the tabs drawn by this module, it draws the controls (using HTML) meant to be visible on the tab. When the **tab** parameter does not match, the module hides any input values in HTML hidden **<input>** tags.

Supported API Version

5.70 and higher

Syntax

Module_Affiliate_BatchEdit_Content(module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab. It can be one of the module's tabs or one of the system generated tabs.
load_fields	This value is set to **1** when the page is initialized so that the module can load default values for any input fields. On subsequent calls, the value is **0** and the module is expected to retain the initially loaded values using hidden input fields, etc.

Return Value

1 on success

0 on error

Module_Affiliate_BatchEdit_Delete

This function performs module-specific actions when affiliates are deleted. This function is called for each affiliate deleted on the **Affiliate Batch Edit** screen. Multiple affiliates can be deleted at once, which will result in multiple calls to this function. This function is called prior to the **Affiliate** record being deleted from the database.

Syntax

Module_Affiliate_BatchEdit_Delete (module var, affiliate var)

Parameters

module	The **Module** record of the current module
affiliate	The **Affiliate** record being deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Affiliate_BatchEdit_Head

This function allows the module to output content in the HTML **<head>** tag of the **Affiliate Batch Edit** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Affiliate_BatchEdit_Head(module var, tab)

Parameters

module The **Module** record of the current module

tab The code of the currently visible tab

Return Value

1 on success

0 on error

Module_Affiliate_BatchEdit_Tabs

This function returns a string that identifies the tabs to be added to the default list of system generated tabs. The string takes the following form:

<code>:<Description>,<code2>:<Description2>

The module may specify as many *<code>:<Description>* pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Supported API Version

5.70 and higher

Syntax

Module_Affiliate_BatchEdit_Tabs(module var)

Parameters

module The **Module** record of the current module

Return Value

A string describing the tabs to be added (see description above)

Module_Affiliate_BatchEdit_Update

This function performs module-specific actions when an affiliate is updated. It is called when an affiliate is updated using the **Edit Here** button on the **Affiliate Batch Edit** screen, after the **Affiliate** record has been updated in the database.

Syntax

Module_Affiliate_BatchEdit_Update (module var, affiliate var)

Parameters

module	The **Module** record of the current module
affiliate	The **Affiliate** record being updated

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Affiliate_BatchEdit_Validate

This function performs module-specific validation. It is called to validate input when an affiliate is edited using the **Edit Here** button on the **Affiliate Batch Edit** screen.

Syntax

Module_Affiliate_BatchEdit_Validate (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 if all fields pass validation

0 if any fields do not pass validation

Note: Validation errors can be reported via the **FieldError** function.

Category Add/Edit Screen Feature (vis_category)

Modules that implement the **Category Add/Edit Screen** feature (**vis_category**) can add tabs to the **Category Add/Edit** screen and are notified when categories are created, updated, or deleted from the **Category Add/Edit** screen.

The **vis_category** feature includes the following functions:

- **Module_Category_Content**
- **Module_Category_Delete**
- **Module_Category_Head**
- **Module_Category_Insert**
- **Module_Category_Tabs**
- **Module_Category_Update**
- **Module_Category_Validate**

Module_Category_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **category:id** is **0**, the system is requesting tabs for the **Add Category** screen. Otherwise, tabs are being requested for the **Edit Category** screen and category will contain the record being edited. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for the category being edited.

Syntax

Module_Category_Content (module var, tab, load_fields, category var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present
category	The **Category** record of the category being edited or NULL if a category is being added

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Category_Delete

This function performs module-specific actions when a category is deleted. It is called prior to the **Category** record being deleted from the database.

Syntax

Module_Category_Delete (module var, category var)

Parameters

module	The **Module** record of the current module
category	The **Category** record that will be deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Category_Head

This function allows the module to output content in the HTML **<head>** tag of the **Add/Edit Category** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Category_Head(module var, tab, category var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
category	The **Category** record of the category being edited or NULL if a category is being added

Return Value

1 on success

0 on error

Module_Category_Insert

This function performs module-specific actions when a category is created. It is called after the **Category** record has been inserted into the database.

Syntax

Module_Category_Insert (module var, category var)

Parameters

module The **Module** record of the current module

category The newly created **Category** record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Category_Tabs

This function returns a list of tab codes and titles that are provided by the module. If **category:id** is **0**, the system is requesting tabs for the **Add Category** screen. Otherwise, tabs are being requested for the **Edit Category** screen and **category** will contain the record being edited.

Syntax

Module_Category_Tabs (module var, category var)

Parameters

module The **Module** record of the current module

category The **Category** record that is being edited or NULL if adding a record

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Category_Update

This function performs module-specific actions when a category is updated. It is called after the **Category** record has been updated in the database.

Syntax

Module_Category_Update (module var, category var)

Parameters

module The **Module** record of the current module

category The updated **Category** record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Category_Validate

This function performs validation of additional tab content when a category is added or updated. If the global variable **Edit_Category** is empty, the validation is performed for an addition. Otherwise, **Edit_Category** contains the code of the category being updated.

Syntax

Module_Category_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Category Batch Edit Screen Feature (vis_categorybe)

Modules that implement the **Category Batch Edit Screen** feature (**vis_categorybe**) are notified of actions that occur on the **Category Batch Edit** screen.

The **vis_categorybe** feature includes the following functions:

- **Module_Category_BatchEdit_Content**
- **Module_Category_BatchEdit_Delete**
- **Module_Category_BatchEdit_Head**
- **Module_Category_BatchEdit_Tabs**
- **Module_Category_BatchEdit_Update**
- **Module_Category_BatchEdit_Validate**

Module_Category_BatchEdit_Content

This function is called to render the content of a **Category Batch Edit** screen tab. When the **tab** parameter matches one of the tabs drawn by this module, it draws the controls (using HTML) meant to be visible on the tab. When the **tab** parameter does not match, the module hides any input values in HTML hidden **<input>** tags.

Supported API Version

5.70 and higher

Syntax

Module_Category_BatchEdit_Content(module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab. It can be one of the module's tabs or one of the system generated tabs.
load_fields	This value is set to **1** when the page is initialized so that the module can load default values for any input fields. On subsequent calls, the value is **0** and the module is expected to retain the initially loaded values using hidden input fields, etc.

Return Value

1 on success

0 on error

Module_Category_BatchEdit_Delete

This function performs module-specific actions when categories are deleted. It is called for each category deleted on the **Category Batch Edit** screen. Multiple categories can be deleted at once, which will result in multiple calls to this function. This function is called prior to the **Category** record being deleted from the database.

Syntax

Module_Category_BatchEdit_Delete (module var, category var)

Parameters

module	The **Module** record of the current module
category	The **Category** record being deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Category_BatchEdit_Head

This function allows the module to output content in the HTML **<head>** tag of the **Category Batch Edit** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Category_BatchEdit_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab

Return Value

1 on success

0 on error

Module_Category_BatchEdit_Tabs

This function returns a string that identifies the tabs to be added to the default list of system generated tabs. The string takes the following form:

```
<code>:<Description>,<code2>:<Description2>
```

The module may specify as many `<code>:<Description>` pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Supported API Version

5.70 and higher

Syntax

Module_Category_BatchEdit_Tabs(module var)

Parameters

module The **Module** record of the current module

Return Value

A string describing the tabs to be added (see description above)

Module_Category_BatchEdit_Update

This function performs module-specific actions when a category is updated. It is called when a category is updated using the **Edit Here** button on the **Category Batch Edit** screen. This function is called after the **Category** record has been updated in the database.

Syntax

Module_Category_BatchEdit_Update (module var, category var)

Parameters

module The **Module** record of the current module

category The **Category** record being updated

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Category_BatchEdit_Validate

This function performs module-specific validation. It is called to validate input when a category is edited using the **Edit Here** button on the **Category Batch Edit** screen.

Syntax

Module_Category_BatchEdit_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if all fields pass validation

0 if any fields do not pass validation

Note: Validation errors can be reported via the **FieldError** function.

Customer Add/Edit Screen Feature (vis_cust)

Modules that implement the **Customer Add/Edit Screen** feature (**vis_cust**) can add tabs to the Administrative Interface **Customer Add/Edit** screen and are notified when customers are created, updated, or deleted from the **Customer Add/Edit** screen.

- **Module_Customer_Content**
- **Module_Customer_Delete**
- **Module_Customer_Head**
- **Module_Customer_Insert**
- **Module_Customer_Tabs**
- **Module_Customer_Update**
- **Module_Customer_Validate**

Module_Customer_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **customer:id** is **0**, the system is requesting tabs for the **Add Customer** screen. Otherwise, tabs are being requested for the **Edit Customer** screen and **customer** will contain the record being edited. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field that normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for the customer being edited.

Syntax

Module_Customer_Content (module var, tab, load_fields, customer var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present
customer	The **Customer** record of the category being edited or NULL if a category is being added

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Customer_Delete

This function performs module-specific actions when a customer is deleted from the Administrative Interface. It is called prior to the **Customer** record being deleted from the database.

Syntax

Module_Customer_Delete (module var, customer var)

Parameters

module	The **Module** record of the current module
category	The **Customer** record that will be deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Customer_Head

This function allows the module to output content in the HTML **<head>** tag of the **Add/Edit Customer** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Customer_Head(module var, tab, customer var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
customer	The **Customer** record of the customer being edited or NULL if a customer is being added

Return Value

1 on success

0 on error

Module_Customer_Insert

This function performs module-specific actions when a customer is created in the Administrative Interface. It is called after the **Customer** record has been inserted into the database.

Syntax

Module_Customer_Insert (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The newly created **Customer** record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Customer_Tabs

This function returns a list of tab codes and titles that are provided by the module. If **customer:id** is **0**, the system is requesting tabs for the **Add Customer** screen. Otherwise, tabs are being requested for the **Edit Customer** screen and **customer** will contain the record being edited.

Syntax

Module_Customer_Tabs (module var, customer var)

Parameters

module	The **Module** record of the current module
category	The **Category** record that is being edited or NULL if adding a record

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Customer_Update

This function performs module-specific actions when a customer is updated in the Administrative Interface. It is called after the **Customer** record has been updated in the database.

Syntax

Module_Customer_Update (module var, customer var)

Parameters

module The **Module** record of the current module

customer The updated **Customer** record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Customer_Validate

This function performs validation of additional tab content when a customer is added or updated from the Administrative Interface. If the global variable **Edit_Customer** is empty, the validation is being performed for an addition. Otherwise, **Edit_Customer** will contain the code of the customer being updated.

Syntax

Module_Customer_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Customer Batch Edit Screen Feature (vis_custbe)

Modules that implement the **Customer Batch Edit Screen** feature (**vis_custbe**) are notified of actions that occur on the **Customer Batch Edit** screen.

The **vis_custbe** feature includes the following functions:

- **Module_Customer_BatchEdit_Content**
- **Module_Customer_BatchEdit_Delete**
- **Module_Customer_BatchEdit_Head**
- **Module_Customer_BatchEdit_Tabs**
- **Module_Customer_BatchEdit_Update**
- **Module_Customer_BatchEdit_Validate**

Module_Customer_BatchEdit_Content

This function is called to render the content of a **Customer Batch Edit** screen tab. When the **tab** parameter matches one of the tabs drawn by this module, it draws the controls (using HTML) meant to be visible on the tab. When the **tab** parameter does not match, the module hides any input values in HTML hidden **<input>** tags.

Supported API Version

5.70 and higher

Syntax

Module_Customer_BatchEdit_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab. It can be one of the module's tabs or one of the system generated tabs.
load_fields	This value is set to **1** when the page is initialized so that the module can load default values for any input fields. On subsequent calls, the value is **0** and the module is expected to retain the initially loaded values using hidden input fields, etc.

Return Value

1 on success

0 on error

Module_Customer_BatchEdit_Delete

This function performs module-specific actions when customers are deleted. It is called for each customer deleted on the **Customer Batch Edit** screen. Multiple customers can be deleted at once, which will result in multiple calls to this function. This function is called prior to the **Customer** record being deleted from the database.

Syntax

Module_Customer_BatchEdit_Delete (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **Customer** record being deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Customer_BatchEdit_Head

This function allows the module to output content in the HTML **<head>** tag of the **Customer Batch Edit** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Customer_BatchEdit_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Customer_BatchEdit_Tabs

This function returns a string that identifies the tabs to be added to the default list of system generated tabs. The string takes the following form:

```
<code>:<Description>,<code2>:<Description2>
```

The module may specify as many `<code>:<Description>` pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Supported API Version

5.70 and higher

Syntax

Module_Customer_BatchEdit_Tabs(module var)

Parameters

module	The **Module** record of the current module

Return Value

A string describing the tabs to be added (see description above)

Module_Customer_BatchEdit_Update

This function performs module-specific actions when a customer is updated. It is called when a customer is updated using the **Edit Here** button on the **Customer Batch Edit** screen. This function is called after the **Customer** record has been updated in the database.

Syntax

Module_Customer_BatchEdit_Update (module var, customer var)

Parameters

module	The **Module** record of the current module
customer	The **Customer** record being updated

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Customer_BatchEdit_Validate

This function performs module-specific validation. It is called to validate input when a customer is edited using the **Edit Here** button on the **Customer Batch Edit** screen.

Syntax

Module_Customer_BatchEdit_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if all fields pass validation

0 if any fields do not pass validation

Note: Validation errors can be reported via the **FieldError** function.

Domain Settings Screen Feature (vis_domain)

This feature allows modules to add tabs to the **Domain Settings** screen.

The **vis_domain** feature includes the following functions:

- **Module_Domain_Content**
- **Module_Domain_Head**
- **Module_Domain_Tabs**
- **Module_Domain_Update**
- **Module_Domain_Validate**

Module_Domain_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render required HTML content for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Domain_Content(module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state; **0** if the module should expect the state to be present.

Return Value

1 on success

0 on error

Module_Domain_Head

This function allows the module to output content in the HTML **<head>** tag of the **Domain Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Domain_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab

Return Value

1 on success

0 on error

Module_Domain_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Domain_Tabs(module var)

Parameters

module The **Module** record of the current module

Return Value

A tab list describing the tabs provided by this module.

Module_Domain_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Domain_Update(module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Domain_Validate

This function performs validation of module tab content.

Syntax

Module_Domain_Validate(module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Order Fulfillment Settings Screen Feature (vis_fulfill)

Modules that implement the **Order Fulfillment Settings Screen** feature (**vis_fulfill**) provide one or more tabs on the **Order Fulfillment Settings** screen.

The **vis_fulfill** feature includes the following functions:

- **Module_Fulfillment_Content**
- **Module_Fulfillment_Head**
- **Module_Fulfillment_Tabs**
- **Module_Fulfillment_Update**
- **Module_Fulfillment_Validate**

Module_Fulfillment_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Fulfillment_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Fulfillment_Head

This function allows the module to output content in the HTML **<head>** tag of the **Order Fulfillment Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Fulfillment_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success
0 on error

Module_Fulfillment_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Fulfillment_Tabs (module var)

Parameters

module	The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Fulfillment_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Fulfillment_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Fulfillment_Validate

This function performs validation of module tab content.

Syntax

Module_Fulfillment_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Logging Settings Screen Feature (vis_log)

Modules that implement the **Logging Settings Screen** feature (**vis_log**) provide one or more tabs on the **Logging Settings** screen.

The **vis_log** feature includes the following functions:

- **Module_Logging_Content**
- **Module_Logging_Head**
- **Module_Logging_Tabs**
- **Module_Logging_Update**
- **Module_Logging_Validate**

Module_Logging_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Logging_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Logging_Head

This function allows the module to output content in the HTML **<head>** tag of the **Logging Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Logging_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Logging_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Logging_Tabs (module var)

Parameters

module The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Logging_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Logging_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Logging_Validate

This function performs validation of module tab content.

Syntax

Module_Logging_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Order Tabs Feature (vis_order)

Modules that implement the **Order Tabs** feature (**vis_order**) provide one or more tabs which are visible in both the legacy order processing and order detail screens.

The **vis_order** feature includes the following functions:

- **Module_Order_Content**
- **Module_Order_Delete** (deprecated)
- **Module_Order_Delete_Order**
- **Module_Order_Head**
- **Module_Order_Tabs**
- **Module_Order_Update**
- **Module_Order_Validate**

Module_Order_Content

This function renders content for visible tabs or hides state information for invisible tabs. The global variable **Edit_Order** will contain the ID of the order being edited. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for the order being edited.

Syntax

Module_Order_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Order_Delete

This function is called prior to the **Order** record being deleted from the database. It is called when an order is deleted from the **Order Detail** screen, the legacy **View Order** screen, the legacy **Order Batch Edit** screen or when a batch of orders is deleted from the legacy **Batch** screen. To determine which order is being deleted will require determining which location is calling the function and then examining the internal state of the Administrative Interface.

Note: Because the order being deleted is not passed to the module as a parameter, the order must be inferred by examining global variables. Refer to *Appendix D: Deleting Orders* on page 383 for further details.

Note: This function has been deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. New modules should also implement **Module_Order_Delete_Order**.

Supported API Version

Supported in versions less than 5.51 and greater than or equal to 5.02

Syntax

Module_Order_Delete (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Order_Delete_Order

This function performs module-specific actions when an **Order** record is deleted from the database. It is called when an order is deleted from the **Order Detail** screen, the legacy **View Order** screen, the legacy **Order Batch Edit** screen or when a batch of orders is deleted from the legacy **Batch** screen.

Supported API Version

Supported in versions greater than or equal to 5.51

Syntax

Module_Order_Delete_Order (module var, order var)

Parameters

module The **Module** record of the current module

order The **Order** record that will be deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Order_Head

This function allows the module to output content in the HTML **<head>** tag of the Legacy Order Processing **Edit Order** screen and new Order Management tab screens. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Order_Head(module var, tab, order var)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code
order	The **Order** record of the order being displayed

Return Value

1 on success

0 on error

Module_Order_Tabs

This function returns a list of tab codes and titles that are provided by the module. The global variable **Edit_Order** contains the ID of the order being edited.

Syntax

Module_Order_Tabs (module var)

Parameters

module	The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Order_Update

This function performs module-specific actions when an order is updated. It is called prior to the **Order** record being updated in the database. The ID of the order being updated is present in the global variable **Edit_Order**.

Syntax

Module_Order_Update (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Order_Validate

This function performs validation of additional tab content when an order is updated. The ID of the order being updated is present in the global variable **Edit_Order**.

Syntax

Module_Order_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Packaging Rules Screen Feature (vis_pkgrules)

This feature allow modules to add tabs to the **Packaging Rules** screen.

The **vis_pkgrules** feature includes the following functions:

- **Module_PackagingRules_Content**
- **Module_PackagingRules_Head**
- **Module_PackagingRules_Tabs**
- **Module_PackagingRules_Update**
- **Module_PackagingRules_Validate**

Module_PackagingRules_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render HTML content required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain information required when **tab** is not visible.

Syntax

Module_PackagingRules_Content(module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Module_PackagingRules_Head

This function allows the module to output content in the HTML **<head>** tag of the **Packaging Rules** screen. It is useful for outputting CSS styles or external JavaScript references.

Syntax

Module_PackagingRules_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_PackagingRules_Tabs

This function returns a list of tab codes and titles that are provided by the module. These tabs will be added to the **Packaging Rules** screen.

Syntax

Module_PackingRules_Tabs (module var)

Parameters

module	The **Module** record of the current module

Return Value

A tab list describing the tabs provided by this module.

Module_PackagingRules_Update

This function performs module-specific actions when package rules are updated.

Syntax

Module_PackagingRules_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Module_PackagingRules_Validate

This function performs validation of tab content when a package rule is updated.

Syntax

Module_PackagingRules_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Payment Settings Screen Feature (vis_payment)

Modules that implement this feature provide one or more tabs on the **Payment Settings** screen.

The **vis_payment** feature includes the following functions:

- **Module_Payment_Content**
- **Module_Payment_Head**
- **Module_Payment_Tabs**
- **Module_Payment_Update**
- **Module_Payment_Validate**

Module_Payment_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Payment_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Payment_Head

This function allows the module to output content in the HTML **<head>** tag of the **Payment Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Payment_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Payment_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Payment_Tabs (module var)

Parameters

module The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Payment_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Payment_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Payment_Validate

This function performs validation of module tab content.

Syntax

Module_Payment_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Product Add/Edit Screen Feature (vis_product)

Modules that implement this feature can add tabs to the **Product Add/Edit** screen and are notified when products are created, updated, or deleted from the **Product Add/Edit** screen.

The **vis_product** feature includes the following functions:

- **Module_Product_Content**
- **Module_Product_Delete**
- **Module_Product_Head**
- **Module_Product_Insert**
- **Module_Product_Tabs**
- **Module_Product_Update**
- **Module_Product_Validate**

Module_Product_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **product:id** is **0**, the system is requesting tabs for the **Add Product** screen. Otherwise, tabs are being requested for the **Edit Product** screen and **product** will contain the record being edited. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for the product being edited.

Syntax

Module_Product_Content (module var, tab, load_fields, product var)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present
product	The **Product** record that is being edited or NULL if adding a record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Product_Delete

This function performs module-specific actions when a product is deleted. It is called prior to the **Product** record being deleted from the database.

Syntax

Module_Product_Delete (module var, product var)

Parameters

module	The **Module** record of the current module
product	The **Product** record that will be deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Product_Head

This function allows the module to output content in the HTML **<head>** tag of the **Add/Edit Product** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Product_Head(module var, tab, product var)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code
product	The **Product** record of the product being edited or NULL if a product is being added

Return Value

1 on success

0 on error

Module_Product_Insert

This function performs module-specific actions when a product is created. It is called after the **Product** record has been inserted into the database.

Syntax

Module_Product_Insert (module var, product var)

Parameters

module	The **Module** record of the current module
product	The newly created **Product** record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Product_Tabs

Returns a list of tab codes and titles that are provided by the module. If **product:id** is **0**, the system is requesting tabs for the **Add Product** screen. Otherwise, tabs are being requested for the **Edit Product** screen and **product** will contain the record being edited.

Syntax

Module_Product_Tabs (module var, product var)

Parameters

module The **Module** record of the current module

product The **Product** record of the product being edited or NULL if a product is being added

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Product_Update

This function performs module-specific actions when a product is updated. It is called after the **Product** record has been updated in the database.

Syntax

Module_Product_Update (module var, product var)

Parameters

module The **Module** record of the current module

product The updated **Product** record

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Product_Validate

This function performs validation of additional tab content when a product is added or updated. If the global variable **Edit_Product** is empty, the validation is being performed for an addition. Otherwise, **Edit_Product** will contain the code of the product being updated.

Syntax

Module_Product_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Product Batch Edit Screen Feature (vis_productbe)

Modules that implement the **Product Batch Edit Screen** feature (**vis_productbe**) are notified of actions that occur on the **Product Batch Edit** screen.

The **vis_productbe** feature includes the following functions:

- **Module_Product_BatchEdit_Content**
- **Module_Product_BatchEdit_Delete**
- **Module_Product_BatchEdit_Head**
- **Module_Product_BatchEdit_Tabs**
- **Module_Product_BatchEdit_Update**
- **Module_Product_BatchEdit_Validate**

Module_Product_BatchEdit_Content

This function is called to render the content of a **Product Batch Edit** screen tab. When the **tab** parameter matches one of the tabs drawn by this module, it draws the controls (using HTML) meant to be visible on the tab. When the **tab** parameter does not match, the module hides any input values in HTML hidden **<input>** tags.

Supported API Version

5.70 and higher

Syntax

Module_Product_BatchEdit_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab. It can be one of the module's tabs or one of the system generated tabs.
load_fields	This value is set to **1** when the page is initialized so that the module can load default values for any input fields. On subsequent calls, the value is **0** and the module is expected to retain the initially loaded values using hidden input fields, etc.

Return Value

1 on success

0 on error

Module_Product_BatchEdit_Delete

This function performs module-specific actions when products are deleted. This function is called for each product deleted on the **Product Batch Edit** screen. Multiple products can be deleted at once, which will result in multiple calls to this function. This function is called prior to the **Product** record being deleted from the database.

Syntax

Module_Product_BatchEdit_Delete (module var, product var)

Parameters

module	The **Module** record of the current module
product	The **Product** record being deleted

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Product_BatchEdit_Head

This function allows the module to output content in the HTML **<head>** tag of the **Product Batch Edit** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Product_BatchEdit_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Product_BatchEdit_Tabs

This function returns a string that identifies the tabs to be added to the default list of system generated tabs. The string takes the following form:

```
<code>:<Description>,<code2>:<Description2>
```

The module may specify as many `<code>:<Description>` pairs as it likes by separating them with commas. A new tab will be drawn for each pair. If the module returns an empty string, no additional tabs are drawn.

Supported API Version

5.70 and higher

Syntax

Module_Product_BatchEdit_Tabs(module var)

Parameters

module The **Module** record of the current module

Return Value

A string describing the tabs to be added (see description above)

Module_Product_BatchEdit_Update

This function performs module-specific actions when a product is updated. It is called when a product is updated using the **Edit Here** button on the **Product Batch Edit** screen. This function is called after the **Product** record has been updated in the database.

Syntax

Module_Product_BatchEdit_Update (module var, product var)

Parameters

module The **Module** record of the current module

product The **Product** record being updated

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Product_BatchEdit_Validate

This function performs module-specific validation. It is called to validate input when a product is edited using the **Edit Here** button on the **Product Batch Edit** screen.

Syntax

Module_Product_BatchEdit_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if all fields pass validation

0 if any fields do not pass validation

Note: Modules can call the **FieldError** function to report invalid input fields.

Shipping Settings Screen Feature (vis_shipping)

Modules that implement the **Shipping Settings Screen** feature (**vis_shipping**) provide one or more tabs on the **Shipping Settings** screen.

The **vis_shipping** feature includes the following functions:

- **Module_Shipping_Content**
- **Module_Shipping_Head**
- **Module_Shipping_Tabs**
- **Module_Shipping_Update**
- **Module_Shipping_Validate**

Module_Shipping_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Shipping_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Shipping_Head

This function allows the module to output content in the HTML **<head>** tag of the **Shipping Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Shipping_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Shipping_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Shipping_Tabs (module var)

Parameters

module The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Shipping_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Shipping_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Shipping_Validate

This function performs validation of module tab content.

Syntax

Module_Shipping_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Edit Store Screen Feature (vis_store)

Modules that implement the **Edit Store Screen** feature (**vis_store**) can add tabs to the **Edit Store** screen, and are notified when the store settings are updated.

The **vis_store** feature includes the following functions:

- **Module_Store_Content**
- **Module_Store_Head**
- **Module_Store_Tabs**
- **Module_Store_Update**
- **Module_Store_Validate**

Module_Store_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Store_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Store_Head

This function allows the module to output content in the HTML **<head>** tag of the **Edit Store** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Store_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Store_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Store_Tabs (module var)

Parameters

module	The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Store_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Store_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Store_Validate

This function performs validation of module tab content.

Syntax

Module_Store_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

System Extension Settings Screen Feature (vis_system)

Modules that implement the **System Extension Settings Screen** feature (**vis_system**) provide one or more tabs on the **System Extension Settings** screen.

The **vis_system** feature includes the following functions:

- **Module_System_Content**
- **Module_System_Head**
- **Module_System_Tabs**
- **Module_System_Update**
- **Module_System_Validate**

Module_System_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_System_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_System_Head

This function allows the module to output content in the HTML **<head>** tag of the **System Extension Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_System_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_System_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_System_Tabs (module var)

Parameters

module	The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_System_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_System_Update (module var)

Parameters

module	The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_System_Validate

This function performs validation of module tab content.

Syntax

Module_System_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Utility Screen Feature (vis_util)

Modules that implement the **Utility Screen** feature (**vis_util**) provide one or more tabs on the Utility screen.

The **vis_util** feature includes the following functions:

- **Module_Utility_Content**
- **Module_Utility_Head**
- **Module_Utility_Tabs**
- **Module_Utility_Update**
- **Module_Utility_Validate**

Module_Utility_Content

This function renders content for visible tabs or hides state information for invisible tabs. If **tab** is the code of one of the module's tabs, the module should render whatever HTML content is required for that specific tab. For all other tabs (or for all tabs if **tab** is not one of the module's tabs), the module should use hidden HTML **<input>** fields to maintain whatever state information is required when a tab is not visible. Typically, this would involve creating hidden input fields for each HTML form field which normally appears on an invisible tab. If **load_fields** is true, the module should initialize all fields to a default value or current values for fields provided on all tabs.

Syntax

Module_Utility_Content (module var, tab, load_fields)

Parameters

module	The **Module** record of the current module
tab	The code of the currently visible tab
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Utility_Head

This function allows the module to output content in the HTML **<head>** tag of the **Store Utility Settings** screen. It is useful for outputting CSS styles or external JavaScript references.

Supported API Version

5.70 and higher

Syntax

Module_Utility_Head(module var, tab)

Parameters

module	The **Module** record of the current module
tab	The currently visible tab code

Return Value

1 on success

0 on error

Module_Utility_Tabs

This function returns a list of tab codes and titles that are provided by the module.

Syntax

Module_Utility_Tabs (module var)

Parameters

module The **Module** record of the current module

Return Value

An **AdminTabList** describing the tabs provided by this module

Module_Utility_Update

This function saves module-specific settings when the **Update** button is pressed.

Syntax

Module_Utility_Update (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Utility_Validate

This function performs validation of module tab content.

Syntax

Module_Utility_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Wizard Configuration Feature (vis_wizard)

Modules that implement the **Wizard Configuration** feature (**vis_wizard**) and one of the following additional features can be configured using a Miva Merchant-provided wizard:

- **payment** (through the **Set Up Payment** wizard)
- **shipping** (through the **Set Up Shipping** wizard)
- **tax** (thought the **Set Up Sales Tax** wizard)
- **storeui** (through the **Design Your Look** wizard)
- **fulfill** (through the **Set Up Fulfillment** wizard)

The **vis_wizard** feature includes the following functions:

- **Module_Is_Wizardable**
- **Module_Wizard_Action**
- **Module_Wizard_Content**
- **Module_Wizard_Summary_Field**
- **Module_Wizard_Summary_Fields**
- **Module_Wizard_Summary_Prompt**
- **Module_Wizard_Validate**
- **Module_Wizard_Validate_Step**

Module_Is_Wizardable

This function is called when the payment, shipping, or sales tax wizard is launched. It determines which modules of the corresponding feature types to include in the module selection drop-down lists.

Syntax

Module_Is_Wizardable (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if the module is wizardable

0 if the module is not wizardable

Module_Wizard_Action

Admin calls this function after the user completes all of a module's wizard steps and after a successful return from **Module_Wizard_Validate**. Instructions can be placed here regarding what to do with or in response to the input submitted by the user.

Syntax

Module_Wizard_Action (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

Module_Wizard_Content

Admin calls this function each time it displays a new step for a module's wizard.

Syntax

Module_Wizard_Content (module var, step, load_fields)

Parameters

module	The **Module** record of the current module
step	**l.step** is a numerical value representing the user's current position within the wizard
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

Module_Wizard_Summary_Field

This function renders the value of the field identified by the **field_id** parameter received from the list set in **Module_Wizard_Summary_Fields**. For example, it can be used to fill in a table listing each of the tax rates entered by the administrator during use of the wizard.

Syntax

Module_Wizard_Summary_Field (module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	String identifier for the field that is to be added to the summary

Return Value

1 on success

0 on error

Module_Wizard_Summary_Fields

Wizard modules make use of this function along with **Module_Wizard_Summary_Prompt** and **Module_Wizard_Summary_Field** to create a table displaying the inputs provided by the administrator over the course of using the wizard.

Syntax

Module_Wizard_Summary_Fields (module var)

Parameters

module	The **Module** record of the current module

Return Value

A comma separated list of field identifiers of the form:

id[,id,id,id-]

Module_Wizard_Summary_Prompt

This function returns a text string for use as a field label or table heading. The value of the prompt is dependent upon the **field_id** parameter, received from the list set in **Module_Wizard_Summary_Fields**.

Syntax

Module_Wizard_Summary_Prompt (module var, field_id)

Parameters

module	The **Module** record of the current module
field_id	String identifier for the field that is to be added to the summary

Return Value

The prompt for the corresponding field ID as a string with a trailing colon character

Module_Wizard_Validate

Admin calls this function after the user completes all of the module's wizard steps. Instructions should be included in the function to insure that the information the user submits is in an acceptable format.

Syntax

Module_Wizard_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

Module_Wizard_Validate_Step

This function is called after each step of a module's wizard is submitted. It provides a way to validate the data from each step as the user proceeds through the wizard instead of validating all of the steps at once at the end.

Syntax

Module_Wizard_Validate_Step (module var, step)

Parameters

module The **Module** record of the current module

step **I.step** is a numerical value representing the user's current position within the wizard

Return Value

1 on success

0 on error

Domain Wizards Feature (wizard)

Modules that implement the **Domain Wizards** feature (**wizard**) provide top-level wizard functionality for a Miva Merchant installation.

The **wizard** feature includes the following functions:

- **WizardModule_Action**
- **WizardModule_Content**
- **WizardModule_Icon**
- **WizardModule_Logo**
- **WizardModule_Privileges**
- **WizardModule_Title**
- **WizardModule_Validate**
- **WizardModule_Validate_Step**

WizardModule_Action

Admin calls this function after the user completes all steps and after a successful return from **WizardModule_Privileges** and **WizardModule_Validate**. Intructions can be placed here regarding what to do with or in reaction to the input submitted by the user.

Syntax

WizardModule_Action (module var)

Parameters

module The **Module** record of the current module

Return Value

1 on success

0 on error

WizardModule_Content

Admin calls this function each time it displays a new screen. It provides the instructions for the main portion of the wizard screen.

Syntax

WizardModule_Content (module var, step, load_fields)

Parameters

module	The **Module** record of the current module
step	**l.step** is a numerical value representing the user's current position within the wizard
load_fields	**1** if the module should initialize its state, **0** if the module should expect the state to be present

Return Value

1 on success

0 on error

Note: Error messages should be reported via the **Error** function.

WizardModule_Icon

This function is used to define the path and name of the icon graphic that is displayed on the **admin.mv** front page under the Domain section.

Syntax

WizardModule_Icon (module var)

Parameters

module	The **Module** record of the current module

Return Value

A string showing the path to the icon image

WizardModule_Logo

This function is used to define the path and name of the icon graphic that displays in the upper left corner of the wizard. Admin calls this function each time it displays a new screen.

Syntax

WizardModule_Logo (module var)

Parameters

module	The **Module** record of the current module

Return Value

A string showing the path to the logo image

WizardModule_Privileges

Admin calls this function when it builds its left menu.

Syntax

WizardModule_Privileges (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if the current user is an administrator

0 if the current user is not an administrator

WizardModule_Title

Admin calls this function each time it displays a new screen.

Syntax

WizardModule_Title (module var, step)

Parameters

module The **Module** record of the current module

step **I.step** is a numerical value representing the user's current position within the wizard

Return Value

A string showing the title of the wizard

WizardModule_Validate

Admin calls this function after the user completes all steps. Instructions should be included in the function to determine that any information the administrator submits is acceptable in format.

Syntax

WizardModule_Validate (module var)

Parameters

module The **Module** record of the current module

Return Value

1 if validation is successful

0 if validation is not successful

Note: Modules can call the **FieldError** function to report invalid input fields.

WizardModule_Validate_Step

Admin calls this function each time it receives input from the submission of a wizard screen (one screen per step). Instructions should be included in the function to determine that any information the administrator submits is acceptable in format.

Syntax

WizardModule_Validate_Step (module var, step)

Parameters

module	The **Module** record of the current module
step	**I.step** is a numerical value representing the user's current position within the wizard

Return Value

1 on success

0 on failure (in case of failure, Admin will not allow the wizard to move on to the next screen)

Note: Invalidity flags can be set for use by the UI in a fashion similar to that of **PaymentModule_Payment_Invalid**.

APPENDIX A

Miva Script Source Files

The source code for the API functions is contained in the LSK. The core functions can be found in **features**, **lib/dbeng** and **lib/dbprim**. The **modules** directory contains examples of implementations of Miva Merchant features.

The following tables list the available source code files. The first table gives a brief description of each core source file. The second table lists the available modules and the features that they implement. These are provided to developers for reference purposes.

LSK Source Files

The following table shows the primary source files in the Miva Merchant LSK.

Path/Filename	Description
/	
ajax.js	Functions that implement a browser-independent AJAX call mechanism
AttributeMachine.js	Browser-side JavaScript functionality for the Attribute Machine (dynamic attribute based inventory display)
json.mv	Entry point for module and core system functions that communicate using JSON
merchant.mv	Functions that implement the top-level shopping cart interface
runtime.js	Stub JavaScript functions for making AJAX calls into Miva Merchant
admin/	
functions.js	JavaScript stubs to call administrative functionality in **json.mvc**
modal.js	JavaScript functions that provide a modal user interface and dim elements below the current dialog box or other UI element
ui.js	Helper functions for JavaScript/AJAX Administrative UI functionality
ui.mv	Utility functions that draw and update Administrative UI features
v55_ui.js	Legacy JavaScript code used in PR5 and older administrative interface
validate.mv	Administrative UI data validation functions
wizardui.mv	Utility functions for building administrative wizards
features/	
aff/aff_db.mv	Affiliate System database functions
agr/agr_db.mv	Availability Group database functions
att/att_db.mv	Attribute Template database functions
cus/cus_db.mv	Customer database functions
cus/cus_rt.mv	Customer functions for shopping interface
inv/inv_db.mv	Inventory System database functions
inv/inv_rt.mv	Inventory System functions for shopping interface
pgr/pgr_db.mv	Price Group database functions
prv/prv_ad.mv	Provisioning system
rpd/rpd_db.mv	Related Products database functions
rpt/rpt_db.mv	Report System database functions
rpt/rpt_json.mv	JSON entry points for the report subsystem
sbm/sbm_db.mv	Miva Submit database functions
sta/sta_db.mv	Statistics database functions
tui/tui_db.mv	Template System database functions
tui/tui_mgr.mv	Template Manager
tui/tui_ut.mv	Template System utility functions
usl/usl_db.mv	Upsale System database functions

Path/Filename	Description
usl/usl_rt.mv	Upsale System functions for shopping interface
json/	
batches.mv	JSON functions that deal with order batches
countries.mv	JSON functions that deal with the **Countries** and **StoreCountries** tables
customers.mv	JSON functions that deal with the **Customers** table
orders.mv	JSON functions that deal with Orders and Order Items
payments.mv	JSON functions that deal with OrderPayments, authorizing payment, etc.
productkits.mv	JSON functions used by the Inventory Kit Builder
products.mv	JSON functions that deal with Products
productvariants.mv	JSON functions that provide Attribute Inventory functionality used on the **Inventory Variants** tab of the **Edit Product** screen
returns.mv	JSON functions that create and manage OrderReturns
runtime.mv	JSON functions used in the shopping interface for Attribute Inventory and other dynamic functionality
shipments.mv	JSON functions that create and manage OrderShipments
states.mv	JSON functions that deal with the **States** table
store.mv	JSON functions that deal with the **Stores** table
storemodules.mv	JSON functions that deal with the **StoreModules** table
trackinglinks.mv	JSON functions that deal with the **TrackingLinks** table
lib/	
config.mv	Set up global variables required for proper operation of Miva Merchant
crypto_public.mv	Functions for encrypting and decrypting payment data
dbapi_mivasql.mv	Database API compatibility layer – MivaSQL
dbapi_mysql.mv	Database API compatibility layer – MySQL
dbapi_public.mv	Database API compatibility layer
util_public.mv	Helper functions
lib/dbeng/	
attributes.mv	High level database functions that manipulate the **sNN_Attributes** and **sNN_Options** tables
baskets.mv	High level database functions that primarily manipulate the **sNN_Baskets** table
batches.mv	High level database functions that primarily manipulate the **sNN_Batches** table
categories.mv	High level database functions that primarily manipulate the **sNN_Categories** table
countries.mv	High level database functions that primarily manipulate the **Countries** table
create_domain.mv	High level database functions that create and initialize domain-level tables
create_store.mv	High level database functions that create and initialize the tables for a store
delete_domain.mv	High level database functions that drop and clean up domain-level tables
delete_store.mv	High level database functions that delete and clean up the database tables for a store
encryption.mv	High level database functions that manipulate the **sNN_Encryption** and **sNN_Private-Keys** tables

Path/Filename	Description
fields.mv	High level database functions that primarily manipulate the **sNN_StandardFields** table
generatedimages.mv	High level database functions that create and manage dynamically resized images (**sNN_GeneratedImages** table)
images.mv	High level database functions that deal with PR8 and newer additional product images (**sNN_Images** table)
imagetypes.mv	High level database functions that deal with registered image types
keys.mv	High level database functions that generate and maintain domain and store-level unique keys
open.mv	Provides functions for connecting to the database and referencing a specific store's database tables
order_compat.mv	High level functions that implement the 5.5 PR6 compatibility layer for the **sNN_Orders** table
orders.mv	High level database functions that primarily manipulate the **sNN_Orders** table
productimages.mv	High level database functions that manage the relationship between PR8 and newer additional product images and individual product records
productkits.mv	High level database functions that primarily manipulate the **sNN_ProductKits** table
products.mv	High level database functions that primarily manipulate the **sNN_Products** table
productvariants.mv	High level database functions that primarily manipulate the **sNN_ProductVariants** table
runtime.mv	High level database functions that provide shopping-interface specific operations
shipmentbatches.mv	High level database functions that create and manage **sNN_ShipmentBatches** records
sort.mv	High level database functions that implement sorting (product, category, etc.)
sql.mv	SQL query generation functions
states.mv	High level database functions that primarily manipulate the **sNN_States** table
trackinglinks.mv	High level database functions that primarily manipulate the **TrackingLinks** table
util.mv	Database layer utility functions
lib/dbprim	
attributes.mv	Primitive database functions that manipulate the **sNN_Attributes** table
basketcharges.mv	Primitive database functions that manipulate the **sNN_BasketCharges** table
basketitems.mv	Primitive database functions that manipulate the **sNN_BasketItems** table
basketoptions.mv	Primitive database functions that manipulate the **sNN_BasketOptions** table
baskets.mv	Primitive database functions that manipulate the **sNN_Baskets** table
batches.mv	Primitive database functions that manipulate the **sNN_Batches** table
categories.mv	Primitive database functions that manipulate the **sNN_Categories** table
categoryxproduct.mv	Primitive database functions that manipulate the **sNN_CategoryXProduct** table
countries.mv	Primitive database functions that manipulate the **Countries** table
domain.mv	Primitive database functions that manipulate the **Domain** table
domainkeys.mv	Primitive database functions that manipulate the **DomainKeys** table
encryption.mv	Primitive database functions that manipulate the **sNN_Encryption** and **sNN_Private-Keys** tables

Path/Filename	Description
generatedimages.mv	Primitive database functions that manipulate the **sNN_GeneratedImages** table
images.mv	Primitive database functions that manipulate the **sNN_Images** table
imagetypes.mv	Primitive database functions that manipulate the **sNN_ImageTypes** table
launchpad.mv	Primitive database functions that manipulate the **LaunchConfig** table
options.mv	Primitive database functions that manipulate the **sNN_Options** table
ordercharges.mv	Primitive database functions that manipulate the **sNN_OrderCharges** table
orderitems.mv	Primitive database functions that manipulate the **sNN_OrderItems** table
orderoptions.mv	Primitive database functions that manipulate the **sNN_OrderOptions** table
orderpayments.mv	Primitive database functions that manipulate the **sNN_OrderPayments** table
orderreturns.mv	Primitive database functions that manipulate the **sNN_OrderReturns** table
orders.mv	Primitive database functions that manipulate the **sNN_Orders** table
ordershipments.mv	Primitive database functions that manipulate the **sNN_OrderShipments** table
productimages.mv	Primitive database functions that manipulate the **sNN_ProductImages** table
productkits.mv	Primitive database functions that manipulate the **sNN_ProductKits** table
products.mv	Primitive database functions that manipulate the **sNN_Products** table
productvariantparts.mv	Primitive database functions that manipulate the **sNN_ProductVariantParts** table
productvariantpricing.mv	Primitive database functions that manipulate the **sNN_ProductVariantPricing** table
productvariants.mv	Primitive database functions that manipulate the **sNN_ProductVariants** table
seo_settings.mv	Primitive database functions that manipulate the **SEOSettings** table
shipmentbatches.mv	Primitive database functions that manipulate the **sNN_ShipmentBatches** table
standardfields.mv	Primitive database functions that manipulate the **sNN_StandardFields** table
states.mv	Primitive database functions that manipulate the **sNN_States** table
storecountries.mv	Primitive database functions that manipulate the **sNN_StoreCountries** table
storekeys.mv	Primitive database functions that manipulate the **sNN_StoreKeys** table
stores.mv	Primitive database functions that manipulate the **Stores** table
trackinglinks.mv	Primitive database functions that manipulate the **TrackingLinks** table
templates/	
batchreport-order-invoice.css	Cascading style sheet for **Batch Report Order Invoice** page
batchreport-order-invoice.mvt	Template for **Batch Report Order Invoice** page
batchreport-shipment-picklist.css	Cascading style sheet for **Batch Report Shipment Picklist** page
batchreport-shipment-picklist.mvt	Template for **Batch Report Shipment Picklist** page
cssui.css	Default style sheet
cssui-abus.mvt	CSSUI template for **About Us** page
cssui-acad.mvt	CSSUI template for **Customer Create** page
cssui-aced.mvt	CSSUI template for **Customer Edit** page
cssui-acln.mvt	CSSUI template for **Customer Account** page
cssui-afad.mvt	CSSUI template for **Affliate Create** page
cssui-afcl.mvt	CSSUI template for **Affiliate Login** page

Path/Filename	Description
cssui-afed.mvt	CSSUI template for **Affiliate Edit** page
cssui-bask.mvt	CSSUI template for **Basket Contents** page
cssui-bske.mvt	CSSUI template for **Checkout: Basket Empty** page
cssui-ctgy.mvt	CSSUI template for **Category Display** page
cssui-ctus.mvt	CSSUI template for **Contact Us** page
cssui-faqs.mvt	CSSUI template for **FAQs** page
cssui-invc.mvt	CSSUI template for **Invoice** page
cssui-logn.mvt	CSSUI template for **Customer Login** page
cssui-mntn.mvt	CSSUI template for **Maintenance Mode** page
cssui-ntfd.mvt	CSSUI template for **Not Found** page
cssui-ocst.mvt	CSSUI template for **Checkout: Customer Information** page
cssui-omin.mvt	CSSUI template for **Checkout: Minimum Purchase Required** page
cssui-opay.mvt	CSSUI template for **Checkout: Payment Information** page
cssui-oprc.mvt	CSSUI template for **Order Already Processed** page
cssui-ordh.mvt	CSSUI template for **Order History List** page
cssui-ordl.mvt	CSSUI template for **Order: Customer Login** page
cssui-ords.mvt	CSSUI template for **Order Status** page
cssui-orhl.mvt	CSSUI template for **Lookup Order History** page
cssui-osel.mvt	CSSUI template for **Checkout: Shipping/Payment Selection** page
cssui-ous1.mvt	CSSUI template for **Checkout: Upsell Product** (single) page
cssui-ousm.mvt	CSSUI template for **Checkout: Upsell Product** (multiple) page
cssui-patr.mvt	CSSUI template for **Missing Product Attributes** page
cssui-plmt.mvt	CSSUI template for **Product Limited Stock** page
cssui-plst.mvt	CSSUI template for **Product List** page
cssui-pout.mvt	CSSUI template for **Product Sold Out** page
cssui-prod.mvt	CSSUI template for **Product Display** page
cssui-prpo.mvt	CSSUI template for **Privacy Policy** page
cssui-sarp.mvt	CSSUI template for **Shipping and Return Policy** page
cssui-sfnt.mvt	CSSUI template for **Storefront** page
cssui-smap.mvt	CSSUI template for **Sitemap** page
cssui-srch.mvt	CSSUI template for **Search** page
cssui-uatm.mvt	CSSUI template for **Upsell: Missing Product Attributes** (multiple) page
cssui-uatr.mvt	CSSUI template for **Upsell: Missing Product Attributes** (single) page
email-backorder-notice.mvt	Email template for **Backorder Notice**
email-orderconf-customer.mvt	Email template for **Order Confirmation: Customer**
email-orderconf-merchant.mvt	Email template for **Order Confirmation: Merchant**
email-return-received.mvt	Email template for **Return Received**
email-rma-issued.mvt	Email template for **RMA Issued**

Path/Filename	Description
email-shipment-shipped.mvt	Email template for **Shipment Shipped**
mmui-acad.mvt	MMUI template for **Customer Create** page
mmui-aced.mvt	MMUI template for **Customer Edit** page
mmui-acln.mvt	MMUI template for **Customer Account** page
mmui-afad.mvt	MMUI template for **Affliate Create** page
mmui-afcl.mvt	MMUI template for **Affiliate Login** page
mmui-afed.mvt	MMUI template for **Affiliate Edit** page
mmui-bask.mvt	MMUI template for **Basket Contents** page
mmui-bske.mvt	MMUI template for **Checkout: Basket Empty** page
mmui-ctgy.mvt	MMUI template for **Category Display** page
mmui-invc.mvt	MMUI template for **Invoice** page
mmui-logn.mvt	MMUI template for **Customer Login** page
mmui-mntn.mvt	MMUI template for **Maintenance Mode** page
mmui-ntfd.mvt	MMUI template for **Not Found** page
mmui-ocst.mvt	MMUI template for **Checkout: Customer Information** page
mmui-omin.mvt	MMUI template for **Checkout: Minimum Purchase Required** page
mmui-opay.mvt	MMUI template for **Checkout: Payment Information** page
mmui-oprc.mvt	MMUI template for **Order Already Processed** page
mmui-ordh.mvt	MMUI template for **Order History List** page
mmui-ordl.mvt	MMUI template for **Order: Customer Login** page
mmui-ords.mvt	MMUI template for **Order Status** page
mmui-orhl.mvt	MMUI template for **Lookup Order History** page
mmui-osel.mvt	MMUI template for **Checkout: Shipping/Payment Selection** page
mmui-ous1.mvt	MMUI template for **Checkout: Upsell Product** (single) page
mmui-ousm.mvt	MMUI template for **Checkout: Upsell Product** (multiple) page
mmui-patr.mvt	MMUI template for **Missing Product Attributes** page
mmui-plmt.mvt	MMUI template for **Product Limited Stock** page
mmui-plst.mvt	MMUI template for **Product List** page
mmui-pout.mvt	MMUI template for **Product Sold Out** page
mmui-prod.mvt	MMUI template for **Product Display** page
mmui-sfnt.mvt	MMUI template for **Storefront** page
mmui-smap.mvt	MMUI template for **Sitemap** page
mmui-smap-css_fw.mvt	MMUI template for the **Sitemap** page when the **css_fw** framework has been applied
mmui-srch.mvt	MMUI template for **Search** page
mmui-uatm.mvt	MMUI template for **Upsell: Missing Product Attributes** (multiple) page
mmui-uatr.mvt	MMUI template for **Upsell: Missing Product Attributes** (single) page

The Modules Directory

The following table lists the modules available in the LSK and the features that they implement.

Path/Filename	Features
modules/batch	
stdacct.mv	batchreport
templatebatchreports.mv	batchreport, util, vis_util, data_store, json, clientside, provision_store
modules/component	
cmp-cssui-afae.mv	component, component_prov, designer, skins
cmp-cssui-afflink.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-cssui-attributes.mv	component, component_prov, designer, skins
cmp-cssui-basket.mv	component, component_prov, designer, skins
cmp-cssui-buttons.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-cssui-cattitle.mv	component, data_store, vis_category, provision_store, skins, not_cat
cmp-cssui-cattree.mv	component, designer, data_store, vis_store, vis_category, provision_store, skins, not_cat
cmp-cssui-countryselect.mv	component
cmp-cssui-custfields.mv	component, not_fields, component_prov, designer, skins
cmp-cssui-custlink	component, data_store, vis_store, provision_store, designer, skins
cmp-cssui-hdft.mv	component, data_store, vis_store, provision_store, component_prov, designer, skins
cmp-cssui-head.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-cssui-html.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-cssui-invc-custfields.mv	component, not_fields, component_prov, designer, skins
cmp-cssui-invc-order.mv	component, component_prov, designer, skins
cmp-cssui-links.mv	component, not_seo
cmp-cssui-messages.mv	component
cmp-cssui-navbar.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-cssui-orderlist.mv	component, skins, component_prov
cmp-cssui-pchdft.mv	component, data_store, vis_product, vis_category, provision_store, skins, fields_prod, fields_cat, not_prod, not_cat
cmp-cssui-prodlayo.mv	component, component_prov, designer, skins
cmp-cssui-prodlist.mv	component, component_prov, designer, skins
cmp-cssui-sitemap.mv	component, component_prov, designer, skins
cmp-mmui-uslmltplattr.mv	component
cmp-cssui-vieworder.mv	component, skins, component_prov
cmp-mmui-afae.mv	component, component_prov, designer, skins
cmp-mmui-afflink.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-attributes.mv	component, component_prov, designer, skins
cmp-mmui-basket.mv	component, component_prov, designer, skins
cmp-mmui-body.mv	component, data_store, vis_store, provision_store, designer, skins

Path/Filename	Features
cmp-mmui-buttons.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-cattitle.mv	component, data_store, vis_category, provision_store, skins, not_cat
cmp-mmui-cattree.mv	component, designer, data_store, vis_store, vis_category, provision_store, skins, not_cat
cmp-mmui-colors.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-custfields.mv	component, not_fields, component_prov, designer, skins
cmp-mmui-custlink.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-fonts.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-invc-custfields.mv	component, not_fields, component_prov, designer, skins
cmp-mmui-invc-order.mv	component, component_prov, designer, skins
cmp-mmui-messages.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-navbar.mv	component, data_store, vis_store, provision_store, designer, skins
cmp-mmui-orderlist.mv	component, component_prov, skins
cmp-mmui-pchdft.mv	component, data_store, vis_product, vis_category, provision_store, skins, fields_prod, fields_cat, not_prod, not_cat
cmp-mmui-prodlayo.mv	component, component_prov, designer, skins
cmp-mmui-prodlist.mv	component, component_prov, designer, skins
cmp-mmui-sitemap.mv	component, component_prov, designer, skins
cmp-mmui-uslmltplattr.mv	component
cmp-mmui-vieworder.mv	component, component_prov, skins
cmp-mv-adminorderfields.mv	component
cmp-mv-attributemachine.mv	data_store, component, component_prov, skins
cmp-mv-inventory.mv	component
cmp-mv-messages.mv	component
cmp-mv-paymentfields.mv	component
cmp-mv-payselect.mv	component
cmp-mv-prodctgy-meta.mv	component, data_store, vis_category, vis_product, fields_prod, vis_store, fields_cat, not_prod, not_cat
cmp-mv-shipselect.mv	component
cmp-mv-stateselect.mv	component
cmp-mv-stdcatfields.mv	component
cmp-mv-stdorderfields.mv	component
cmp-mv-stdorderitemfields.mv	component
cmp-mv-stdprodfields.mv	component
cmp-mv-stdreturnfields.mv	component
cmp-mv-stdshipmentfields.mv	component
cmp-mv-stdstorefields.mv	component
cmp-mv-taxfields.mv	component
cmp-mv-uslprodfields.mv	component

Path/Filename	Features
cssui_seo.mv	Include file that is used to generate SEO-friendly shortlinks for CSSUI components. It is included in other modules (such as **cmp-cssui-links**) via **<MvINCLUDE>**.
modules/currency	
eurocur.mv	currency, vis_store, provision_store, data_store
gencurr.mv	currency, vis_store, provision_store, data_store
usmoney.mv	currency
modules/export	
afilexprt.mv	export
attrexp.mv	export
export_include.mv	Helper functions for export modules. Included via **<MvINCLUDE>** into many of the export modules.
flatcat.mv	export
flatcus.mv	export
flatord.mv	export
prodexp.mv	export
modules/fulfill	
customl.mv	fulfill, vis_fulfill, vis_wizard, provision_store, data_store
meremail.mv	fulfill, vis_fulfill, vis_wizard, provision_store, data_store
templateorderemails.mv	fulfill, vis_fulfill, vis_order, not_orderitem, not_ordershpmnt, data_store, json, clientside, provision_store
modules/import	
categoryimport.mv	import
customerimport.mv	import
flatcati.mv	import
flatcusi.mv	import
import_include.mv	Helper functions for import modules. Included via **<MvINCLUDE>** into many of the import modules.
prodimpt.mv	import
productimport.mv	import
provimpt.mv	import
provisioningimport.mv	import
modules/log	
elf.mv	log, vis_log, data_store, provision_store
malf.mv	log, vis_log, data_store, provision_store
modules/payment	
check.mv	payment, vis_wizard
check_detailed.mv	payment, vis_wizard
cod.mv	payment, vis_payment, vis_wizard, provision_store, data_store
mod10.mv	payment, vis_payment, vis_wizard, provision_store, data_store

Path/Filename	Features
modules/report	
geosales.mv	report
productsales.mv	report
sales.mv	report
stats.mv	report
modules/shipping	
baseunit.mv	shipping, vis_shipping, vis_wizard, provision_store, data_store
flatrate.mv	shipping, vis_shipping, vis_wizard, provision_store, data_store
minunit.mv	shipping, vis_shipping, vis_wizard, provision_store, data_store
ptbship.mv	shipping, vis_product, vis_shipping, vis_wizard, provision_store, data_store, not_prod
qship.mv	shipping, vis_shipping, vis_wizard, provision_store, data_store
wtbship.mv	shipping, vis_product, vis_shipping, vis_wizard, provision_store, data_store, not_prod
modules/system	
ex-system.mv	system, vis_system
modules/tax	
canvat.mv	tax, vis_product, vis_store, vis_wizard, provision_store, data_store, not_prod
devat.mv	tax, vis_product, vis_store, vis_wizard, provision_store, data_store, not_prod
shoptax.mv	tax, vis_store, vis_wizard, provision_store, data_store
statetax.mv	tax, vis_store, vis_wizard, provision_store, data_store
vat.mv	tax, vis_product, vis_store, vis_wizard, provision_store, data_store, not_prod
modules/ui	
cssui.mv	storeui, vis_store, data_store, not_seo
mmui.mv	storeui, vis_store, vis_wizard, data_store, not_seo
mmui_stsl.mv	storeselui, data_domain
modules/util	
customfld.mv	util, vis_util, vis_category, not_cat, fields_cat, vis_product, not_prod, fields_prod, vis_-cust, not_cust, fields_cust, data_store, provision_store
productimagecustomfields.mv	util, fields_prod, provision_store

APPENDIX B

API Functions

Miva Merchant Functions

The following table shows available functions and where they can be found. Module API functions are filed under their lowercase Feature designation (e.g., **batchreport**). All others are listed with the location of the source code file.

Function	File Location/Feature	Module
Action_AddMultipleUpsoldProductsToBasket	features/usl/usl_rt.mv	Upsale Runtime
Action_AddProductToBasket	/merchant.mv	Shopping Interface
Action_AddUpsoldProductToBasket	features/usl/usl_rt.mv	Upsale Runtime
Action_AuthorizePayment	/merchant.mv	Shopping Interface
Action_AuthorizePayment_LowLevel	/merchant.mv	Shopping Interface
Action_CalculateShipping	/merchant.mv	Shopping Interface
Action_CalculateTax	/merchant.mv	Shopping Interface
Action_Customer_EmailPassword	features/cus/cus_rt.mv	Customer Runtime
Action_Customer_Insert	features/cus/cus_rt.mv	Customer Runtime
Action_Customer_Login	features/cus/cus_rt.mv	Customer Runtime
Action_Customer_Logout	features/cus/cus_rt.mv	Customer Runtime
Action_Customer_Update	features/cus/cus_rt.mv	Customer Runtime
Action_Customer_Upsell	/merchant.mv	Shopping Interface
Action_PaymentManipulateShipping	/merchant.mv	Shopping Interface
Action_RemoveProductFromBasket	/merchant.mv	Shopping Interface
Action_Save_OrderInformation	/merchant.mv	Shopping Interface

Function	File Location/Feature	Module
Action_UpdateQuantity	/merchant.mv	Shopping Interface
Adjusted_Price	lib/util_public.mv	Helper Functions
Adjusted_Price_LowLevel	lib/util_public.mv	Helper Functions
AFF_Create_Data_Files	features/aff/aff_db.mv	Affiliate System
AFF_Store_Delete	features/aff/aff_db.mv	Affiliate System
Affiliate_Create_Order	features/aff/aff_db.mv	Affiliate System
Affiliate_Delete	features/aff/aff_db.mv	Affiliate System
Affiliate_Delete_ID	features/aff/aff_db.mv	Affiliate System
Affiliate_Insert	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_Code	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_First	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_ID	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_Last	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_Next	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_Previous	features/aff/aff_db.mv	Affiliate System
Affiliate_Load_Session	features/aff/aff_db.mv	Affiliate System
Affiliate_Read	features/aff/aff_db.mv	Affiliate System
Affiliate_Update	features/aff/aff_db.mv	Affiliate System
Affiliate_Update_Balance	features/aff/aff_db.mv	Affiliate System
Affiliate_Update_SessionID	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Delete_Aff	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Delete_ID	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Insert	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Load_ID	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Read	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Read_Affiliate	features/aff/aff_db.mv	Affiliate System
AffiliateEarning_Update	features/aff/aff_db.mv	Affiliate System
AffiliateEarningList_Load_Offset_Affiliate	features/aff/aff_db.mv	Affiliate System
AffiliateEarningList_Load_Offset_Payout	features/aff/aff_db.mv	Affiliate System
AffiliateEarningList_Load_Payout	features/aff/aff_db.mv	Affiliate System
AffiliateEmail_Insert	features/aff/aff_db.mv	Affiliate System
AffiliateEmail_Load	features/aff/aff_db.mv	Affiliate System
AffiliateEmail_Read	features/aff/aff_db.mv	Affiliate System
AffiliateEmail_Update	features/aff/aff_db.mv	Affiliate System
AffiliateList_Load_All	features/aff/aff_db.mv	Affiliate System
AffiliateList_Load_Offset_All	features/aff/aff_db.mv	Affiliate System
AffiliateList_Load_Offset_EmailList_All	features/aff/aff_db.mv	Affiliate System
AffiliateList_Load_Offset_EmailList_Assigned	features/aff/aff_db.mv	Affiliate System

Function	File Location/Feature	Module
AffiliateList_Load_Offset_EmailList_Unassigned	features/aff/aff_db.mv	Affiliate System
AffiliateManage_Insert	features/aff/aff_db.mv	Affiliate System
AffiliateManage_Load	features/aff/aff_db.mv	Affiliate System
AffiliateManage_Read	features/aff/aff_db.mv	Affiliate System
AffiliateManage_Update	features/aff/aff_db.mv	Affiliate System
AffiliateOptions_Insert	features/aff/aff_db.mv	Affiliate System
AffiliateOptions_Load	features/aff/aff_db.mv	Affiliate System
AffiliateOptions_Read	features/aff/aff_db.mv	Affiliate System
AffiliateOptions_Update	features/aff/aff_db.mv	Affiliate System
AffiliatePayout_Delete_ID	features/aff/aff_db.mv	Affiliate System
AffiliatePayout_Insert	features/aff/aff_db.mv	Affiliate System
AffiliatePayout_Load_ID	features/aff/aff_db.mv	Affiliate System
AffiliatePayout_Read	features/aff/aff_db.mv	Affiliate System
AffiliatePayout_Update	features/aff/aff_db.mv	Affiliate System
AffiliatePayoutList_Load_Offset_All	features/aff/aff_db.mv	Affiliate System
AffiliatePayoutList_Load_Offset_Processed	features/aff/aff_db.mv	Affiliate System
AffiliatePayoutList_Load_Offset_Unprocessed	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Delete_Affil_code	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Delete_All	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Delete_All_OlderThan	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Insert	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Load_Session	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Read	features/aff/aff_db.mv	Affiliate System
AffiliateSession_Update_Session	features/aff/aff_db.mv	Affiliate System
AffiliateSessionList_Load_Affil_code	features/aff/aff_db.mv	Affiliate System
AGR_Create_Data_Files	features/agr/agr_db.mv	Availability Group
AGR_Store_Create	features/agr/agr_db.mv	Availability Group
AGR_Store_Delete	features/agr/agr_db.mv	Availability Group
AlphaNumericOnly	lib/util_public.mv	Helper Functions
AppendRuntimeParameter	lib/util_public.mv	Helper Functions
ATT_Create_Data_Files	features/att/att_db.mv	Attribute Template
ATT_Store_Create	features/att/att_db.mv	Attribute Template
ATT_Store_Delete	features/att/att_db.mv	Attribute Template
Attribute_Delete	lib/dbeng/attributes.mv	Database API
Attribute_Delete_All_Product	lib/dbprim/attributes.mv	Database API
Attribute_Delete_ID	lib/dbprim/attributes.mv	Database API
Attribute_Insert	lib/dbeng/attributes.mv	Database API
Attribute_Insert_LowLevel	lib/dbprim/attributes.mv	Database API

Function	File Location/Feature	Module
Attribute_Load_Code	lib/dbprim/attributes.mv	Database API
Attribute_Load_ID	lib/dbprim/attributes.mv	Database API
Attribute_Read	lib/dbprim/attributes.mv	Database API
Attribute_Sort_Swap	lib/dbeng/sort.mv	Database API
Attribute_Update	lib/dbprim/attributes.mv	Database API
Attribute_Update_Default	lib/dbprim/attributes.mv	Database API
Attribute_Update_DisplayOrder	lib/dbprim/attributes.mv	Database API
Attribute_Update_Inventory	lib/dbeng/attributes.mv	Database API
Attribute_Update_Inventory_LowLevel	lib/dbprim/attributes.mv	Database API
AttributeList_Load_Product	lib/dbprim/attributes.mv	Database API
AttributeList_Load_Product_Inventory	lib/dbprim/attributes.mv	Database API
AttributeList_Load_ProductVariant_Possible	lib/dbeng/productvariants.mv	Database API
AttributeList_Load_Template	features/att/att_db.mv	Attribute Template
AttributeTemplate_Copy	features/att/att_db.mv	Attribute Template
AttributeTemplate_Decrement_ReferenceCount	features/att/att_db.mv	Attribute Template
AttributeTemplate_Decrement_ReferenceCount_Product	features/att/att_db.mv	Attribute Template
AttributeTemplate_Delete	features/att/att_db.mv	Attribute Template
AttributeTemplate_Delete_ID	features/att/att_db.mv	Attribute Template
AttributeTemplate_Increment_ReferenceCount	features/att/att_db.mv	Attribute Template
AttributeTemplate_Insert	features/att/att_db.mv	Attribute Template
AttributeTemplate_Load_Code	features/att/att_db.mv	Attribute Template
AttributeTemplate_Load_ID	features/att/att_db.mv	Attribute Template
AttributeTemplate_Read	features/att/att_db.mv	Attribute Template
AttributeTemplate_Update	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Delete	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Delete_Attr	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Delete_ID	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Delete_ID_LowLevel	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Delete_Template	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Insert	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Load_Code	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Load_Product_Code	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Read	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Update	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Update_Inventory	features/att/att_db.mv	Attribute Template
AttributeTemplateAttr_Update_Inventory_LowLevel	features/att/att_db.mv	Attribute Template
AttributeTemplateAttributeList_Update_Offsets	features/att/att_db.mv	Attribute Template
AttributeTemplateAttributeList_Update_Offsets_PastEnd	features/att/att_db.mv	Attribute Template

Function	File Location/Feature	Module
AttributeTemplateAttributeOptionList_Update_Offsets	features/att/att_db.mv	Attribute Template
AttributeTemplateAttributeOptionList_Update_Offsets _PastEnd	features/att/att_db.mv	Attribute Template
AttributeTemplateAttrList_Load_All	features/att/att_db.mv	Attribute Template
AttributeTemplateAttrList_Load_Template	features/att/att_db.mv	Attribute Template
AttributeTemplateList_Load_All	features/att/att_db.mv	Attribute Template
AttributeTemplateList_Load_Offset	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Delete	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Delete_All_Attribute	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Delete_All_Attribute_LowLevel	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Delete_ID	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Delete_ID_LowLevel	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Delete_Template	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Insert	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Load_Code	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Load_ID	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Read	features/att/att_db.mv	Attribute Template
AttributeTemplateOption_Update	features/att/att_db.mv	Attribute Template
AttributeTemplateOptionList_Load_Attribute	features/att/att_db.mv	Attribute Template
AttributeTree_Load_Product	lib/dbeng/productvariants.mv	Database API
AttributeTree_Load_ProductKit	lib/dbeng/productkits.mv	Database API
AvailabilityGroup_Delete	features/agr/agr_db.mv	Availability Group
AvailabilityGroup_Delete_ID	features/agr/agr_db.mv	Availability Group
AvailabilityGroup_Insert	features/agr/agr_db.mv	Availability Group
AvailabilityGroup_Load_ID	features/agr/agr_db.mv	Availability Group
AvailabilityGroup_Load_Name	features/agr/agr_db.mv	Availability Group
AvailabilityGroup_Read	features/agr/agr_db.mv	Availability Group
AvailabilityGroup_Update	features/agr/agr_db.mv	Availability Group
AvailabilityGroupList_Load_All	features/agr/agr_db.mv	Availability Group
AvailabilityGroupList_Load_Offset	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Delete	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Delete_All_AvailabilityGroup	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Delete_All_Category	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Delete_LowLevel	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Insert	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Insert_LowLevel	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Load	features/agr/agr_db.mv	Availability Group
AvailGroupXCategory_Read	features/agr/agr_db.mv	Availability Group

Function	File Location/Feature	Module
AvailGroupXCustomer_Delete	features/agr/agr_db.mv	Availability Group
AvailGroupXCustomer_Delete_All_AvailabilityGroup	features/agr/agr_db.mv	Availability Group
AvailGroupXCustomer_Delete_All_Customer	features/agr/agr_db.mv	Availability Group
AvailGroupXCustomer_Insert	features/agr/agr_db.mv	Availability Group
AvailGroupXCustomer_Load	features/agr/agr_db.mv	Availability Group
AvailGroupXCustomer_Read	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Delete	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Delete_All_AvailabilityGroup	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Delete_All_Product	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Delete_LowLevel	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Insert	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Insert_LowLevel	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Load	features/agr/agr_db.mv	Availability Group
AvailGroupXProduct_Read	features/agr/agr_db.mv	Availability Group
Basket_Clear_Affiliate_Session	lib/dbprim/baskets.mv	Database API
Basket_Clear_Customer_ID	lib/dbprim/baskets.mv	Database API
Basket_Clear_Customer_Information	lib/dbprim/baskets.mv	Database API
Basket_Delete	features/aff/aff_db.mv	Affiliate System
Basket_Delete_All	lib/dbeng/baskets.mv	Database API
Basket_Delete_All_OlderThan	lib/dbeng/baskets.mv	Database API
Basket_Delete_ID	lib/dbprim/baskets.mv	Database API
Basket_Dirty	lib/dbeng/baskets.mv	Database API
Basket_Insert	lib/dbprim/baskets.mv	Database API
Basket_InventoryAdjust_ProductList	lib/dbeng/baskets.mv	Database API
Basket_Load_Session	lib/dbprim/baskets.mv	Database API
Basket_Quantity	lib/dbeng/baskets.mv	Database API
Basket_Quantity_All	lib/dbeng/baskets.mv	Database API
Basket_Quantity_Upsold	lib/dbeng/baskets.mv	Database API
Basket_Read	lib/dbprim/baskets.mv	Database API
Basket_Reset	lib/dbeng/baskets.mv	Database API
Basket_Reset_Basket	lib/dbeng/baskets.mv	Database API
Basket_Reset_Contents	lib/dbeng/baskets.mv	Database API
Basket_SubTotal	lib/dbeng/baskets.mv	Database API
Basket_SubTotal_Taxable	lib/dbeng/baskets.mv	Database API
Basket_Total	lib/dbeng/baskets.mv	Database API
Basket_Update_Affiliate_Session	lib/dbprim/baskets.mv	Database API
Basket_Update_Customer_ID	lib/dbprim/baskets.mv	Database API
Basket_Update_Customer_Information	lib/dbprim/baskets.mv	Database API

Function	File Location/Feature	Module
Basket_Update_LastUpdate	lib/dbprim/baskets.mv	Database API
Basket_Update_Order	lib/dbprim/baskets.mv	Database API
Basket_Update_Shipping	lib/dbprim/baskets.mv	Database API
Basket_Weight	lib/dbeng/baskets.mv	Database API
BasketCharge_Count_Type	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Delete_All_Basket	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Delete_All_Module	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Delete_All_Type	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Delete_Charge	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Insert	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Read	lib/dbprim/basketcharges.mv	Database API
BasketCharge_Total	lib/dbeng/baskets.mv	Database API
BasketChargeList_Load_Basket	lib/dbprim/basketcharges.mv	Database API
BasketChargeList_Load_Type	lib/dbprim/basketcharges.mv	Database API
BasketItem_Delete_All_Basket	lib/dbprim/basketitems.mv	Database API
BasketItem_Delete_Line	lib/dbprim/basketitems.mv	Database API
BasketItem_Increment_Quantity	lib/dbprim/basketitems.mv	Database API
BasketItem_Insert	lib/dbprim/basketitems.mv	Database API
BasketItem_Load_Line	lib/dbprim/basketitems.mv	Database API
BasketItem_Read	lib/dbprim/basketitems.mv	Database API
BasketItem_Total	lib/dbeng/baskets.mv	Database API
BasketItem_Weight	lib/dbeng/baskets.mv	Database API
BasketItemList_Load_Basket	lib/dbprim/basketitems.mv	Database API
BasketItemList_Load_Basket_Product	lib/dbprim/basketitems.mv	Database API
BasketItemList_Load_Upsold	lib/dbprim/basketitems.mv	Database API
BasketOption_Delete_All_Attribute	lib/dbprim/basketoptions.mv	Database API
BasketOption_Delete_All_Basket	lib/dbprim/basketoptions.mv	Database API
BasketOption_Delete_All_Line	lib/dbprim/basketoptions.mv	Database API
BasketOption_Insert	lib/dbprim/basketoptions.mv	Database API
BasketOption_Read	lib/dbprim/basketoptions.mv	Database API
BasketOption_Total	lib/dbeng/baskets.mv	Database API
BasketOption_Weight	lib/dbeng/baskets.mv	Database API
BasketOptionList_Load_Line	lib/dbprim/basketoptions.mv	Database API
Batch_Create	lib/dbeng/batches.mv	Database API
Batch_Create_OrderList	lib/dbeng/batches.mv	Database API
Batch_Delete_ID	lib/dbprim/batches.mv	Database API
Batch_Insert	lib/dbprim/batches.mv	Database API
Batch_Load_ID	lib/dbprim/batches.mv	Database API

Function	File Location/Feature	Module
Batch_Read	lib/dbprim/batches.mv	Database API
BatchList_Load_All	lib/dbprim/batches.mv	Database API
BatchList_Load_Closed	lib/dbprim/batches.mv	Database API
BatchList_Load_Offset	lib/dbprim/batches.mv	Database API
BatchReportModule_Order_Reports	Feature batchreport	Module API
BatchReportModule_Report	Feature batchreport	Module API
BatchReportModule_Run_OrderList	Feature batchreport	Module API
BatchReportModule_Run_ShipmentList	Feature batchreport	Module API
BatchReportModule_Shipment_Reports	Feature batchreport	Module API
BeginContent	admin/ui.mv	Administrative UI
BeginScreen	admin/ui.mv	Administrative UI
BeginScreen_End	admin/ui.mv	Administrative UI
BeginScreen_Start	admin/ui.mv	Administrative UI
BestSellerList_Load_Offset	features/sta/sta_db.mv	Statistics
BoxPackingModule_Box_Delete	Feature boxpacking	Module API
BoxPackingModule_Box_Field	Feature boxpacking	Module API
BoxPackingModule_Box_Fields	Feature boxpacking	Module API
BoxPackingModule_Box_Insert	Feature boxpacking	Module API
BoxPackingModule_Box_Invalid	Feature boxpacking	Module API
BoxPackingModule_Box_Prompt	Feature boxpacking	Module API
BoxPackingModule_Box_Provision	Feature boxpacking	Module API
BoxPackingModule_Box_Update	Feature boxpacking	Module API
BoxPackingModule_Box_Validate	Feature boxpacking	Module API
BoxPackingModule_Box_Items	Feature boxpacking	Module API
Build_Attribute_Hash	lib/util_public.mv	Helper Functions
Category_Decrement_AvailabilityGroupCount	lib/dbprim/categories.mv	Database API
Category_Decrement_AvailabilityGroupCount_All	lib/dbeng/categories.mv	Database API
Category_Delete	lib/dbeng/categories.mv	Database API
Category_Delete_ID	lib/dbprim/categories.mv	Database API
Category_Increment_AvailabilityGroupCount	lib/dbprim/categories.mv	Database API
Category_Insert	lib/dbprim/categories.mv	Database API
Category_Load_Code	lib/dbprim/categories.mv	Database API
Category_Load_DisplayOrder	lib/dbprim/categories.mv	Database API
Category_Load_First	lib/dbeng/categories.mv	Database API
Category_Load_ID	lib/dbprim/categories.mv	Database API
Category_Load_Last	lib/dbeng/categories.mv	Database API
Category_Load_Next	lib/dbeng/categories.mv	Database API
Category_Load_Previous	lib/dbeng/categories.mv	Database API

Function	File Location/Feature	Module
Category_Read	lib/dbprim/categories.mv	Database API
Category_Sort_All	lib/dbeng/sort.mv	Database API
Category_Sort_Swap	lib/dbeng/sort.mv	Database API
Category_Update	lib/dbprim/categories.mv	Database API
Category_Update_DisplayOrder	lib/dbprim/categories.mv	Database API
Category_Update_Parent_All	lib/dbprim/categories.mv	Database API
CategoryList_Load_All	lib/dbprim/categories.mv	Database API
CategoryList_Load_Offset	lib/dbeng/categories.mv	Database API
CategoryList_Load_Offset_AvailabilityGroup_All	lib/dbeng/categories.mv	Database API
CategoryList_Load_Offset_AvailabilityGroup_Assigned	lib/dbeng/categories.mv	Database API
CategoryList_Load_Offset_AvailabilityGroup _Unassigned	lib/dbeng/categories.mv	Database API
CategoryList_Load_Offset_Product_All	lib/dbeng/categories.mv	Database API
CategoryList_Load_Offset_Product_Assigned	lib/dbeng/categories.mv	Database API
CategoryList_Load_Offset_Product_Unassigned	lib/dbeng/categories.mv	Database API
CategoryList_Load_Parent	lib/dbprim/categories.mv	Database API
CategoryList_Update_Offsets	lib/dbeng/categories.mv	Database API
CategoryList_Update_Offsets_PastEnd	lib/dbeng/categories.mv	Database API
CategoryXProduct_Delete	lib/dbeng/categories.mv	Database API
CategoryXProduct_Delete_All_Category	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Delete_All_Product	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Delete_LowLevel	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Insert	lib/dbeng/categories.mv	Database API
CategoryXProduct_Insert_LowLevel	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Load	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Load_Category	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Load_DisplayOrder	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Read	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Sort_All	lib/dbeng/sort.mv	Database API
CategoryXProduct_Sort_Swap	lib/dbeng/sort.mv	Database API
CategoryXProduct_Update_DisplayOrder	lib/dbprim/categoryxproduct.mv	Database API
CategoryXProduct_Update_Offsets	lib/dbeng/categories.mv	Database API
CategoryXProduct_Update_Offsets_PastEnd	lib/dbeng/categories.mv	Database API
ClearCookie	lib/util_public.mv	Helper Functions
ComponentModule_Content	Feature component	Module API
ComponentModule_Defaults	Feature component	Module API
ComponentModule_Initialize	Feature component	Module API
ComponentModule_Page_Assign	Feature component	Module API

Function	File Location/Feature	Module
ComponentModule_Page_Unassign	Feature component	Module API
ComponentModule_Prerender	Feature component	Module API
ComponentModule_Provision	Feature component_prov	Module API
ComponentModule_Render_End	Feature component	Module API
ComponentModule_Render_Head	Feature component	Module API
ComponentModule_Render_Start	Feature component	Module API
ComponentModule_Tabs	Feature component	Module API
ComponentModule_Update	Feature component	Module API
ComponentModule_Validate	Feature component	Module API
Country_Delete	lib/dbprim/countries.mv	Database API
Country_Insert	lib/dbprim/countries.mv	Database API
Country_Load_Alpha	lib/dbprim/countries.mv	Database API
Country_Load_ID	lib/dbprim/countries.mv	Database API
Country_Load_Iso_code	lib/dbprim/countries.mv	Database API
Country_Load_Name	lib/dbprim/countries.mv	Database API
Country_Read	lib/dbprim/countries.mv	Database API
Country_Select	lib/util_public.mv	Helper Functions
Country_Update	lib/dbprim/countries.mv	Database API
CountryList_Load_All	lib/dbprim/countries.mv	Database API
CountryList_Load_Alpha	lib/dbprim/countries.mv	Database API
CountryList_Load_Offset_All	lib/dbeng/countries.mv	Database API
CountryList_Load_Offset_Store_All	lib/dbeng/countries.mv	Database API
CountryList_Load_Offset_Store_Assigned	lib/dbeng/countries.mv	Database API
CountryList_Load_Offset_Store_Unassigned	lib/dbeng/countries.mv	Database API
Create_Data_Files	features/usl/usl_db.mv	Upsale Database
CreateDataFiles	lib/dbeng/create_domain.mv	Database API
CurrencyModule_AddFormatPlainText	Feature currency	Module API
CurrencyModule_AddFormatPlainTextShort	Feature currency	Module API
CurrencyModule_AddFormatting	Feature currency	Module API
CurrencyModule_Output_CurrencyFormat_JavaScript	Feature currency	Module API
CUS_Create_Data_Files	features/cus/cus_db.mv	Customer Database
CUS_Store_Create	features/cus/cus_db.mv	Customer Database
CUS_Store_Delete	features/cus/cus_db.mv	Customer Database
Customer_Decrement_PriceGroupCount	features/cus/cus_db.mv	Customer Database
Customer_Decrement_PriceGroupCount_All	features/cus/cus_db.mv	Customer Database
Customer_Delete	features/cus/cus_db.mv	Customer Database
Customer_Delete_ID	features/cus/cus_db.mv	Customer Database
Customer_Increment_PriceGroupCount	features/cus/cus_db.mv	Customer Database

Function	File Location/Feature	Module
Customer_Insert	features/cus/cus_db.mv	Customer Database
Customer_Load_First	features/cus/cus_db.mv	Customer Database
Customer_Load_ID	features/cus/cus_db.mv	Customer Database
Customer_Load_Last	features/cus/cus_db.mv	Customer Database
Customer_Load_Login	features/cus/cus_db.mv	Customer Database
Customer_Load_Next	features/cus/cus_db.mv	Customer Database
Customer_Load_Previous	features/cus/cus_db.mv	Customer Database
Customer_Read	features/cus/cus_db.mv	Customer Database
Customer_Update	features/cus/cus_db.mv	Customer Database
CustomerEmail_Insert	features/cus/cus_db.mv	Customer Database
CustomerEmail_Load	features/cus/cus_db.mv	Customer Database
CustomerEmail_Read	features/cus/cus_db.mv	Customer Database
CustomerEmail_Update	features/cus/cus_db.mv	Customer Database
CustomerList_Load_All	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_AvailabilityGroup_All	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_AvailabilityGroup_Assigned	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_AvailabilityGroup _Unassigned	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_EmailList_All	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_EmailList_Assigned	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_EmailList_Unassigned	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_PriceGroup_All	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_PriceGroup_Assigned	features/cus/cus_db.mv	Customer Database
CustomerList_Load_Offset_PriceGroup_Unassigned	features/cus/cus_db.mv	Customer Database
Customer_Validate	features/cus/cus_rt.mv	Customer Runtime
Data_Entry_Error	lib/util_public.mv	Helper Functions
DB_Close_PrivateKey	lib/dbapi.mv	DB Compatibility Layer
DB_Configuration_Save	lib/dbapi.mv	DB Compatibility Layer
DB_Open	lib/dbapi.mv	DB Compatibility Layer
DB_Open_Parameters	lib/dbapi.mv	DB Compatibility Layer
DB_Open_PrivateKey	lib/dbapi.mv	DB Compatibility Layer
DB_Open_PrivateKey_Parameters	lib/dbapi.mv	DB Compatibility Layer
DB_Supported_List	lib/dbapi.mv	DB Compatibility Layer
DB_Supported_Name	lib/dbapi.mv	DB Compatibility Layer
Decrypt_Order	lib/crypto_public.mv	Encrypt/Decrypt
Decrypt_OrderPayment	lib/crypto_public.mv	Encrypt/Decrypt
Decrypt_Payment	lib/crypto_public.mv	Encrypt/Decrypt

Function	File Location/Feature	Module
DesignerComponentModule_Export	Feature designer	Module API
DesignerComponentModule_ImportProvisionLines	Feature designer	Module API
DetermineSessionID	lib/util_public.mv	Helper Functions
DiscountModule_Capabilities	Feature discount	Module API
DiscountModule_Discount_Basket	Feature discount	Module API
DiscountModule_Discount_Items	Feature discount	Module API
DiscountModule_Discount_PreItems	Feature discount	Module API
DiscountModule_Discount_Shipping	Feature discount	Module API
DiscountModule_Discount_ShippingMethodList	Feature discount	Module API
DiscountModule_Field	Feature discount	Module API
DiscountModule_Fields	Feature discount	Module API
DiscountModule_Invalid	Feature discount	Module API
DiscountModule_Item_Eligible	Feature discount	Module API
DiscountModule_PriceGroup_Delete	Feature discount	Module API
DiscountModule_Prompt	Feature discount	Module API
DiscountModule_Provision_Settings	Feature discount	Module API
DiscountModule_Update	Feature discount	Module API
DiscountModule_Validate	Feature discount	Module API
Domain_Delete	lib/dbeng/delete_domain.mv	Database API
Domain_Insert	lib/dbprim/domain.mv	Database API
Domain_Load	lib/dbprim/domain.mv	Database API
Domain_Read	lib/dbprim/domain.mv	Database API
Domain_Update	lib/dbprim/domain.mv	Database API
DomainKey_Generate	lib/dbeng/keys.mv	Database API
DomainKey_Insert	lib/dbprim/domainkeys.mv	Database API
DomainKey_Load_Type	lib/dbprim/domainkeys.mv	Database API
DomainKey_Read	lib/dbprim/domainkeys.mv	Database API
DomainKey_Update	lib/dbeng/keys.mv	Database API
Draw_FieldSet_Close	admin/ui.mv	Administrative UI
Draw_FieldSet_Open	admin/ui.mv	Administrative UI
Draw_ProgressBar	lib/util_public.mv	Helper Functions
Draw_ProgressBar_Begin	lib/util_public.mv	Helper Functions
Draw_ProgressBar_End	lib/util_public.mv	Helper Functions
Draw_ProgressBar_Update	lib/util_public.mv	Helper Functions
DrawButton_Reset	admin/wizardui.mv	Wizard UI
DrawButtons	admin/ui.mv	Administrative UI
DrawButtons_NextPrevious	admin/ui.mv	Administrative UI
DrawCheck	admin/ui.mv	Administrative UI

Function	File Location/Feature	Module
DrawCheckbox	lib/util_public.mv	Helper Functions
DrawCheckboxModAlert	lib/util_public.mv	Helper Functions
DrawCheck_Active	admin/ui.mv	Administrative UI
DrawCheck_Administrator	admin/ui.mv	Administrative UI
DrawCheck_Assigned	admin/ui.mv	Administrative UI
DrawCheck_CreateUsers	admin/ui.mv	Administrative UI
DrawCheck_Default	admin/ui.mv	Administrative UI
DrawCheck_Method	admin/ui.mv	Administrative UI
DrawCheck_Required	admin/ui.mv	Administrative UI
DrawCheck_Taxable	admin/ui.mv	Administrative UI
DrawImgButton_Active	admin/ui.mv	Administrative UI
DrawImgButton_Add	admin/ui.mv	Administrative UI
DrawImgButton_Add_Adjustment	admin/ui.mv	Administrative UI
DrawImgButton_Add_Affiliate	admin/ui.mv	Administrative UI
DrawImgButton_Add_Attribute	admin/ui.mv	Administrative UI
DrawImgButton_Add_Category	admin/ui.mv	Administrative UI
DrawImgButton_Add_Country	admin/ui.mv	Administrative UI
DrawImgButton_Add_CryptoKey	admin/ui.mv	Administrative UI
DrawImgButton_Add_Customer	admin/ui.mv	Administrative UI
DrawImgButton_Add_Extension	admin/ui.mv	Administrative UI
DrawImgButton_Add_Group	admin/ui.mv	Administrative UI
DrawImgButton_Add_Item	admin/ui.mv	Administrative UI
DrawImgButton_Add_List	admin/ui.mv	Administrative UI
DrawImgButton_Add_Method	admin/ui.mv	Administrative UI
DrawImgButton_Add_Module	admin/ui.mv	Administrative UI
DrawImgButton_Add_Option	admin/ui.mv	Administrative UI
DrawImgButton_Add_Page	admin/ui.mv	Administrative UI
DrawImgButton_Add_Payout	admin/ui.mv	Administrative UI
DrawImgButton_Add_Product	admin/ui.mv	Administrative UI
DrawImgButton_Add_Province	admin/ui.mv	Administrative UI
DrawImgButton_Add_Range	admin/ui.mv	Administrative UI
DrawImgButton_Add_Rate	admin/ui.mv	Administrative UI
DrawImgButton_Add_State	admin/ui.mv	Administrative UI
DrawImgButton_Add_Template	admin/ui.mv	Administrative UI
DrawImgButton_Add_Upsale	admin/ui.mv	Administrative UI
DrawImgButton_Add_User	admin/ui.mv	Administrative UI
DrawImgButton_All	admin/ui.mv	Administrative UI
DrawImgButton_Assigned	admin/ui.mv	Administrative UI

Function	File Location/Feature	Module
DrawImgButton_Available	admin/ui.mv	Administrative UI
DrawImgButton_Edit	admin/ui.mv	Administrative UI
DrawImgButton_EditHere	admin/ui.mv	Administrative UI
DrawImgButton_Export_Page	admin/ui.mv	Administrative UI
DrawImgButton_Help	admin/ui.mv	Administrative UI
DrawImgButton_Import_Page	admin/ui.mv	Administrative UI
DrawImgButton_Links	admin/ui.mv	Administrative UI
DrawImgButton_Lock	admin/ui.mv	Administrative UI
DrawImgButton_Lookup	admin/ui.mv	Administrative UI
DrawImgButton_NewX	admin/ui.mv	Administrative UI
DrawImgButton_Next	admin/ui.mv	Administrative UI
DrawImgButton_Previous	admin/ui.mv	Administrative UI
DrawImgButton_Refresh	admin/ui.mv	Administrative UI
DrawImgButton_Search	admin/ui.mv	Administrative UI
DrawImgButton_Select	admin/ui.mv	Administrative UI
DrawImgButton_Unassigned	admin/ui.mv	Administrative UI
DrawImgButton_Unbatched	admin/ui.mv	Administrative UI
DrawImgButton_Uncategorized	admin/ui.mv	Administrative UI
DrawImgButton_Upload	admin/ui.mv	Administrative UI
DrawImgButtonNew_All	admin/ui.mv	Administrative UI
DrawOption	lib/util_public.mv	Helper Functions
DrawOption_SelectOne	lib/util_public.mv	Helper Functions
DrawRadio	lib/util_public.mv	Helper Functions
DrawTabs	admin/ui.mv	Administrative UI
DrawTemplateTextArea	lib/util_public.mv	Helper Functions
DrawTemplateTextArea_SetModified	lib/util_public.mv	Helper Functions
Email_Add_AngleBrackets	lib/util_public.mv	Helper Functions
Email_Validate	lib/util_public.mv	Helper Functions
Encrypt_Payment	lib/crypto_public.mv	Encrypt/Decrypt
Encryption_Available	lib/crypto_public.mv	Encrypt/Decrypt
Encryption_Available	lib/util_public.mv	Helper Functions
Encryption_Decrement_ReferenceCount	lib/dbprim/encryption.mv	Database API
Encryption_Delete_ID	lib/dbprim/encryption.mv	Database API
Encryption_Increment_ReferenceCount	lib/dbprim/encryption.mv	Database API
Encryption_Insert	lib/dbprim/encryption.mv	Database API
Encryption_Key_Create	lib/crypto_public.mv	Encrypt/Decrypt
Encryption_Load_ID	lib/dbprim/encryption.mv	Database API
Encryption_Load_Prompt	lib/dbprim/encryption.mv	Database API

Function	File Location/Feature	Module
Encryption_Read	lib/dbprim/encryption.mv	Database API
Encryption_Update	lib/dbprim/encryption.mv	Database API
EncryptionList_Load_All	lib/dbprim/encryption.mv	Database API
EncryptionList_Load_Offset	lib/dbeng/encryption.mv	Database API
EncryptionList_Load_Offset_Order	lib/dbeng/encryption.mv	Database API
EndContent	admin/ui.mv	Administrative UI
EndScreen	admin/ui.mv	Administrative UI
EOF_Return	lib/dbeng/util.mv	Database API
Error	features/sbm/sbm_db.mv	Miva Submit
Error	lib/util_public.mv	Helper Functions
Error_Is_EOF	lib/dbeng/util.mv	Database API
Error_ListLoad_EOF	lib/dbeng/util.mv	Database API
Error_Load_EOF	lib/dbeng/util.mv	Database API
Error_Store_Closed	/merchant.mv	Shopping Interface
escape	features/tui/tui_mgr.mv	Template Manager
ExportModule_Export	Feature export	Module API
ExportModule_Screen	Feature export	Module API
ExportModule_Validate	Feature export	Module API
FeedModule_Capabilities	Feature feed	Module API
FeedModule_Delete	Feature feed	Module API
FeeModule_Field	Feature feed	Module API
FeedModule_Fields	Feature feed	Module API
FeedModule_Invalid	Feature feed	Module API
FeedModule_Output	Feature feed	Module API
FeedModule_Output_Custom	Feature feed	Module API
FeedModule_Prompt	Feature feed	Module API
FeedModule_Provision_Settings	Feature feed	Module API
FeedModule_Update	Feature feed	Module API
FeedModule_Validate	Feature feed	Module API
FieldError	admin/ui.mv	Administrative UI
FirstSparseArrayElement	lib/util_public.mv	Helper Functions
Format_Address	lib/util_public.mv	Helper Functions
Format_Date	lib/util_public.mv	Helper Functions
Format_RFC1123_DateTime	lib/util_public.mv	Helper Functions
Format_Time	lib/util_public.mv	Helper Functions
Format_Time_Short	lib/util_public.mv	Helper Functions
FulfillmentModule_ProcessOrder	Feature fulfill	Module API
GenerateAttachmentMIME	lib/util_public.mv	Helper Functions

Function	File Location/Feature	Module
GenerateBodyMIME	lib/util_public.mv	Helper Functions
GenerateBoundary	lib/util_public.mv	Helper Functions
Help_Button	admin/wizardui.mv	Wizard UI
Hex_Decode	lib/util_public.mv	Helper Functions
Hex_Encode	lib/util_public.mv	Helper Functions
HTML_Format_Marks	lib/util_public.mv	Helper Functions
ImportModule_Capabilities	Feature import	Module API
ImportModule_Delimited_Columns	Feature import	Module API
ImportModule_Delimited_Import_Begin	Feature import	Module API
ImportModule_Delimited_Import_End	Feature import	Module API
ImportModule_Delimited_Import_Record	Feature import	Module API
ImportModule_Import	Feature import	Module API
ImportModule_Persistent_Field	Feature import	Module API
ImportModule_Persistent_Fields	Feature import	Module API
ImportModule_Persistent_Invalid	Feature import	Module API
ImportModule_Persistent_Prompt	Feature import	Module API
ImportModule_Persistent_Provision	Feature import	Module API
ImportModule_Persistent_StatusFields	Feature import	Module API
ImportModule_Persistent_Update	Feature import	Module API
ImportModule_Persistent_Validate	Feature import	Module API
ImportModule_Raw_Import_Begin	Feature import	Module API
ImportModule_Raw_Import_Deserialize	Feature import	Module API
ImportModule_Raw_Import_End	Feature import	Module API
ImportModule_Raw_Import_Record	Feature import	Module API
ImportModule_Raw_Import_Serialize	Feature import	Module API
ImportModule_Screen	Feature import	Module API
ImportModule_Validate	Feature import	Module API
INV_Create_Data_Files	features/inv/inv_db.mv	Inventory Database
INV_Store_Create	features/inv/inv_db.mv	Inventory Database
INV_Store_Delete	features/inv/inv_db.mv	Inventory Database
Inventory_Adjust	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Adjust_LowLevel	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Adjust_Variant	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Adjust_VariantPricing	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Check_Available	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Check_Available_LowLevel	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Check_Available_Throw_Inventory_Limited	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Check_Available_Throw_Inventory_Out	features/inv/inv_rt.mv	Inventory Runtime

Function	File Location/Feature	Module
Inventory_Check_Available_Variant	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Load_Variant	features/inv/inv_rt.mv	Inventory Runtime
Inventory_Tokenize_Message	features/inv/inv_rt.mv	Inventory Runtime
InventoryDefaultProductSettings	features/inv/inv_db.mv	Inventory Database
InventoryProductCount	features/inv/inv_db.mv	Inventory Database
InventoryProductCount_Adjust	features/inv/inv_db.mv	Inventory Database
InventoryProductCount_Delete	features/inv/inv_db.mv	Inventory Database
InventoryProductCount_Insert	features/inv/inv_db.mv	Inventory Database
InventoryProductCount_Read	features/inv/inv_db.mv	Inventory Database
InventoryProductCount_Update	features/inv/inv_db.mv	Inventory Database
InventoryProductSettings_Delete	features/inv/inv_db.mv	Inventory Database
InventoryProductSettings_Insert	features/inv/inv_db.mv	Inventory Database
InventoryProductSettings_Load	features/inv/inv_db.mv	Inventory Database
InventoryProductSettings_Read	features/inv/inv_db.mv	Inventory Database
InventoryProductSettings_Update	features/inv/inv_db.mv	Inventory Database
InventorySettings_Insert	features/inv/inv_db.mv	Inventory Database
InventorySettings_Load	features/inv/inv_db.mv	Inventory Database
InventorySettings_Read	features/inv/inv_db.mv	Inventory Database
InventorySettings_Update	features/inv/inv_db.mv	Inventory Database
Item_Decrement_ReferenceCount	features/tui/tui_db.mv	Template Database
Item_Delete_ID	features/tui/tui_db.mv	Template Database
Item_Increment_ReferenceCount	features/tui/tui_db.mv	Template Database
Item_Insert	features/tui/tui_db.mv	Template Database
Item_Load_Code	features/tui/tui_db.mv	Template Database
Item_Load_ID	features/tui/tui_db.mv	Template Database
Item_Load_Module	features/tui/tui_db.mv	Template Database
Item_Load_PageList	features/tui/tui_db.mv	Template Database
Item_Read	features/tui/tui_db.mv	Template Database
Item_Update	features/tui/tui_db.mv	Template Database
ItemExtension_Delete_All_Item	features/tui/tui_db.mv	Template Database
ItemExtension_Delete_All_Module	features/tui/tui_db.mv	Template Database
ItemExtension_Delete_ID	features/tui/tui_db.mv	Template Database
ItemExtension_Insert	features/tui/tui_db.mv	Template Database
ItemExtension_Load_ID	features/tui/tui_db.mv	Template Database
ItemExtension_Load_ItemModule	features/tui/tui_db.mv	Template Database
ItemExtension_Read	features/tui/tui_db.mv	Template Database
ItemExtension_Update_Order	features/tui/tui_db.mv	Template Database
ItemExtensionList_Load_Item	features/tui/tui_db.mv	Template Database

Function	File Location/Feature	Module
ItemExtensionModuleList_Load_Page_Render	features/tui/tui_db.mv	Template Database
ItemList_Load_All	features/tui/tui_db.mv	Template Database
ItemList_Load_Codes	features/tui/tui_db.mv	Template Database
ItemList_Load_Module	features/tui/tui_db.mv	Template Database
ItemList_Load_Offset	features/tui/tui_db.mv	Template Database
ItemList_Load_Offset_Page_All	features/tui/tui_db.mv	Template Database
ItemList_Load_Offset_Page_Assigned	features/tui/tui_db.mv	Template Database
ItemList_Load_Offset_Page_Unassigned	features/tui/tui_db.mv	Template Database
ItemModuleList_Load_Page	features/tui/tui_db.mv	Template Database
ItemModuleList_Load_Page_Active	features/tui/tui_db.mv	Template Database
ItemModuleList_Load_Page_Render	features/tui/tui_db.mv	Template Database
ItemModuleList_Load_Page_Store	features/tui/tui_db.mv	Template Database
ItemModuleList_Load_Page_Store_Active	features/tui/tui_db.mv	Template Database
JavaScript_Set_A_Variable	admin/ui.mv	Administrative UI
JavaScript_SetVariables	admin/ui.mv	Administrative UI
JavaScriptEncode	admin/ui.mv	Administrative UI
JavaScriptEncode_NoEntities	admin/ui.mv	Administrative UI
JSON_Module_Upload_ProcessFileUpload	feature json_upload	Module API
JSON_Module_Upload_ValidateFileUpload	feature json_upload	Module API
LaunchPadButton_Insert	lib/dbprim/launchpad.mv	Database API
LaunchPadButton_Load_ID	lib/dbprim/launchpad.mv	Database API
LaunchPadButton_Read	lib/dbprim/launchpad.mv	Database API
LaunchPadButton_Update	lib/dbprim/launchpad.mv	Database API
LaunchPadButtonList_Load_All	lib/dbprim/launchpad.mv	Database API
LaunchPadButtonList_Load_Default	lib/dbprim/launchpad.mv	Database API
LaunchPadConfig_Insert	lib/dbprim/launchpad.mv	Database API
LaunchPadConfig_Load	lib/dbprim/launchpad.mv	Database API
LaunchPadConfig_Read	lib/dbprim/launchpad.mv	Database API
LaunchPadConfig_Update	lib/dbprim/launchpad.mv	Database API
ListLoad_EOF_Return	lib/dbeng/util.mv	Database API
LogModule_Action	Feature log	Module API
LogModule_Screen	Feature log	Module API
LogModule_UIException	Feature log	Module API
ManagedTemplate_Delete_History	features/tui/tui_db.mv	Template Database
ManagedTemplate_Delete_ID	features/tui/tui_db.mv	Template Database
ManagedTemplate_Insert	features/tui/tui_db.mv	Template Database
ManagedTemplate_Load_Filename	features/tui/tui_db.mv	Template Database
ManagedTemplate_Load_ID	features/tui/tui_db.mv	Template Database

Function	File Location/Feature	Module
ManagedTemplate_Read	features/tui/tui_db.mv	Template Database
ManagedTemplate_Update	features/tui/tui_db.mv	Template Database
ManagedTemplateList_Load_All	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Delete_All_Template	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Delete_ID	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Insert	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Load_ID	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Load_Template_Current	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Read	features/tui/tui_db.mv	Template Database
ManagedTemplateVersion_Update	features/tui/tui_db.mv	Template Database
ManagedTemplateVersionList_Load_Template	features/tui/tui_db.mv	Template Database
Merchant	/merchant.mv	Shopping Interface
Merchant_Actions	/merchant.mv	Shopping Interface
Merchant_Screens	/merchant.mv	Shopping Interface
Message_Error	lib/util_public.mv	Helper Functions
Message_Information	lib/util_public.mv	Helper Functions
Module_Affiliate_BatchEdit_Content	Feature vis_affilbe	Module API
Module_Affiliate_BatchEdit_Delete	Feature vis_affilbe	Module API
Module_Affiliate_BatchEdit_Head	Feature vis_affilbe	Module API
Module_Affiliate_BatchEdit_Tabs	Feature vis_affilbe	Module API
Module_Affiliate_BatchEdit_Update	Feature vis_affilbe	Module API
Module_Affiliate_BatchEdit_Validate	Feature vis_affilbe	Module API
Module_Affiliate_Content	Feature vis_affil	Module API
Module_Affiliate_Delete	Feature vis_affil	Module API
Module_Affiliate_Head	Feature vis_affil	Module API
Module_Affiliate_Insert	Feature vis_affil	Module API
Module_Affiliate_Tabs	Feature vis_affil	Module API
Module_Affiliate_Update	Feature vis_affil	Module API
Module_Affiliate_Validate	Feature vis_affil	Module API
Module_Box_Field_Capabilities	Feature fields_box	Module API
Module_Box_Field_Name	Feature fields_box	Module API
Module_Box_Field_Query	Feature fields_box	Module API
Module_Box_Field_Query_OrderBy	Feature fields_box	Module API
Module_Box_Field_Query_OrderBy_ LoadIndexRecord	Feature fields_box	Module API
Module_Box_Field_Query_Search	Feature fields_box	Module API
Module_Box_Field_Query_Value	Feature fields_box	Module API
Module_Box_Field_Value	Feature fields_box	Module API
Module_Box_Field_Value_Array	Feature fields_box	Module API

Function	File Location/Feature	Module
Module_Box_Fields	Feature fields_box	Module API
Module_Box_Set_Field	Feature fields_box	Module API
Module_Box_Set_Field_Array	Feature fields_box	Module API
Module_Category_Set_Field	Feature fields_cat	Module API
Module_Category_Set_Field_Array	Feature fields_cat	Module API
Module_Category_Fields_Mapped	Feature fields_cat_map	Module API
Module_Category_BatchEdit_Content	Feature vis_categorybe	Module API
Module_Category_BatchEdit_Delete	Feature vis_categorybe	Module API
Module_Category_BatchEdit_Head	Feature vis_categorybe	Module API
Module_Category_BatchEdit_Tabs	Feature vis_categorybe	Module API
Module_Category_BatchEdit_Update	Feature vis_categorybe	Module API
Module_Category_BatchEdit_Validate	Feature vis_categorybe	Module API
Module_Category_Content	Feature vis_category	Module API
Module_Category_Delete	Feature vis_category	Module API
Module_Category_Field_Capabilities	Feature fields_cat	Module API
Module_Category_Field_Name	Feature fields_cat	Module API
Module_Category_Field_Query	Feature fields_cat	Module API
Module_Category_Field_Query_OrderBy	Feature fields_cat	Module API
Module_Category_Field_Query_OrderBy_ LoadIndexRecord	Feature fields_cat	Module API
Module_Category_Field_Query_Search	Feature fields_cat	Module API
Module_Category_Field_Query_Value	Feature fields_cat	Module API
Module_Category_Field_Value	Feature fields_cat	Module API
Module_Category_Field_Value_Array	Feature fields_cat	Module API
Module_Category_Fields	Feature fields_cat	Module API
Module_Category_Fields_Mapped	Feature fields_cat_map	Module API
Module_Category_Head	Feature vis_category	Module API
Module_Category_Insert	Feature vis_category	Module API
Module_Category_Set_Field	Feature fields_cat	Module API
Module_Category_Set_Field_Array	Feature fields_cat	Module API
Module_Category_Tabs	Feature vis_category	Module API
Module_Category_Update	Feature vis_category	Module API
Module_Category_Validate	Feature vis_category	Module API
Module_Cleanup_Store	Feature cleanup_store	Module API
Module_Clientside	Feature clientside	Module API
Module_Customer_BatchEdit_Content	Feature vis_custbe	Module API
Module_Customer_BatchEdit_Delete	Feature vis_custbe	Module API
Module_Customer_BatchEdit_Head	Feature vis_custbe	Module API

Function	File Location/Feature	Module
Module_Customer_BatchEdit_Tabs	Feature vis_custbe	Module API
Module_Customer_BatchEdit_Update	Feature vis_custbe	Module API
Module_Customer_BatchEdit_Validate	Feature vis_custbe	Module API
Module_Customer_Content	Feature vis_cust	Module API
Module_Customer_Delete	Feature vis_cust	Module API
Module_Customer_Field_Capabilities	Feature fields_cust	Module API
Module_Customer_Field_Name	Feature fields_cust	Module API
Module_Customer_Field_Query	Feature fields_cust	Module API
Module_Customer_Field_Query_OrderBy	Feature fields_cust	Module API
Module_Customer_Field_Query_OrderBy_ LoadIndexRecord	Feature fields_cust	Module API
Module_Customer_Field_Query_Search	Feature fields_cust	Module API
Module_Customer_Field_Query_Value	Feature fields_cust	Module API
Module_Customer_Field_Value	Feature fields_cust	Module API
Module_Customer_Field_Value_Array	Feature fields_cust	Module API
Module_Customer_Fields	Feature fields_cust	Module API
Module_Customer_Fields_Mapped	Feature fields_cust_map	Module API
Module_Customer_Head	Feature vis_cust	Module API
Module_Customer_Insert	Feature vis_cust	Module API
Module_Customer_Runtime_ChangeEmailAddress	Feature custrt	Module API
Module_Customer_Runtime_ChangePassword	Feature custrt	Module API
Module_Customer_Runtime_Insert	Feature custrt	Module API
Module_Customer_Runtime_Update	Feature custrt	Module API
Module_Customer_Runtime_Validate	Feature custrt	Module API
Module_Customer_Set_Field	Feature fields_cust	Module API
Module_Customer_Set_Field_Array	Feature fields_cust	Module API
Module_Customer_Tabs	Feature vis_cust	Module API
Module_Customer_Update	Feature vis_cust	Module API
Module_Customer_Validate	Feature vis_cust	Module API
Module_Description	All Modules	Module API
Module_Domain_Content	Feature vis_domain	Module API
Module_Domain_Head	Feature vis_domain	Module API
Module_Domain_Tabs	Feature vis_domain	Module API
Module_Domain_Update	Feature vis_domain	Module API
Module_Domain_Validate	Feature vis_domain	Module API
Module_External_Requirements_Met	Feature externalreq	Module API
Module_Fulfillment_Content	Feature vis_fulfill	Module API
Module_Fulfillment_Head	Feature vis_fulfill	Module API

Function	File Location/Feature	Module
Module_Fulfillment_Tabs	Feature vis_fulfill	Module API
Module_Fulfillment_Update	Feature vis_fulfill	Module API
Module_Fulfillment_Validate	Feature vis_fulfill	Module API
Module_Install	Feature data_domain	Module API
Module_Install_Store	Feature data_store	Module API
Module_Is_Wizardable	Feature vis_wizard	Module API
Module_JSON	Feature json	Module API
Module_Logging_Content	Feature vis_log	Module API
Module_Logging_Head	Feature vis_log	Module API
Module_Logging_Tabs	Feature vis_log	Module API
Module_Logging_Update	Feature vis_log	Module API
Module_Logging_Validate	Feature vis_log	Module API
Module_Notify_Category_Delete	Feature not_cat	Module API
Module_Notify_Category_Insert	Feature not_cat	Module API
Module_Notify_Category_Update	Feature not_cat	Module API
Module_Notify_Customer_Delete	Feature not_cust	Module API
Module_Notify_Customer_Insert	Feature not_cust	Module API
Module_Notify_Customer_Update	Feature not_cust	Module API
Module_Notify_DigitalDownload_Created	Feature not_digital	Module API
Module_Notify_DigitalDownload_Deleted	Feature not_digital	Module API
Module_Notify_GiftCertificate_Created	Feature not_giftcert	Module API
Module_Notify_GiftCertificate_Deleted	Feature not_giftcert	Module API
Module_Notify_GiftCertificate_Redeemed	Feature not_giftcert	Module API
Module_Notify_GiftCertificate_Updated	Feature not_giftcert	Module API
Module_Notify_Image_Delete	Feature not_image	Module API
Module_Notify_Image_Insert	Feature not_image	Module API
Module_Notify_Order_BatchChange	Feature not_order	Module API
Module_Notify_Order_Delete	Feature not_order	Module API
Module_Notify_Order_Insert	Feature not_order	Module API
Module_Notify_Order_StatusChange	Feature not_order	Module API
Module_Notify_Order_TotalChange	Feature not_order	Module API
Module_Notify_OrderItem_Delete	Feature not_orderitem	Module API
Module_Notify_OrderItem_Insert	Feature not_orderitem	Module API
Module_Notify_OrderItem_StatusChange	Feature not_orderitem	Module API
Module_Notify_OrderItem_Update	Feature not_orderitem	Module API
Module_Notify_OrderReturn_Delete	Feature not_orderreturn	Module API
Module_Notify_OrderReturn_Insert	Feature not_orderreturn	Module API
Module_Notify_OrderReturn_StatusChange	Feature not_orderreturn	Module API

Function	File Location/Feature	Module
Module_Notify_OrderShipment_Delete	Feature not_ordershpmnt	Module API
Module_Notify_OrderShipment_Insert	Feature not_ordershpmnt	Module API
Module_Notify_OrderShipment_StatusChange	Feature not_ordershpmnt	Module API
Module_Notify_Payment_AuthorizationFailure	Feature not_payment	Module API
Module_Notify_Product_Delete	Feature not_prod	Module API
Module_Notify_Product_Insert	Feature not_prod	Module API
Module_Notify_Product_Update	Feature not_prod	Module API
Module_Notify_SEOSettings	Feature not_seo	Module API
Module_Notify_StandardFields	Feature not_fields	Module API
Module_Notify_Subscription_Changed	Feature not_subscript	Module API
Module_Notify_Subscription_Created	Feature not_subscript	Module API
Module_Notify_Subscription_Deleted	Feature not_subscript	Module API
Module_Notify_URI_Delete	Feature not_uri	Module API
Module_Notify_URI_Insert	Feature not_uri	Module API
Module_Notify_URI_Update	Feature not_uri	Module API
Module_Order_Content	Feature vis_order	Module API
Module_Order_Delete	Feature vis_order	Module API
Module_Order_Delete_Order	Feature vis_order	Module API
Module_Order_Field_Capabilities	Feature fields_ordr	Module API
Module_Order_Field_Name	Feature fields_ordr	Module API
Module_Order_Field_Query	Feature fields_ordr	Module API
Module_Order_Field_Query_OrderBy	Feature fields_ordr	Module API
Module_Order_Field_Query_OrderBy_LoadIndexRecord	Feature fields_ordr	Module API
Module_Order_Field_Query_Search	Feature fields_ordr	Module API
Module_Order_Field_Query_Value	Feature fields_ordr	Module API
Module_Order_Field_Value	Feature fields_ordr	Module API
Module_Order_Field_Value_Array	Feature fields_ordr	Module API
Module_Order_Fields	Feature fields_ordr	Module API
Module_Order_Fields_Mapped	Feature fields_ordr_map	Module API
Module_Order_Head	Feature vis_order	Module API
Module_Order_Set_Field	Feature fields_ordr	Module API
Module_Order_Set_Field_Array	Feature fields_ordr	Module API
Module_Order_Tabs	Feature vis_order	Module API
Module_Order_Update	Feature vis_order	Module API
Module_Order_Validate	Feature vis_order	Module API
Module_PackagingRules_Content	Feature vis_pkgrules	Module API
Module_PackagingRules_Head	Feature vis_pkgrules	Module API
Module_PackagingRules_Tabs	Feature vis_pkgrules	Module API

Function	File Location/Feature	Module
Module_PackagingRules_Update	Feature vis_pkgrules	Module API
Module_PackagingRules_Validate	Feature vis_pkgrules	Module API
Module_Payment_Content	Feature vis_payment	Module API
Module_Payment_Head	Feature vis_payment	Module API
Module_Payment_Tabs	Feature vis_payment	Module API
Module_Payment_Update	Feature vis_payment	Module API
Module_Payment_Validate	Feature vis_payment	Module API
Module_Product_BatchEdit_Content	Feature vis_productbe	Module API
Module_Product_BatchEdit_Delete	Feature vis_productbe	Module API
Module_Product_BatchEdit_Head	Feature vis_productbe	Module API
Module_Product_BatchEdit_Tabs	Feature vis_productbe	Module API
Module_Product_BatchEdit_Update	Feature vis_productbe	Module API
Module_Product_BatchEdit_Validate	Feature vis_productbe	Module API
Module_Product_Content	Feature vis_product	Module API
Module_Product_Delete	Feature vis_product	Module API
Module_Product_Facets	Feature prod_facet	Module API
Module_Product_Facet_Query_Search	Feature prod_facet	Module API
Module_Product_Facet_Value_Prompt	Feature prod_facet	Module API
Module_Product_Facet_Value_Prompt_JavaScript	Feature prod_facet	Module API
Module_Product_Facet_Value_Selected	Feature prod_facet	Module API
Module_Product_Facet_Values_Query	Feature prod_facet	Module API
Module_Product_Field_Capabilities	Feature fields_prod	Module API
Module_Product_Field_Name	Feature fields_prod	Module API
Module_Product_Field_Query	Feature fields_prod	Module API
Module_Product_Field_Query_OrderBy	Feature fields_prod	Module API
Module_Product_Field_Query_OrderBy_ LoadIndexRecord	Feature fields_prod	Module API
Module_Product_Field_Query_Search	Feature fields_prod	Module API
Module_Product_Field_Query_Value	Feature fields_prod	Module API
Module_Product_Field_Value	Feature fields_prod	Module API
Module_Product_Field_Value_Array	Feature fields_prod	Module API
Module_Product_Fields	Feature fields_prod	Module API
Module_Product_Fields_Mapped	Feature fields_prod_map	Module API
Module_Product_Head	Feature vis_product	Module API
Module_Product_Insert	Feature vis_product	Module API
Module_Product_Set_Field	Feature fields_prod	Module API
Module_Product_Set_Field_Array	Feature fields_prod	Module API
Module_Product_Tabs	Feature vis_product	Module API

Function	File Location/Feature	Module
Module_Product_Update	Feature vis_product	Module API
Module_Product_Validate	Feature vis_product	Module API
Module_Provision_Store	Feature provision_store	Module API
Module_Shipping_Content	Feature vis_shipping	Module API
Module_Shipping_Head	Feature vis_shipping	Module API
Module_Shipping_Tabs	Feature vis_shipping	Module API
Module_Shipping_Update	Feature vis_shipping	Module API
Module_Shipping_Validate	Feature vis_shipping	Module API
Module_Store_Content	Feature vis_store	Module API
Module_Store_Head	Feature vis_store	Module API
Module_Store_Tabs	Feature vis_store	Module API
Module_Store_Update	Feature vis_store	Module API
Module_Store_Validate	Feature vis_store	Module API
Module_System_Content	Feature vis_system	Module API
Module_System_Head	Feature vis_system	Module API
Module_System_Tabs	Feature vis_system	Module API
Module_System_Update	Feature vis_system	Module API
Module_System_Validate	Feature vis_system	Module API
Module_Uninstall	Feature data_domain	Module API
Module_Uninstall_Store	Feature data_store	Module API
Module_Upgrade	Feature data_domain	Module API
Module_Upgrade_Store	Feature data_store	Module API
Module_Utility_Content	Feature vis_util	Module API
Module_Utility_Head	Feature vis_util	Module API
Module_Utility_Tabs	Feature vis_util	Module API
Module_Utility_Update	Feature vis_util	Module API
Module_Utility_Validate	Feature vis_util	Module API
Module_Wizard_Action	Feature vis_wizard	Module API
Module_Wizard_Content	Feature vis_wizard	Module API
Module_Wizard_Summary_Field	Feature vis_wizard	Module API
Module_Wizard_Summary_Fields	Feature vis_wizard	Module API
Module_Wizard_Summary_Prompt	Feature vis_wizard	Module API
Module_Wizard_Validate	Feature vis_wizard	Module API
Module_Wizard_Validate_Step	Feature vis_wizard	Module API
name_suffix	features/tui/tui_mgr.mv	Template Manager
NextSparseArrayElement	lib/util_public.mv	Helper Functions
OpenBasket	lib/util_public.mv	Helper Functions
OpenBasket_LowLevel	lib/util_public.mv	Helper Functions

Function	File Location/Feature	Module
OpenDataFiles	lib/dbeng/open.mv	Database API
Option_Delete	lib/dbeng/attributes.mv	Database API
Option_Delete_All_Attribute	lib/dbprim/options.mv	Database API
Option_Delete_All_Product	lib/dbprim/options.mv	Database API
Option_Delete_ID	lib/dbprim/options.mv	Database API
Option_Insert	lib/dbprim/options.mv	Database API
Option_Load_Code	lib/dbprim/options.mv	Database API
Option_Load_ID	lib/dbprim/options.mv	Database API
Option_Read	lib/dbprim/options.mv	Database API
Option_Sort_Swap	lib/dbeng/sort.mv	Database API
Option_Update	lib/dbprim/options.mv	Database API
Option_Update_DisplayOrder	lib/dbprim/options.mv	Database API
OptionList_Load_Attribute	lib/dbprim/options.mv	Database API
OptionList_Load_ProductVariant_Possible	lib/dbeng/productvariants.mv	Database API
Order_Change_Shipping	lib/dbeng/orders.mv	Database API
Order_Count_Batch	lib/dbprim/orders.mv	Database API
Order_Create	lib/dbeng/order_compat.mv	Database API
Order_Create_Empty	lib/dbeng/orders.mv	Database API
Order_Create_LowLevel	lib/dbeng/orders.mv	Database API
Order_Delete	lib/dbeng/orders.mv	Database API
Order_Delete_ID	lib/dbprim/orders.mv	Database API
Order_Insert	lib/dbprim/orders.mv	Database API
Order_Load_Customer	lib/dbprim/orders.mv	Database API
Order_Load_First	lib/dbeng/order_compat.mv	Database API
Order_Load_ID	lib/dbeng/order_compat.mv	Database API
Order_Load_Last	lib/dbeng/order_compat.mv	Database API
Order_Load_Next	lib/dbeng/order_compat.mv	Database API
Order_Load_Previous	lib/dbeng/order_compat.mv	Database API
Order_Read	lib/dbeng/order_compat.mv	Database API
Order_Update_Batch	lib/dbprim/orders.mv	Database API
Order_Update_Customer_Information	lib/dbprim/orders.mv	Database API
Order_Update_Payment	lib/dbeng/order_compat.mv	Database API
Order_Update_PaymentTotals	lib/dbeng/orders.mv	Database API
Order_Update_Processed	lib/dbeng/order_compat.mv	Database API
Order_Update_Shipping	lib/dbprim/orders.mv	Database API
Order_Update_Status	lib/dbeng/orders.mv	Database API
Order_Update_Total	lib/dbprim/orders.mv	Database API
Order_Validate	/merchant.mv	Shopping Interface

Function	File Location/Feature	Module
OrderCharge_Copy_Basket	lib/dbeng/orders.mv	Database API
OrderCharge_Delete_All_Order	lib/dbprim/ordercharges.mv	Database API
OrderCharge_Delete_All_Type	lib/dbprim/ordercharges.mv	Database API
OrderCharge_Insert	lib/dbprim/ordercharges.mv	Database API
OrderCharge_Read	lib/dbprim/ordercharges.mv	Database API
OrderCharge_Total	lib/dbeng/orders.mv	Database API
OrderCharge_Total_Type	lib/dbprim/ordercharges.mv	Database API
OrderCharge_Update_Amount	lib/dbprim/ordercharges.mv	Database API
OrderChargeList_Load_Order	lib/dbprim/ordercharges.mv	Database API
OrderChargeList_Load_Type	lib/dbprim/ordercharges.mv	Database API
OrderItem_Copy_Basket	lib/dbeng/orders.mv	Database API
OrderItem_Copy_Basket_WithInventory	lib/dbeng/orders.mv	Database API
OrderItem_Count_Return	lib/dbprim/orderitems.mv	Database API
OrderItem_Count_Shipment	lib/dbprim/orderitems.mv	Database API
OrderItem_Delete	lib/dbprim/orderitems.mv	Database API
OrderItem_Delete_All_Order	lib/dbprim/orderitems.mv	Database API
OrderItem_Insert	lib/dbprim/orderitems.mv	Database API
OrderItem_Insert_LowLevel	lib/dbprim/orderitems.mv	Database API
OrderItem_Load_Line	lib/dbprim/orderitems.mv	Database API
OrderItem_NotifyList_Add	lib/dbeng/orders.mv	Database API
OrderItem_Read	lib/dbprim/orderitems.mv	Database API
OrderItem_Update	lib/dbprim/orderitems.mv	Database API
OrderItem_Update_Status_LowLevel	lib/dbprim/orderitems.mv	Database API
OrderItem_Update_Status_Return	lib/dbeng/orders.mv	Database API
OrderItem_Update_Status_Shipment	lib/dbeng/orders.mv	Database API
OrderItemList_BackOrder	lib/dbeng/orders.mv	Database API
OrderItemList_BackOrder_Dates	lib/dbeng/orders.mv	Database API
OrderItemList_Cancel	lib/dbeng/orders.mv	Database API
OrderItemList_CreateReturn	lib/dbeng/orders.mv	Database API
OrderItemList_CreateShipment	lib/dbeng/orders.mv	Database API
OrderItemList_CreateShipment_LowLevel	lib/dbeng/orders.mv	Database API
OrderItemList_Delete	lib/dbeng/orders.mv	Database API
OrderItemList_Load_Order	lib/dbprim/orderitems.mv	Database API
OrderItemList_Load_ProductID	lib/dbprim/orderitems.mv	Database API
OrderItemList_Load_Return	lib/dbprim/orderitems.mv	Database API
OrderItemList_Load_Shipment	lib/dbprim/orderitems.mv	Database API
OrderItemList_Load_Status	lib/dbprim/orderitems.mv	Database API
OrderItemList_Received	lib/dbeng/orders.mv	Database API

Function	File Location/Feature	Module
OrderItemList_RemoveFromReturns	lib/dbeng/orders.mv	Database API
OrderItemList_RemoveFromShipments	lib/dbeng/orders.mv	Database API
OrderList_Load_Batch	lib/dbeng/order_compat.mv	Database API
OrderList_Load_Batch_Unprocessed	lib/dbeng/order_compat.mv	Database API
OrderList_Load_Customer	lib/dbprim/orders.mv	Database API
OrderList_Load_Customer_Offset	lib/dbeng/orders.mv	Database API
OrderList_Load_Offset	lib/dbeng/order_compat.mv	Database API
OrderList_Load_Offset_Uncategorized	lib/dbeng/order_compat.mv	Database API
OrderMinimumsMet	lib/util_public.mv	Helper Functions
OrderOption_Copy_Basket	lib/dbeng/orders.mv	Database API
OrderOption_Delete_All_Line	lib/dbprim/orderoptions.mv	Database API
OrderOption_Delete_All_Order	lib/dbprim/orderoptions.mv	Database API
OrderOption_Insert	lib/dbprim/orderoptions.mv	Database API
OrderOption_Read	lib/dbprim/orderoptions.mv	Database API
OrderOption_Total_Line	lib/dbeng/orders.mv	Database API
OrderOptionList_Load_Line	lib/dbprim/orderoptions.mv	Database API
OrderPayment_Create	lib/dbeng/orders.mv	Database API
OrderPayment_Delete	lib/dbeng/orders.mv	Database API
OrderPayment_Delete_All_Order	lib/dbeng/orders.mv	Database API
OrderPayment_Insert	lib/dbprim/orderpayments.mv	Database API
OrderPayment_Load_ID	lib/dbprim/orderpayments.mv	Database API
OrderPayment_Read	lib/dbprim/orderpayments.mv	Database API
OrderPayment_Update	lib/dbeng/orders.mv	Database API
OrderPayment_Update_Amounts	lib/dbprim/orderpayments.mv	Database API
OrderPayment_Update_LowLevel	lib/dbprim/orderpayments.mv	Database API
OrderPaymentList_Load_Order	lib/dbprim/orderpayments.mv	Database API
OrderPaymentList_Load_Order_Authorization	lib/dbprim/orderpayments.mv	Database API
OrderPaymentList_Load_Order_Capture	lib/dbprim/orderpayments.mv	Database API
OrderPaymentList_Load_Order_Module_Authorization	lib/dbprim/orderpayments.mv	Database API
OrderReturn_Delete	lib/dbprim/orderreturns.mv	Database API
OrderReturn_Delete_All_Order	lib/dbprim/orderreturns.mv	Database API
OrderReturn_Insert	lib/dbprim/orderreturns.mv	Database API
OrderReturn_Insert_LowLevel	lib/dbprim/orderreturns.mv	Database API
OrderReturn_Load_ID	lib/dbprim/orderreturns.mv	Database API
OrderReturn_NotifyList_Add	lib/dbeng/orders.mv	Database API
OrderReturn_Read	lib/dbprim/orderreturns.mv	Database API
OrderReturn_Update_Status	lib/dbeng/orders.mv	Database API
OrderReturn_Update_Status_LowLevel	lib/dbprim/orderreturns.mv	Database API

Function	File Location/Feature	Module
OrderReturnList_Update_Status	lib/dbeng/orders.mv	Database API
OrderShipment_Delete	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Delete_All_Order	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Insert	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Insert_LowLevel	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Load_Code	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Load_ID	lib/dbprim/ordershipments.mv	Database API
OrderShipment_NotifyList_Add	lib/dbeng/orders.mv	Database API
OrderShipment_Read	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Update_Label	lib/dbprim/ordershipments.mv	Database API
OrderShipment_Update_Status	lib/dbeng/orders.mv	Database API
OrderShipment_Update_Status_LowLevel	lib/dbprim/ordershipments.mv	Database API
OrderShipmentList_Load_Order	lib/dbprim/ordershipments.mv	Database API
OrderShipmentList_Update_Status	lib/dbeng/orders.mv	Database API
OutputCookies	lib/util_public.mv	Helper Functions
Page_Delete_ID	features/tui/tui_db.mv	Template Database
Page_Insert	features/tui/tui_db.mv	Template Database
Page_Load_Code	features/tui/tui_db.mv	Template Database
Page_Load_First	features/tui/tui_db.mv	Template Database
Page_Load_Last	features/tui/tui_db.mv	Template Database
Page_Load_Next	features/tui/tui_db.mv	Template Database
Page_Load_Previous	features/tui/tui_db.mv	Template Database
Page_Read	features/tui/tui_db.mv	Template Database
Page_Update	features/tui/tui_db.mv	Template Database
PageList_Load_All	features/tui/tui_db.mv	Template Database
PageList_Load_Item	features/tui/tui_db.mv	Template Database
PageList_Load_Offset	features/tui/tui_db.mv	Template Database
PageList_Load_Offset_Item_All	features/tui/tui_db.mv	Template Database
PageList_Load_Offset_Item_Assigned	features/tui/tui_db.mv	Template Database
PageList_Load_Offset_Item_Unassigned	features/tui/tui_db.mv	Template Database
PageList_Load_UI	features/tui/tui_db.mv	Template Database
PageXItem_Delete_All_Item_LowLevel	features/tui/tui_db.mv	Template Database
PageXItem_Delete_All_Page_LowLevel	features/tui/tui_db.mv	Template Database
PageXItem_Delete_LowLevel	features/tui/tui_db.mv	Template Database
PageXItem_Insert_LowLevel	features/tui/tui_db.mv	Template Database
PageXItem_Load	features/tui/tui_db.mv	Template Database
PageXItem_Read	features/tui/tui_db.mv	Template Database
ParseCookies	lib/util_public.mv	Helper Functions

Function	File Location/Feature	Module
PaymentModule_Authorize	Feature payment	Module API
PaymentModule_Balance	Feature payment	Module API
PaymentModule_Capabilities	Feature payment	Module API
PaymentModule_Enabled_Methods	Feature payment	Module API
PaymentModule_LeftNavigation	Feature payment	Module API
PaymentModule_Manipulate_Shipping	Feature payment	Module API
PaymentModule_Method_Capabilities	Feature payment	Module API
PaymentModule_Order_Authorize	Feature payment	Module API
PaymentModule_Order_Authorize_Field	Feature payment	Module API
PaymentModule_Order_Authorize_Fields	Feature payment	Module API
PaymentModule_Order_Authorize_Hide_Additional_Fields	Feature payment	Module API
PaymentModule_Order_Authorize_Invalid	Feature payment	Module API
PaymentModule_Order_Authorize_Methods	Feature payment	Module API
PaymentModule_Order_Authorize_PaymentCard	Feature payment	Module API
PaymentModule_Order_Authorize_Prompt	Feature payment	Module API
PaymentModule_Order_Authorize_Validate	Feature payment	Module API
PaymentModule_Order_Content	Feature payment	Module API
PaymentModule_Order_Delete	Feature payment	Module API
PaymentModule_Order_Head	Feature payment	Module API
PaymentModule_OrderPayment_Capture	Feature payment	Module API
PaymentModule_OrderPayment_Refund	Feature payment	Module API
PaymentModule_OrderPayment_VOID	Feature payment	Module API
PaymentModule_Order_Tabs	Feature payment	Module API
PaymentModule_Order_Update	Feature payment	Module API
PaymentModule_Order_Validate	Feature payment	Module API
PaymentModule_Payment_Description	Feature payment	Module API
PaymentModule_Payment_Field	Feature payment	Module API
PaymentModule_Payment_Fields	Feature payment	Module API
PaymentModule_Payment_Hide_Additional_Fields	Feature payment	Module API
PaymentModule_Payment_Invalid	Feature payment	Module API
PaymentModule_Payment_Message	Feature payment	Module API
PaymentModule_Payment_Methods	Feature payment	Module API
PaymentModule_Payment_Prompt	Feature payment	Module API
PaymentModule_Payment_URL	Feature payment	Module API
PaymentModule_Payment_Validate	Feature payment	Module API
PaymentModule_Process	Feature payment	Module API
PaymentModule_Report_Description	Feature payment	Module API

Function	File Location/Feature	Module
PaymentModule_Report_Fields	Feature payment	Module API
PaymentModule_Report_Label	Feature payment	Module API
PaymentModule_Report_Value	Feature payment	Module API
PaymentModule_Runtime_Authorize	Feature payment	Module API
PaymentModule_Runtime_Authorize_PaymentCard	Feature payment	Module API
PaymentModule_Runtime_SplitPayment_Authorize	Feature payment	Module API
PaymentModule_Runtime_SplitPayment_Prepare	Feature payment	Module API
PaymentModule_Runtime_SplitPayment_Rollback	Feature payment	Module API
PGR_Create_Data_Files	features/pgr/pgr_db.mv	Price Group
PGR_Store_Create	features/pgr/pgr_db.mv	Price Group
PGR_Store_Delete	features/pgr/pgr_db.mv	Price Group
Phone_Validate	lib/util_public.mv	Helper Functions
PreviousSparseArrayElement	lib/util_public.mv	Helper Functions
PriceGroup_Delete	features/pgr/pgr_db.mv	Price Group
PriceGroup_Delete_ID	features/pgr/pgr_db.mv	Price Group
PriceGroup_Insert	features/pgr/pgr_db.mv	Price Group
PriceGroup_Load_ID	features/pgr/pgr_db.mv	Price Group
PriceGroup_Load_Name	features/pgr/pgr_db.mv	Price Group
PriceGroup_Read	features/pgr/pgr_db.mv	Price Group
PriceGroup_Update	features/pgr/pgr_db.mv	Price Group
PriceGroupList_Load_All	features/pgr/pgr_db.mv	Price Group
PriceGroupList_Load_Offset	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Delete	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Delete_All_Customer	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Delete_All_PriceGroup	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Delete_LowLevel	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Insert	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Insert_LowLevel	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Load	features/pgr/pgr_db.mv	Price Group
PriceGroupXCustomer_Read	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Delete	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Delete_All_PriceGroup	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Delete_All_Product	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Delete_LowLevel	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Insert	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Insert_LowLevel	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Load	features/pgr/pgr_db.mv	Price Group
PriceGroupXProduct_Read	features/pgr/pgr_db.mv	Price Group

Function	File Location/Feature	Module
PrivateKey_Delete_ID	lib/dbprim/encryption.mv	Database API
PrivateKey_Insert	lib/dbprim/encryption.mv	Database API
PrivateKey_Load_ID	lib/dbprim/encryption.mv	Database API
PrivateKey_Read	lib/dbprim/encryption.mv	Database API
ProdStat_Count	features/sta/sta_db.mv	Statistics
ProdStats_Add_Counts	features/sta/sta_db.mv	Statistics
ProdStats_Insert	features/sta/sta_db.mv	Statistics
ProdStats_Load_Product	features/sta/sta_db.mv	Statistics
ProdStats_Read	features/sta/sta_db.mv	Statistics
ProdStats_Update	features/sta/sta_db.mv	Statistics
ProdStatsList_Load_Max	features/sta/sta_db.mv	Statistics
Product_Count	lib/dbprim/products.mv	Database API
Product_Decrement_AvailabilityGroupCount	lib/dbprim/products.mv	Database API
Product_Decrement_AvailabilityGroupCount_All	lib/dbeng/products.mv	Database API
Product_Decrement_CategoryCount	lib/dbprim/products.mv	Database API
Product_Decrement_PriceGroupCount	lib/dbprim/products.mv	Database API
Product_Decrement_PriceGroupCount_All	lib/dbeng/products.mv	Database API
Product_Delete	lib/dbeng/products.mv	Database API
Product_Delete_ID	lib/dbprim/products.mv	Database API
Product_Increment_AvailabilityGroupCount	lib/dbprim/products.mv	Database API
Product_Increment_CategoryCount	lib/dbprim/products.mv	Database API
Product_Increment_PriceGroupCount	lib/dbprim/products.mv	Database API
Product_Insert	lib/dbprim/products.mv	Database API
Product_Load_Code	lib/dbprim/products.mv	Database API
Product_Load_Code_WithRuntimeInventory	lib/dbeng/products.mv	Database API
Product_Load_DisplayOrder	lib/dbprim/products.mv	Database API
Product_Load_First	lib/dbeng/products.mv	Database API
Product_Load_ID	lib/dbprim/products.mv	Database API
Product_Load_Last	lib/dbeng/products.mv	Database API
Product_Load_Next	lib/dbeng/products.mv	Database API
Product_Load_Previous	lib/dbeng/products.mv	Database API
Product_Read	lib/dbprim/products.mv	Database API
Product_Sort_All	lib/dbeng/sort.mv	Database API
Product_Sort_Swap	lib/dbeng/sort.mv	Database API
Product_Update	lib/dbprim/products.mv	Database API
Product_Update_DisplayOrder	lib/dbprim/products.mv	Database API
ProductAttributeList_Update_Offsets	lib/dbeng/attributes.mv	Database API
ProductAttributeList_Update_Offsets_PastEnd	lib/dbeng/attributes.mv	Database API

Function	File Location/Feature	Module
ProductAttributeOptionList_Update_Offsets	lib/dbeng/attributes.mv	Database API
ProductAttributeOptionList_Update_Offsets_PastEnd	lib/dbeng/attributes.mv	Database API
ProductKit_Delete_All_AttributeTemplateAttribute	lib/dbprim/productkits.mv	Database API
ProductKit_Delete_All_AttributeTemplateOption	lib/dbprim/productkits.mv	Database API
ProductKit_Delete_All_Part	lib/dbprim/productkits.mv	Database API
ProductKit_Delete_All_Product	lib/dbprim/productkits.mv	Database API
ProductKit_Delete_All_Product_Attribute	lib/dbprim/productkits.mv	Database API
ProductKit_Delete_All_Product_Option	lib/dbprim/productkits.mv	Database API
ProductKit_Delete_Option	lib/dbprim/productkits.mv	Database API
ProductKit_Generate_Variants	lib/dbeng/productkits.mv	Database API
ProductKit_Generate_Variants_Recursive	lib/dbeng/productkits.mv	Database API
ProductKit_Insert	lib/dbprim/productkits.mv	Database API
ProductList_Load_All	lib/dbprim/products.mv	Database API
ProductList_Load_Offset	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Active	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_AvailabilityGroup_All	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_AvailabilityGroup_Assigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_AvailabilityGroup_Unassigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Category_All	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Category_Assigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Category_Unassigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_PriceGroup_All	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_PriceGroup_Assigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_PriceGroup_Unassigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_RelatedProducts_All	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Uncategorized	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Upsell_All	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Upsell_Assigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_Upsell_Unassigned	lib/dbeng/products.mv	Database API
ProductList_Load_Offset_UpsellXProduct_Required	lib/dbeng/products.mv	Database API
ProductList_Load_RelatedProducts_Offset_Assigned	lib/dbeng/products.mv	Database API
ProductList_Load_RelatedProducts_Offset_Unassigned	lib/dbeng/products.mv	Database API
ProductList_Load_Variant	lib/dbeng/productvariants.mv	Database API
ProductList_Update_Offsets	lib/dbeng/products.mv	Database API
Products_Update_Offsets_PastEnd	lib/dbeng/products.mv	Database API
ProductVariant_Decrement_Part_Count	lib/dbprim/productvariants.mv	Database API
ProductVariant_Delete	lib/dbeng/productvariants.mv	Database API
ProductVariant_Delete_All_AttributeTemplateAttribute	lib/dbeng/productvariants.mv	Database API

Function	File Location/Feature	Module
ProductVariant_Delete_All_AttributeTemplateOption	lib/dbeng/productvariants.mv	Database API
ProductVariant_Delete_All_Product	lib/dbeng/productvariants.mv	Database API
ProductVariant_Delete_All_Product_Attribute	lib/dbeng/productvariants.mv	Database API
ProductVariant_Delete_All_Product_LowLevel	lib/dbprim/productvariants.mv	Database API
ProductVariant_Delete_All_Product_NotDefault	lib/dbeng/productvariants.mv	Database API
ProductVariant_Delete_All_Product_NotDefault _LowLevel	lib/dbprim/productvariants.mv	Database API
ProductVariant_Delete_All_Product_Option	lib/dbeng/productvariants.mv	Database API
ProductVariant_Delete_LowLevel	lib/dbprim/productvariants.mv	Database API
ProductVariant_Generate	lib/dbeng/productvariants.mv	Database API
ProductVariant_Generate_Part	lib/dbeng/productvariants.mv	Database API
ProductVariant_Generate_Recursive	lib/dbeng/productvariants.mv	Database API
ProductVariant_Insert_Attributes	lib/dbeng/productvariants.mv	Database API
ProductVariant_Insert_LowLevel	lib/dbprim/productvariants.mv	Database API
ProductVariant_Is_Possible	lib/dbeng/productvariants.mv	Database API
ProductVariant_Load_Attributes	lib/dbeng/productvariants.mv	Database API
ProductVariant_Load_Attributes_WithInventory	lib/dbeng/productvariants.mv	Database API
ProductVariant_Read	lib/dbprim/productvariants.mv	Database API
ProductVariant_Sum_Pricing	lib/dbeng/productvariants.mv	Database API
ProductVariantList_Load_AttributeTemplateAttribute	lib/dbprim/productvariants.mv	Database API
ProductVariantList_Load_AttributeTemplateOption	lib/dbprim/productvariants.mv	Database API
ProductVariantList_Load_Part	lib/dbprim/productvariants.mv	Database API
ProductVariantList_Load_Product_Attribute	lib/dbprim/productvariants.mv	Database API
ProductVariantList_Load_Product_Option	lib/dbprim/productvariants.mv	Database API
ProductVariantPart_Delete_All_Part	lib/dbeng/productvariants.mv	Database API
ProductVariantPart_Delete_All_Part_LowLevel	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPart_Delete_All_Product	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPart_Delete_All_Product_NotDefault	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPart_Delete_All_Variant	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPart_Insert	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPart_Read	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPartList_Load_Part	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPartList_Load_Variant	lib/dbprim/productvariantparts.mv	Database API
ProductVariantPricing_Delete_All_Product	lib/dbprim/productvariantpricing.mv	Database API
ProductVariantPricing_Delete_All_Product_NotDefault	lib/dbprim/productvariantpricing.mv	Database API
ProductVariantPricing_Delete_Variant	lib/dbprim/productvariantpricing.mv	Database API
ProductVariantPricing_Insert	lib/dbprim/productvariantpricing.mv	Database API
ProductVariantPricing_Load_Variant	lib/dbprim/productvariantpricing.mv	Database API

Function	File Location/Feature	Module
ProductVariantPricing_Read	lib/dbprim/productvariantpricing.mv	Database API
PRV_Action_Provision_Domain	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_AdminDisablingUpdate	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_Country_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_Country_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_Country_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_SEOSettings_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_Settings	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_StoreCreate	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_StoreDelete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_StoreUpdate	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_TrackingLink_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_TrackingLink_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_TrackingLink_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_UserAdd	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_UserDelete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Domain_UserUpdate	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Fragment	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Affiliate_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Affiliate_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Affiliate_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AffiliateEmailNotification_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AffiliateLostPassword Email_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AffiliateOptions_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplate_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplate_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplate_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplateAttribute_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplateAttribute_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplateAttribute_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplateAttribute Option_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AttributeTemplateAttribute Option_Delete	features/prv/prv_ad.mv	Provisioning System

Function	File Location/Feature	Module
PRV_Action_Provision_Store_AttributeTemplateAttribute Option_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroup_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroup_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroup_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroupCatego-ry_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroupCatego-ry_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroup Customer_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroup Customer_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroupProduct _Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_AvailabilityGroupProduct _Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Category_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Category_CustomField	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Category_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Category_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_CategoryProduct_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_CategoryProduct _Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Countries_Replace	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Country	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Country_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Country_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Customer_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Customer_CustomField	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Customer_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Customer_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_CustomerFields_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_CustomerLostPassword Email_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Encryption_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Encryption_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Encryption_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Group_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Group_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Group_Update	features/prv/prv_ad.mv	Provisioning System

Function	File Location/Feature	Module
PRV_Action_Provision_Store_GroupUser_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_GroupUser_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_InventoryEmail Notification_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_InventoryProductSet-tings_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_InventorySettings_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Item_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Item_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Item_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ItemExtension_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ItemExtension_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_MivaSubmitSettings _Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Module	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Order_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Order_Backorder_Items	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_OrderShipment_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_OrderShipment_SetStatus	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Page_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Page_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Page_Item	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Page_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PageItem_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PageItem_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroup_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroup_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroup_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroupCustomer _Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroupCustomer _Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroupProduct _Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_PriceGroupProduct _Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Product_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Product_CustomField	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Product_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Product_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttribute_Add	features/prv/prv_ad.mv	Provisioning System

Function	File Location/Feature	Module
PRV_Action_Provision_Store_ProductAttribute_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttribute_Delete_All	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttribute_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttributeOption_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttributeOption_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttributeOption_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttributeTemplate_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductAttributeTemplate_Copy	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductKit_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductKit_Delete_All	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductKit_Generate_Variants	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductKit_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductRelatedProduct_Assign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductRelatedProduct_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Delete_All	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Delete_Default	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Generate	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_ProductVariant_Update_Default	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_Skin_Select	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_State_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_State_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_State_DeleteAll	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_State_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_UpsellSettings_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_UpsoldProduct_Add	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_UpsoldProduct_Delete	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_UpsoldProduct_Update	features/prv/prv_ad.mv	Provisioning System
PRV_Action_Provision_Store_UpsoldProductRequiredProduct_Assign	features/prv/prv_ad.mv	Provisioning System

Function	File Location/Feature	Module
PRV_Action_Provision_Store_UpsoldProductRequired Product_Unassign	features/prv/prv_ad.mv	Provisioning System
PRV_Action_ProvisionEntireFile	features/prv/prv_ad.mv	Provisioning System
PRV_Action_ProvisionStringData	features/prv/prv_ad.mv	Provisioning System
PRV_Attribute_Boolean	features/prv/prv_ad.mv	Provisioning System
PRV_Attribute_List	features/prv/prv_ad.mv	Provisioning System
PRV_Attribute_Number	features/prv/prv_ad.mv	Provisioning System
PRV_Attribute_Text	features/prv/prv_ad.mv	Provisioning System
PRV_Backorder_ProductList	features/prv/prv_ad.mv	Provisioning System
PRV_ErroredStamp	features/prv/prv_ad.mv	Provisioning System
PRV_ExecutedStamp	features/prv/prv_ad.mv	Provisioning System
PRV_Frameset	features/prv/prv_ad.mv	Provisioning System
PRV_Inventory_PartList	features/prv/prv_ad.mv	Provisioning System
PRV_Inventory_ProductAttributeAndOptionList	features/prv/prv_ad.mv	Provisioning System
PRV_LogError	features/prv/prv_ad.mv	Provisioning System
PRV_LogMessage	features/prv/prv_ad.mv	Provisioning System
PRV_LogParseError	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Calculate_Charges	features/prv/prv_ad.mv	Provisioning System
PRV_Order_ChargeList	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Cleanup_Dummy_Basket	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Create_Dummy_Basket	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Item	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Item_OptionList	features/prv/prv_ad.mv	Provisioning System
PRV_Order_ItemList	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Product	features/prv/prv_ad.mv	Provisioning System
PRV_Order_Product_OptionList	features/prv/prv_ad.mv	Provisioning System
PRV_Order_TriggerFulfillmentModules	features/prv/prv_ad.mv	Provisioning System
PRV_OrderItem_InsertAttributes	features/prv/prv_ad.mv	Provisioning System
PRV_OrderItem_Split	features/prv/prv_ad.mv	Provisioning System
PRV_OrderShipment_ProductList	features/prv/prv_ad.mv	Provisioning System
PRV_ProvisionAsNeeded	features/prv/prv_ad.mv	Provisioning System
PRV_Screen_Info	features/prv/prv_ad.mv	Provisioning System
PRV_Screen_Work	features/prv/prv_ad.mv	Provisioning System
PRV_Status	features/prv/prv_ad.mv	Provisioning System
PRV_Tag_Boolean	features/prv/prv_ad.mv	Provisioning System
PRV_Tag_Code	features/prv/prv_ad.mv	Provisioning System
PRV_Tag_Date	features/prv/prv_ad.mv	Provisioning System
PRV_Tag_Exists	features/prv/prv_ad.mv	Provisioning System

Function	File Location/Feature	Module
PRV_Tag_List	features/prv/prv_ad.mv	Provisioning System
PRV_Tag_Number	features/prv/prv_ad.mv	Provisioning System
PRV_Tag_Text	features/prv/prv_ad.mv	Provisioning System
QuickSortArray	lib/util_public.mv	Helper Functions
QuickSortArray_Compare	lib/util_public.mv	Helper Functions
QuickSortArray_Partition	lib/util_public.mv	Helper Functions
QuickSortArray_Sort	lib/util_public.mv	Helper Functions
QuickSortArray_Swap	lib/util_public.mv	Helper Functions
RelatedProduct_Delete	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Delete_Product	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Insert	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Load_Product	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Read	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Update_Offsets	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Update_Offsets_PastEnd	features/rpd/rpd_db.mv	Related Products
RelatedProduct_Update_Product	features/rpd/rpd_db.mv	Related Products
RelatedProductList_Customer_Load_Product	features/rpd/rpd_db.mv	Related Products
ReportModule_Calculate_DateRange_All	Feature report	Module API
ReportModule_Capabilities	Feature report	Module API
ReportModule_Chart_Type	Feature report	Module API
ReportModule_Display	Feature report	Module API
ReportModule_Export	Feature report	Module API
ReportModule_Field	Feature report	Module API
ReportModule_Fields	Feature report	Module API
ReportModule_Format_Vertical_Label	Feature report	Module API
ReportModule_HTML_Chart	Feature report	Module API
ReportModule_Invalid	Feature report	Module API
ReportModule_Prompt	Feature report	Module API
ReportModule_Provision_Settings	Feature report	Module API
ReportModule_Run	Feature report	Module API
ReportModule_Run_DateRange	Feature report	Module API
ReportModule_Run_Intervals	Feature report	Module API
ReportModule_SVG_Line_Chart_Definition	Feature report	Module API
ReportModule_Tabular_Definition	Feature report	Module API
ReportModule_Update	Feature report	Module API
ReportModule_Validate	Feature report	Module API
RPD_Store_Create	features/rpd/rpd_db.mv	Related Products
RPD_Store_Delete	features/rpd/rpd_db.mv	Related Products

Function	File Location/Feature	Module
Runtime_Basket_Empty	lib/dbeng/runtime.mv	Database API
Runtime_BasketItem_Delete	lib/dbeng/runtime.mv	Database API
Runtime_BasketItem_Delete_Transaction	lib/dbeng/runtime.mv	Database API
Runtime_BasketItem_Delete_UpsoId	lib/dbeng/runtime.mv	Database API
Runtime_BasketItem_Increment_Quantity	lib/dbeng/runtime.mv	Database API
Runtime_BasketItem_Insert	lib/dbeng/runtime.mv	Database API
Runtime_Category_Load_Code	lib/dbeng/runtime.mv	Database API
Runtime_Category_Load_ID	lib/dbeng/runtime.mv	Database API
Runtime_CategoryList_Load_All	lib/dbeng/runtime.mv	Database API
Runtime_CategoryList_Load_Parent	lib/dbeng/runtime.mv	Database API
Runtime_PriceGroupList_Load_Product	lib/dbeng/runtime.mv	Database API
Runtime_Product_InventoryFields_Read	features/inv/inv_rt.mv	Inventory Runtime
Runtime_Product_Load_BasketItem	lib/dbeng/runtime.mv	Database API
Runtime_Product_Load_Code	lib/dbeng/runtime.mv	Database API
Runtime_ProductList_Load_All	lib/dbeng/runtime.mv	Database API
Runtime_ProductList_Load_Basket	lib/dbeng/runtime.mv	Database API
Runtime_ProductList_Load_Offset_All	lib/dbeng/runtime.mv	Database API
Runtime_ProductList_Load_Offset_Category	lib/dbeng/runtime.mv	Database API
Runtime_ProductList_Load_Offset_RelatedProducts	lib/dbeng/runtime.mv	Database API
Runtime_ProductList_Load_Offset_Search	lib/dbeng/runtime.mv	Database API
Runtime_UpsoldProductList_Load	features/usl/usl_db.mv	Upsale Database
SBM_CreateDataFiles	features/sbm/sbm_db.mv	Miva Submit
SBM_DeleteDataFiles	features/sbm/sbm_db.mv	Miva Submit
SBM_Domain_Pack	features/sbm/sbm_db.mv	Miva Submit
SBM_Store_Create	features/sbm/sbm_db.mv	Miva Submit
SBM_Store_Delete	features/sbm/sbm_db.mv	Miva Submit
ScheduledTaskModule_Capabilities	Feature scheduledtask	Module API
ScheduledTaskModule_Delete	Feature scheduledtask	Module API
ScheduledTaskModule_Execute	Feature scheduledtask	Module API
ScheduledTaskModule_Field	Feature scheduledtask	Module API
ScheduledTaskModule_Fields	Feature scheduledtask	Module API
ScheduledTaskModule_Invalid	Feature scheduledtask	Module API
ScheduledTaskModule_Operations	Feature scheduledtask	Module API
ScheduledTaskModule_Prompt	Feature scheduledtask	Module API
ScheduledTaskModule_Provision_Settings	Feature scheduledtask	Module API
ScheduledTaskModule_Update	Feature scheduledtask	Module API
ScheduledTaskModule_Validate	Feature scheduledtask	Module API
Screen_CheckAll	admin/ui.mv	Administrative UI

Function	File Location/Feature	Module
Screen_CheckAllModified	admin/ui.mv	Administrative UI
Send_Email_Attachment	lib/util_public.mv	Helper Functions
SendEmail	lib/util_public.mv	Helper Functions
SendMimeEmail	lib/util_public.mv	Helper Functions
SEOSettings_Create	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Default_Values	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Insert	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Load	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Normalize	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Read	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Sample_Product	lib/dbprim/seo_settings.mv	Database API
SEOSettings_Update	lib/dbprim/seo_settings.mv	Database API
SetCookie	lib/util_public.mv	Helper Functions
SetRuntimeCookies	lib/util_public.mv	Helper Functions
SetRuntimePaths	lib/util_public.mv	Helper Functions
ShippingModule_Basket_Methods	Feature shipping	Module API
ShippingModule_Calculate_Basket	Feature shipping	Module API
ShippingModule_Description	Feature shipping	Module API
ShippingModule_Enabled_Methods	Feature shipping	Module API
ShippingModule_Label_Boxes	Feature shipping_label	Module API
ShippingModule_Label_Field	Feature shipping_label	Module API
ShippingModule_Label_Fields	Feature shipping_label	Module API
ShippingModule_Label_Generate	Feature shipping_label	Module API
ShippingModule_Label_Invalid	Feature shipping_label	Module API
ShippingModule_Label_Methods	Feature shipping_label	Module API
ShippingModule_Label_Package_Field	Feature shipping_label	Module API
ShippingModule_Label_Package_Fields	Feature shipping_label	Module API
ShippingModule_Label_Package_Invalid	Feature shipping_label	Module API
ShippingModule_Label_Package_Prompt	Feature shipping_label	Module API
ShippingModule_Label_Package_Validate	Feature shipping_label	Module API
ShippingModule_Label_Package_Validate_Package	Feature shipping_label	Module API
ShippingModule_Label_Prompt	Feature shipping_label	Module API
ShippingModule_Label_Render	Feature shipping_label	Module API
ShippingModule_Label_Shipment_Field	Feature shipping_label	Module API
ShippingModule_Label_Shipment_Fields	Feature shipping_label	Module API
ShippingModule_Label_Shipment_Invalid	Feature shipping_label	Module API
ShippingModule_Label_Shipment_Prompt	Feature shipping_label	Module API
ShippingModule_Label_Shipment_Validate	Feature shipping_label	Module API

Function	File Location/Feature	Module
ShippingModule_Label_Validate	Feature shipping_label	Module API
ShippingModule_Label_Void_Shipment	Feature shipping_label	Module API
ShippingModule_Labels_Generate	Feature shipping_label	Module API
ShippingModule_Order_Content	Feature shipping	Module API
ShippingModule_Order_Delete	Feature shipping	Module API
ShippingModule_Order_Head	Feature shipping	Module API
ShippingModule_Order_Tabs	Feature shipping	Module API
ShippingModule_Order_Update	Feature shipping	Module API
ShippingModule_Order_Validate	Feature shipping	Module API
ShippingModule_Report_Fields	Feature shipping	Module API
ShippingModule_Report_Label	Feature shipping	Module API
ShippingModule_Report_Value	Feature shipping	Module API
ShippingModule_Shipping_Methods	Feature shipping	Module API
Show_BoldNA	admin/ui.mv	Administrative UI
Show_NA	admin/ui.mv	Administrative UI
_SKIN_Export	features/tui/tui_ut.mv	Template Utility
_SKIN_Install	features/tui/tui_ut.mv	Template Utility
SKIN_Cleanup	features/tui/tui_ut.mv	Template Utility
SKIN_Delete_ID	features/tui/tui_db.mv	Template Database
SKIN_Directory_Delete	features/tui/tui_ut.mv	Template Utility
SKIN_Directory_Verify	features/tui/tui_ut.mv	Template Utility
SKIN_Enabled	features/tui/tui_db.mv	Template Database
SKIN_Export	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Components	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Config	features/tui/tui_ut.mv	Template Utility
SKIN_Export_CSS	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Images	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Page_Items	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Page_Settings	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Pages	features/tui/tui_ut.mv	Template Utility
SKIN_Export_Remove_Assignment	features/tui/tui_ut.mv	Template Utility
SKIN_Export_RequiredItems	features/tui/tui_ut.mv	Template Utility
SKIN_Fix_Page_Settings	features/tui/tui_ut.mv	Template Utility
SKIN_Image_List	features/tui/tui_ut.mv	Template Utility
SKIN_Insert	features/tui/tui_db.mv	Template Database
SKIN_Install	features/tui/tui_ut.mv	Template Utility
SKIN_Install_Components	features/tui/tui_ut.mv	Template Utility
SKIN_Install_CSS_Files	features/tui/tui_ut.mv	Template Utility

Function	File Location/Feature	Module
SKIN_Install_Images	features/tui/tui_ut.mv	Template Utility
SKIN_Install_One_Component	features/tui/tui_ut.mv	Template Utility
SKIN_Install_One_Page	features/tui/tui_ut.mv	Template Utility
SKIN_Install_Pages	features/tui/tui_ut.mv	Template Utility
SKIN_Load_Code	features/tui/tui_db.mv	Template Database
SKIN_Lookup_TemplateID	features/tui/tui_ut.mv	Template Utility
SKIN_Preview	features/tui/tui_ut.mv	Template Utility
SKIN_Read	features/tui/tui_db.mv	Template Database
SKIN_Register	features/tui/tui_ut.mv	Template Utility
SKIN_Register_Unpacked	features/tui/tui_ut.mv	Template Utility
SKIN_Sample_Retrieve	features/tui/tui_ut.mv	Template Utility
SKIN_Store_Create	features/tui/tui_db.mv	Template Database
SKIN_Unpack	features/tui/tui_ut.mv	Template Utility
SKIN_Update	features/tui/tui_db.mv	Template Database
SKIN_Validate	features/tui/tui_ut.mv	Template Utility
SKINList_Load_All	features/tui/tui_db.mv	Template Database
SKINList_Load_All_Order	features/tui/tui_db.mv	Template Database
SKINList_Load_Offset	features/tui/tui_db.mv	Template Database
SkinsComponentModule_Description	Feature skins	Module API
SkinsComponentModule_Export	Feature skins	Module API
SkinsComponentModule_Export_Item	Feature skins	Module API
SortOffsetArray	lib/util_public.mv	Helper Functions
SortOffsetArray_PastEnd	lib/util_public.mv	Helper Functions
SQL_Search_Clause	lib/dbeng/sql.mv	Database API
STA_Store_Create	features/sta/sta_db.mv	Statistics
STA_Store_Delete	features/sta/sta_db.mv	Statistics
STA_Store_Reset	features/sta/sta_db.mv	Statistics
StandardFields_Insert	lib/dbprim/standardfields.mv	Database API
StandardFields_Load	lib/dbprim/standardfields.mv	Database API
StandardFields_Read	lib/dbprim/standardfields.mv	Database API
StandardFields_Update	lib/dbeng/fields.mv	Database API
StandardFields_Update_LowLevel	lib/dbprim/standardfields.mv	Database API
StandardFields_Validate	lib/util_public.mv	Helper Functions
StandardFields_Validate_NonOptional_Address	lib/util_public.mv	Helper Functions
StartMimeEmail	lib/util_public.mv	Helper Functions
State_Count	lib/dbprim/states.mv	Database API
State_Delete	lib/dbprim/states.mv	Database API
State_Insert	lib/dbprim/states.mv	Database API

Function	File Location/Feature	Module
State_Load_Code	lib/dbprim/states.mv	Database API
State_Read	lib/dbprim/states.mv	Database API
State_Select	lib/util_public.mv	Helper Functions
State_Select_US	lib/util_public.mv	Helper Functions
State_Select_US_Init	lib/util_public.mv	Helper Functions
State_Update	lib/dbprim/states.mv	Database API
StateList_Load_All	lib/dbprim/states.mv	Database API
StateList_Load_Offset	lib/dbeng/states.mv	Database API
Stats_Inc_Hits	features/sta/sta_db.mv	Statistics
Stats_Inc_Orders	features/sta/sta_db.mv	Statistics
Stats_Inc_Products	features/sta/sta_db.mv	Statistics
Stats_Inc_Revenue	features/sta/sta_db.mv	Statistics
Stats_Inc_Visits	features/sta/sta_db.mv	Statistics
Stats_Insert	features/sta/sta_db.mv	Statistics
Stats_Load	features/sta/sta_db.mv	Statistics
Stats_Read	features/sta/sta_db.mv	Statistics
Stats_Update	features/sta/sta_db.mv	Statistics
Stats_Update_Products_From_Basket	features/sta/sta_db.mv	Statistics
Store_Count	lib/dbprim/stores.mv	Database API
Store_Count_License	lib/dbprim/stores.mv	Database API
Store_Create	features/aff/aff_db.mv	Affiliate System
Store_Create	lib/dbeng/create_store.mv	Database API
Store_Create	features/usl/usl_db.mv	Upsale Database
Store_Delete	lib/dbeng/delete_store.mv	Database API
Store_Delete	features/usl/usl_db.mv	Upsale Database
Store_Delete_ID	lib/dbprim/stores.mv	Database API
Store_Insert	lib/dbprim/stores.mv	Database API
Store_Load_Code	lib/dbprim/stores.mv	Database API
Store_Load_ID	lib/dbprim/stores.mv	Database API
Store_Open	lib/dbeng/open.mv	Database API
Store_Read	lib/dbprim/stores.mv	Database API
Store_Update	lib/dbprim/stores.mv	Database API
Store_Update_Encryption	lib/dbprim/stores.mv	Database API
Store_Update_Modules	lib/dbprim/stores.mv	Database API
StoreCountry_Delete_All	lib/dbprim/storecountries.mv	Database API
StoreCountry_Delete_ID	lib/dbprim/storecountries.mv	Database API
StoreCountry_Insert	lib/dbprim/storecountries.mv	Database API
StoreCountry_Load_Alpha	lib/dbprim/storecountries.mv	Database API

Function	File Location/Feature	Module
StoreCountry_Load_ID	lib/dbprim/storecountries.mv	Database API
StoreCountry_Load_ISO_Code	lib/dbprim/storecountries.mv	Database API
StoreCountry_Read	lib/dbprim/storecountries.mv	Database API
StoreCountry_Select	lib/util_public.mv	Helper Functions
StoreCountryList_Load_All	lib/dbprim/storecountries.mv	Database API
StoreKey_Delete	lib/dbprim/storekeys.mv	Database API
StoreKey_Generate	lib/dbeng/keys.mv	Database API
StoreKey_Insert	lib/dbprim/storekeys.mv	Database API
StoreKey_Load	lib/dbprim/storekeys.mv	Database API
StoreKey_Read	lib/dbprim/storekeys.mv	Database API
StoreList_Load_All	lib/dbprim/stores.mv	Database API
StoreList_Load_User	lib/dbprim/stores.mv	Database API
StoreUIModule_Create_Frameworks	Feature storeui	Module API
StoreUIModule_Create_Items	Feature storeui	Module API
StoreUIModule_Create_Pages	Feature storeui	Module API
StoreUIModule_Create_URIs	Feature storeui	Module API
StoreUIModule_Default_Framework	Feature storeui	Module API
StoreUIModule_Dispatch	Feature storeui	Module API
StoreUIModule_Exception	Feature storeui	Module API
StoreUIModule_Screen_Secure	Feature storeui	Module API
StoreUIModule_Thumbnail	Feature storeui	Module API
StoreUtilityModule_Action	Feature util	Module API
StoreUtilityModule_Action_Privileges	Feature util	Module API
StoreUtilityModule_LeftNavigation	Feature util	Module API
StoreUtilityModule_Screen	Feature util	Module API
StoreUtilityModule_Screen_Privileges	Feature util	Module API
StoreUtilityModule_Validate	Feature util	Module API
StoreWizardModule_Action	Feature storewizard	Module API
StoreWizardModule_Content	Feature storewizard	Module API
StoreWizardModule_Icon	Feature storewizard	Module API
StoreWizardModule_Logo	Feature storewizard	Module API
StoreWizardModule_Privileges	Feature storewizard	Module API
StoreWizardModule_Title	Feature storewizard	Module API
StoreWizardModule_Validate	Feature storewizard	Module API
StoreWizardModule_Validate_Step	Feature storewizard	Module API
StructureMembers	lib/util_public.mv	Helper Functions
StyleSheet	admin/ui.mv	Administrative UI
Submit_Config_Insert	features/sbm/sbm_db.mv	Miva Submit

Function	File Location/Feature	Module
Submit_Config_Load	features/sbm/sbm_db.mv	Miva Submit
Submit_Config_Read	features/sbm/sbm_db.mv	Miva Submit
Submit_Config_Update	features/sbm/sbm_db.mv	Miva Submit
SystemModule_Action	Feature system	Module API
SystemModule_Screen	Feature system	Module API
SystemModule_UIException	Feature system	Module API
TaxModule_Calculate_Basket	Feature tax	Module API
TaxModule_Order_Field	Feature tax	Module API
TaxModule_Order_Fields	Feature tax	Module API
TaxModule_Order_Hide_Fields	Feature tax	Module API
TaxModule_Order_Invalid	Feature tax	Module API
TaxModule_Order_Prompt	Feature tax	Module API
TaxModule_Order_Required	Feature tax	Module API
TaxModule_Order_Validate	Feature tax	Module API
TaxModule_ProcessOrder	Feature tax	Module API
TemplateManager_Check_Export_Enabled	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Component_Prerender	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Component_Render_End	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Component_Render_Start	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Export_Path	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Immutable_ManagedTemplate Version	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ImportDest_Path	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Item	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Item_LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Item_Module	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ItemExtension	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ItemExtension_LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ManagedTemplate	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ManagedTemplate _NoDuplicates	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ManagedTemplateVersion	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_ManagedTemplateVersion _LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Page	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Page_Admin	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Create_Page_LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Decrement_ReferenceCount _Transaction	features/tui/tui_mgr.mv	Template Manager

Function	File Location/Feature	Module
TemplateManager_Delete_All_ManagedTemplate_Files	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Delete_All_Module_Items _Transaction	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Delete_Item	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Delete_ItemExtension	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Delete_ManagedTemplate	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Delete_Page	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_Copy	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_ExternalFiles	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_Page	features/tui/tui_ut.mv	Template Utility
TemplateManager_Export_Settings_Create	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_Settings_Delete	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_Settings_Load	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_Settings_Update	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Export_Template	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Get_Basehref_Path	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Get_Export_Path	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Import_Copy	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Import_ExternalFiles	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Import_Template	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Increment_ReferenceCount _Transaction	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Initialize_Item	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Initialize_Item_LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Install_Template	features/tui/tui_mgr.mv	Template Manager
TemplateManager_ItemExtension_Sort_Swap	features/tui/tui_mgr.mv	Template Manager
TemplateManager_ManagedTemplate_Version_Select	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Page_Assign_Item	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Page_Assign_Item_LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Page_Unassign_Item	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Render_Page	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Render_Page_Admin	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Render_Page_LowLevel	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Replace_ManagedTemplateVersion	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Reset_Exception	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Revert_ManagedTemplateVersion	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Throw_Exception	features/tui/tui_mgr.mv	Template Manager
TemplateManager_Update_Item	features/tui/tui_mgr.mv	Template Manager
Tokenize_Name	lib/util_public.mv	Helper Functions

Function	File Location/Feature	Module
TrackingLink_Delete	lib/dbprim/trackinglinks.mv	Database API
TrackingLink_Insert	lib/dbprim/trackinglinks.mv	Database API
TrackingLink_Load_Type	lib/dbprim/trackinglinks.mv	Database API
TrackingLink_Read	lib/dbprim/trackinglinks.mv	Database API
TrackingLink_Update	lib/dbprim/trackinglinks.mv	Database API
TrackingLinkList_Load_All	lib/dbprim/trackinglinks.mv	Database API
TrackingLinkList_Load_Offset	lib/dbeng/trackinglinks.mv	Database API
Trim_Boolean	admin/validate.mv	Administrative UI
TUI_CreateDataFiles	features/tui/tui_db.mv	Template Database
TUI_HTML_Parse_External_File_Source	features/tui/tui_mgr.mv	Template Manager
TUI_HTML_Parse_Tag_Attribute	features/tui/tui_mgr.mv	Template Manager
TUI_HTML_Tag_CharInsideQuotes	features/tui/tui_mgr.mv	Template Manager
TUI_Store_Create	features/tui/tui_db.mv	Template Database
TUI_Store_Delete	features/tui/tui_db.mv	Template Database
TUI_Strip_Leading_Slash	features/tui/tui_mgr.mv	Template Manager
UIException	/merchant.mv	Shopping Interface
UIModule_StoreSelection_Content	Feature storeselui	Module API
UIModule_StoreSelection_Render	Feature storeselui	Module API
UIModule_StoreSelection_Tabs	Feature storeselui	Module API
UIModule_StoreSelection_Thumbnail	Feature storeselui	Module API
UIModule_StoreSelection_Update	Feature storeselui	Module API
UIModule_StoreSelection_Validate	Feature storeselui	Module API
Unique_Code	lib/util_public.mv	Helper Functions
UpdateSessionID	lib/util_public.mv	Helper Functions
Upsell_Delete_Product	features/usl/usl_db.mv	Upsale Database
Upsell_Increment_Score	features/usl/usl_db.mv	Upsale Database
Upsell_Insert	features/usl/usl_db.mv	Upsale Database
Upsell_Insert_LowLevel	features/usl/usl_db.mv	Upsale Database
Upsell_Load_Eligible_ProductList	features/usl/usl_db.mv	Upsale Database
Upsell_Load_Product	features/usl/usl_db.mv	Upsale Database
Upsell_Price	features/usl/usl_rt.mv	Upsale Runtime
Upsell_Product_Read	features/usl/usl_db.mv	Upsale Database
Upsell_Read	features/usl/usl_db.mv	Upsale Database
Upsell_Reset_Score	features/usl/usl_db.mv	Upsale Database
Upsell_Update_Product	features/usl/usl_db.mv	Upsale Database
UpsellList_Load	features/usl/usl_db.mv	Upsale Database
UpsellList_Load_Offset	features/usl/usl_db.mv	Upsale Database
UpsellOptions_Insert	features/usl/usl_db.mv	Upsale Database

Function	File Location/Feature	Module
UpsellOptions_Load	features/usl/usl_db.mv	Upsale Database
UpsellOptions_Read	features/usl/usl_db.mv	Upsale Database
UpsellOptions_Update	features/usl/usl_db.mv	Upsale Database
UpsellProduct_Delete_ID	features/usl/usl_db.mv	Upsale Database
UpsellXProduct_Delete	features/usl/usl_db.mv	Upsale Database
UpsellXProduct_Insert	features/usl/usl_db.mv	Upsale Database
UpsellXProduct_Load	features/usl/usl_db.mv	Upsale Database
UpsellXProduct_Load_Basket	features/usl/usl_db.mv	Upsale Database
UpsellXProduct_Read	features/usl/usl_db.mv	Upsale Database
UpsellXProductList_Load_Product	features/usl/usl_db.mv	Upsale Database
v56_Address_Compatibility	/merchant.mv	Shopping Interface
v56_CustomerAddress_Compatibility	features/cus/cus_rt.mv	Customer Runtime
v56_Order_Create	lib/dbeng/orders.mv	Database API
v56_Order_Load_ID	lib/dbprim/orders.mv	Database API
v56_Order_Read	lib/dbprim/orders.mv	Database API
v56_Order_Update_Total	lib/dbprim/orders.mv	Database API
v56_OrderItem_Insert	lib/dbeng/orders.mv	Database API
v56_OrderItem_Update	lib/dbeng/orders.mv	Database API
Validate_CC_Mod10	admin/validate.mv	Administrative UI
Validate_Attributes	lib/util_public.mv	Helper Functions
Validate_Attributes_DetermineVariant	lib/util_public.mv	Helper Functions
Validate_Code	admin/validate.mv	Administrative UI
Validate_Code_NonBlank_Optional	admin/validate.mv	Administrative UI
Validate_Currency_NonNegative_Optional	admin/validate.mv	Administrative UI
Validate_Currency_NonNegative_Required	admin/validate.mv	Administrative UI
Validate_Currency_Optional	admin/validate.mv	Administrative UI
Validate_Currency_Required	admin/validate.mv	Administrative UI
Validate_Decimal	admin/validate.mv	Administrative UI
Validate_Email	admin/validate.mv	Administrative UI
Validate_Extension	admin/validate.mv	Administrative UI
Validate_FloatingPoint	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Length_NonNegative_Optional	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Length_NonNegative_Required	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Length_Positive_Optional	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Length_Positive_Required	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Length_Required	admin/validate.mv	Administrative UI
Validate_FloatingPoint_NonNegative_Optional	admin/validate.mv	Administrative UI
Validate_FloatingPoint_NonNegative_Required	admin/validate.mv	Administrative UI

Function	File Location/Feature	Module
Validate_FloatingPoint_Optional	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Positive_Required	admin/validate.mv	Administrative UI
Validate_FloatingPoint_Required	admin/validate.mv	Administrative UI
Validate_Group_Name	admin/validate.mv	Administrative UI
Validate_Image_Extension	admin/validate.mv	Administrative UI
Validate_Login	admin/validate.mv	Administrative UI
Validate_Method_Length	admin/validate.mv	Administrative UI
Validate_Passphrase	admin/validate.mv	Administrative UI
Validate_Phone	admin/validate.mv	Administrative UI
Validate_WholeNumber	admin/validate.mv	Administrative UI
Validate_WholeNumber_NonNegative_Optional	admin/validate.mv	Administrative UI
Validate_WholeNumber_NonNegative_Required	admin/validate.mv	Administrative UI
Validate_WholeNumber_Optional	admin/validate.mv	Administrative UI
Validate_WholeNumber_Positive_Optional	admin/validate.mv	Administrative UI
Validate_WholeNumber_Positive_Required	admin/validate.mv	Administrative UI
Validate_WholeNumber_Range_Required	admin/validate.mv	Administrative UI
Validate_WholeNumber_Required	admin/validate.mv	Administrative UI
Validate_Zip	admin/validate.mv	Administrative UI
Wizard_BeginScreen	admin/wizardui.mv	Wizard UI
Wizard_EndScreen	admin/wizardui.mv	Wizard UI
Wizard_FieldError	admin/wizardui.mv	Wizard UI
Wizard_Module_FieldError	admin/wizardui.mv	Wizard UI
Wizard_Module_HideFields	admin/wizardui.mv	Wizard UI
Wizard_Update_Frames	admin/wizardui.mv	Wizard UI
WizardModule_Action	Feature wizard	Module API
WizardModule_Content	Feature wizard	Module API
WizardModule_Icon	Feature wizard	Module API
WizardModule_Logo	Feature wizard	Module API
WizardModule_Privileges	Feature wizard	Module API
WizardModule_Title	Feature wizard	Module API
WizardModule_Validate	Feature wizard	Module API
WizardModule_Validate_Step	Feature wizard	Module API
Zip_Validate	lib/util_public.mv	Helper Functions

APPENDIX C

Deprecated Elements

This appendix lists elements that have been removed from Miva Merchant since API version 5.00 PR5. These elements were deprecated in order to maintain backwards compatibility with previous versions of the software.

Fields

- **cmp_mv_stdorderfields::bill_addr** (replaced with **bill_addr1** and **bill_addr2**)
- **cmp_mv_stdorderfields::orderpayment_id** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::pay_data** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::pay_id** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::pay_secdat** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::pay_secid** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::pay_seckey** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::processed** (used only with v5.5 PR6 compatibility layer)
- **cmp_mv_stdorderfields::ship_addr** (replaced with **ship_addr1** and **ship_addr2**)
- **Order::bill_addr** (replaced with **bill_addr1** and **bill_addr2**)
- **Order::orderpayment_id** (used only with v5.5 PR6 compatibility layer)
- **Order::pay_data** (used only with v5.5 PR6 compatibility layer)
- **Order::pay_id** (used only with v5.5 PR6 compatibility layer)
- **Order::pay_secdat** (used only with v5.5 PR6 compatibility layer)
- **Order::pay_secid** (used only with v5.5 PR6 compatibility layer)
- **Order::pay_seckey** (used only with v5.5 PR6 compatibility layer)
- **Order::processed** (used only with v5.5 PR6 compatibility layer)
- **Order::ship_addr** (replaced with **ship_addr1** and **ship_addr2**)

Files

- **features/sbm/sbm_db.mv**

Functions

- **Decrypt_Order** (replaced with **Decrypt_OrderPayment**)
- **Decrypt_Payment** (replaced with **Decrypt_OrderPayment**)
- **Module_Order_Delete** (replaced with **Module_Order_Delete_Order**)
- **PaymentModule_Authorize** (replaced with **PaymentModule_Runtime_Authorize**)
- **PaymentModule_Process** (replaced with **PaymentModule_OrderPayment_Capture**)
- **SBM_CreateDataFiles**
- **SBM_DeleteDataFiles**
- **SBM_Domain_Pack**
- **SBM_Store_Create**
- **SBM_Store_Delete**
- **ShippingModule_Label_Field** (replaced with **ShippingModule_Label_Shipment_Field**)
- **ShippingModule_Label_Fields** (replaced with **ShippingModule_Label_Shipment_Fields**)
- **ShippingModule_Label_Generate** (replaced with **ShippingModule_Labels_Generate**)
- **ShippingModule_Label_Invalid** (replaced with **ShippingModule_Label_Shipment_Invalid**)
- **ShippingModule_Label_Package_Validate** (replaced with **ShippingModule_Label_Package_Validate_Package**)
- **ShippingModule_Label_Prompt** (replaced with **ShippingModule_Label_Shipment_Prompt**)
- **ShippingModule_Label_Validate** (replaced with **ShippingModule_Label_Shipment_Validate**)
- **ShippingModule_Shipping_Methods** (replaced with **ShippingModule_Basket_Methods**)
- **SKIN_Export_CSS**
- **SkinsComponentModule_Export** (replaced with **SkinsComponentModule_Export_Item**)
- **Submit_Config_Insert**
- **Submit_Config_Load**
- **Submit_Config_Read**
- **Submit_Config_Update**
- **Validate_Attributes** (replaced with **Validate_Attributes_DetermineVariant**)

APPENDIX D

Deleting Orders

The ***XXX*_Order_Delete** functions (**Module_Order_Delete**, **PaymentModule_Order_Delete** and **ShippingModule_Order_Delete**) are called from the **Order Edit** and **Batch Edit** screens. These three functions use the single parameter, **module var**, to pass orders to the module. This means that the web developer must examine the page state (i.e., the global variables) to determine exactly which orders are being deleted.

Note: **Module_Order_Delete** was deprecated and should only be used to ensure compatibility with older versions of Miva Merchant. For API versions 5.51 and above, use **Module_Order_Delete_Order** instead.

On the Order Edit screen, **g.Edit_Order** will contain the ID of the order being deleted. On the Order Batch Edit screen you must examine the arrays **g.Order_Remove[]** and **g.Order_ID[]**.

Miva Script Code Example

The following code example demonstrates the logic for deleting orders via the ***XXX*_Order_Delete** functions (**Module_Order_Delete**, **PaymentModule_Order_Delete** and **ShippingModule_Order_Delete**). Use this example as a guide for determining how orders are deleted.

```
<MvCOMMENT>The top three conditionals below test from where the order is being
 deleted</MvCOMMENT>
<MvIF EXPR = "{ len( g.Edit_Order ) GT 0 }">
    <MvCOMMENT>Delete an order from the Edit Order screen.</MvCOMMENT>
    <MvCOMMENT>g.Edit_Order is the ID of the order that's being removed
     </MvCOMMENT>
    <MvIF EXPR = "{ NOT [ g.Module_Library_DB ].Order_Load_ID( g.Edit_Order,
     l.order ) }">
        <MvFUNCTIONRETURN VALUE = 0>
    </MvIF>

    <MvCOMMENT>
        Delete Module Order Data code goes here.
        Ex: <MvASSIGN NAME = "l.delete_order" VALUE = "{ MyModule_Delete_Order
          ( l.order:id ) }">
    </MvCOMMENT>
<MvELSEIF EXPR = "{ g.Order_Count GT 0 }">
    <MvCOMMENT>Delete orders from the Order Batch Edit screen.</MvCOMMENT>
    <MvCOMMENT>Since Module_Order_Delete is called for each order that should be
     removed, the code below ensures we only run it once.</MvCOMMENT>

    <MvIF EXPR = "{ g.Module_Remove_Order }">
        <MvFUNCTIONRETURN VALUE = 1>
    <MvELSE>
        <MvASSIGN NAME = "g.Module_Remove_Order" VALUE = 1>
    </MvIF>

<MvCOMMENT>This code is responsible for deleting the module data of each order
</MvCOMMENT>

    <MvASSIGN NAME = "l.ord_pos" VALUE = 1>
    <MvWHILE EXPR = "{ ( l.ord_pos LE g.Order_Count ) }">
        <MvCOMMENT>Check the status of each Order Remove checkbox</MvCOMMENT>
        <MvIF EXPR = "{ g.Order_Remove[ l.ord_pos ] }">
            <MvCOMMENT>If the order should be removed; load the order data into
              l.order (structure)</MvCOMMENT>
            <MvIF EXPR = "{ NOT [ g.Module_Library_DB ].Order_Load_ID( g.Order_ID
              [ l.ord_pos ], l.order ) }">
                <MvFUNCTIONRETURN VALUE = 0>
            </MvIF>
```

```
                <MvCOMMENT>
                    Delete Module Order Data code goes here.
                    Ex: <MvASSIGN NAME = "l.delete_order" VALUE =
                     "{ MyModule_Delete_Order( l.order:id ) }">
                </MvCOMMENT>
            </MvIF>

            <MvASSIGN NAME = "l.ord_pos" VALUE = "{ l.ord_pos + 1 }">
        </MvWHILE>
    <MvELSEIF EXPR = "{ g.StoreBatch_Count GT 0 }">
        <MvCOMMENT>Delete orders from the Batches Batch Edit screen.</MvCOMMENT>
        <MvCOMMENT>Since Module_Order_Delete is called for each order that should be
         removed, the code below ensures we only run it once.</MvCOMMENT>

        <MvIF EXPR = "{ g.Module_Remove_Order }">
            <MvFUNCTIONRETURN VALUE = 1>
        <MvELSE>
            <MvASSIGN NAME = "g.Module_Remove_Order" VALUE = 1>
        </MvIF>

    <MvCOMMENT>This code is responsible for deleting the module data of each order
    </MvCOMMENT>

        <MvASSIGN NAME = "l.batch_pos" VALUE = 1>
        <MvWHILE EXPR = "{ l.batch_pos LE g.StoreBatch_Count }">
            <MvCOMMENT>Check the status of each Batch Remove checkbox</MvCOMMENT>
            <MvIF EXPR = "{ g.StoreBatch_Remove[ l.batch_pos ] }">
                <MvCOMMENT>If the order should be removed; load the batch record into
                  l.batch (structure)</MvCOMMENT>
                <MvIF EXPR = "{ [ g.Module_Library_DB ].Batch_Load_ID(
                  StoreBatch_Remove_ID[ l.batch_pos ], l.batch ) }">
                    <MvCOMMENT>Load the order list into l.order (array of
                     structures)</MvCOMMENT>
                    <MvDO FILE = "{ g.Module_Library_DB }" NAME = "l.order_count"
                     VALUE = "{ OrderList_Load_Batch( l.batch:id, l.order ) }">

                    <MvASSIGN NAME = "l.pos" VALUE = 1>
                    <MvWHILE EXPR = "{ l.pos LE l.order_count }">
                        <MvCOMMENT>
                            Delete Module Order Data code goes here.
                            Ex: <MvASSIGN NAME = "l.delete_order" VALUE =
                             "{ MyModule_Delete_Order( l.order[ l.pos ]:id ) }">
                        </MvCOMMENT>
```

```
                    <MvASSIGN NAME = "l.pos" VALUE = "{ l.pos + 1 }">
                </MvWHILE>
            </MvIF>
        </MvIF>

        <MvASSIGN NAME = "l.batch_pos" VALUE = "{ l.batch_pos + 1 }">
    </MvWHILE>
</MvIF>
```

Index of Functions

B

C

D

E

F

I

J

L

M

P

R

S

T

U

W