MIVA MERCHANT

Miva Merchant 5.5 API

What's New in PR8

version 1.0

Table of Contents

CHAPTER 1 *Overview*

There are two ways that Miva Merchant changes the Module API:

- Modify the behavior of existing features by changing, adding or removing functions to a feature;
- Add new features.

Modifying existing functions is avoided when Miva Merchant changes the Module API. Instead, new functions are added and old functions are deprecated. This is done so that the same module can support the current version of the software while remaining backwards compatible with previous versions. Since only one function can exist in a module with a given name, reusing an existing function name and changing the parameters is not feasible as it would not support both versions of the software simultaneously.

When the behavior of a particular module feature is changed, the Miva Merchant software looks at the supported API version of the module to determine what version of the functionality is appropriate for that module. As of **pr8-update-4**, the highest version of the Module API is 5.71.

This document describes new features, deprecated features and changed behavior in Miva Merchant 5.5 Product Release 8.

CHAPTER 2

New Functions in Existing Module Features

This chapter describes new functions that were added to existing module features in PR8.

Feature batchreport

The **batchreport** feature was modified in PR8 to support the following additional functionality:

- The ability to have multiple reports (each identified by a report code) handled by a single module.
- The ability to run reports on a list of orders that are not necessarily part of a batch rather than only on orders that have already been batched.
- Shipment reports – the functions for shipment reports are functionally identical to the functions for order reports.

Four new functions were added to the **batchreport** feature in PR8. These functions are described below.

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_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)

Feature import

The import subsystem was completely redesigned in PR8. In previous versions, the **import** modules were responsible for doing all of the file I/O and parsing as well as maintaining their own user interface.

In PR8, the following changes were made:

- A standardized mechanism for handling I/O was implemented
- Delimited (CSV, tab, etc.) file parsing
- Consistent user interface for both configuration and execution of an import

Eighteen new functions were added to the **import** feature in PR8. These functions are described below.

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 may 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_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_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 may control the values of the status fields by using the **Import_Session_StatusField_Get**, **Import_Session_StatusField_Set** and **Import_Session_StatusField_Increment** 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 may 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

Feature payment

The **payment** module API was modified in PR8 to handle field validation and prompts differently in the administrative interface than in the shopping interface. For example, administrators often do not have the CVV2 code for a credit card. The new functionality allows the CVV2 field to be required for the shopping interface but optional for the administrative interface.

Eight new functions were added to the **payment** feature in PR8. These functions all mirror existing **PaymentModule_Payment_*XXX*** functions and have identical input and output. The only difference is when and where the functions are called. For example, **PaymentModule_Order_Authorize_Fields** is identical to **PaymentModule_Payment_Fields**, except the **Order_Authorize** function is called when authorizing a credit card from the administrative interface and the **Payment** function is called when a shopper is checking out.

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 which 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 which receives an array of supported payment methods

Return Value

The count of elements in **methods**.

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 which 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 which 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_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 currently visible tab code
order	The **Order** record of the order being displayed

Return Value

1 on success

0 on error

Feature shipping

PR8 Update 4 introduced the ability to calculate a packaging solution at checkout time — for example, how many boxes and what size boxes are required to ship the order. Many third party shipping integrations, such as FedEx® and UPS®, now require this information in order to provide a rate estimate. The changes to the shipping API in PR8 Update 4 allow the module to be informed of the packing solution generated by the core software.

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_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 currently visible tab code
order	The **Order** record of the order being displayed

Return Value

1 on success

0 on error

Feature shipping_label

Extensive changes and improvements were made to the **shipping_label** module in PR8 Update 4, including:

- Support for multiple packages and multiple shipping labels from a single order shipment
- Shipments can be reconfigured from the Generate Shipping Label dialog
- Improved label display screen with print button

Fourteen new shipping_label functions were introduced. These functions are described below.

ShippingModule_Label_Boxes

This function allows a module to return a list of shipper-specific boxes that may 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_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.

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 may 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_Field, ShippingModule_Label_Package_Fields, ShippingModule_Label_Package_Invalid, ShippingModule_Label_Package_Prompt, ShippingModule_Label_Package_Validate**).

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.

Supported API Version

5.71 and higher (new in PR8 Update 4)

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 may report validation errors either through **ShippingModule_Label_Package_Invalid** or by calling the **FieldError** function.

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

The textual prompt for the field. HTML is permitted.

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 may report invalid fields through **ShippingModule_Label_Shipment_Invalid** or by calling the **FieldError** function.

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

Feature vis_affil

This feature allows modules to add tabs to the **Add/Edit Affiliate** screen.

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 currently visible tab code
affiliate	The record of the affiliate being edited or NULL if adding an affiliate

Return Value

1 on success

0 on error

Feature vis_affilbe

This feature allows modules to add tabs to the **Affiliate Batch Edit** screen.

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_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 currently visible tab code

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>:<Description>
```

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)

Feature vis_category

This feature allows modules to add tabs to the **Add/Edit Category** screen.

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 currently visible tab code
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

Feature vis_categorybe

This feature allows modules to add tabs to the **Category Batch Edit** screen.

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_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 currently visible tab code

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>:<Description>
```

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)

Feature vis_cust

This feature allows modules to add tabs to the **Add/Edit Customer** screen.

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 currently visible tab code
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

Feature vis_custbe

This feature allows modules to add tabs to the **Customer Batch Edit** screen.

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_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>:<Description>
```

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)

Feature vis_domain

This feature allows modules to add tabs to the **Domain Settings** screen.

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 currently visible tab code

Return Value

1 on success

0 on error

Feature vis_fulfill

This feature allows modules to add tabs to the **Order Fulfillment Settings** screen.

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

Feature vis_log

This feature allows modules to add tabs to the **Logging Settings** screen.

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

Feature vis_order

This feature allows modules to add tabs to the Legacy Order Processing **Edit Order** screen and the new **Manage Orders** screen.

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

Feature vis_payment

This feature allows modules to add tabs to the **Payment Settings** screen.

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

Feature vis_product

This feature allows modules to add tabs to the **Add/Edit Product** screen.

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

Feature vis_productbe

This feature allows modules to add tabs to the **Product Batch Edit** screen.

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_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>:<Description>
```

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)

Feature vis_shipping

This feature allows modules to add tabs to the **Shipping Settings** screen.

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

Feature vis_store

This feature allows modules to add tabs to the **Edit Store** screen.

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

Feature vis_system

This feature allows modules to add tabs to the **System Extension Settings** screen.

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

Feature vis_util

This feature allows modules to add tabs to the **Store Utility Settings** screen.

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

CHAPTER 3

New Module Features

This chapter describes new module features that were added in PR8 and the functions they contain.

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 may be put in each box.

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

0 if any module-provided field is invalid

Note: Validation failures may 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

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**.

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.

Supported API Version

5.70 and higher

Syntax

Module_Clientside(module var)

Parameters

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

Return Value

None. This function must not return a value.

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 may 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. Must always be set to **Module**.
- **Module_Code** – String (required). The code of the module that provides the JSON function.
- **Module_Function** – String (required). 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.

Module_JSON

This function is called when a module function is requested through **json.mvc**. The module-specific function will be indicated by **g.Module_Function**. 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

5.70 and higher

Syntax

Module_JSON(module var)

Parameters

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

Return Value

Ignored

Feature not_cat

Modules which implement this feature are notified when a category is created or deleted.

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.

Supported API Version

5.70 and higher

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.

Supported API Version

5.70 and higher

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 deleted

Return Value

Ignored

Feature not_cust

Modules which implement this feature are notified when a customer is created or deleted.

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

5.70 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

5.70 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 category being deleted

Return Value

Ignored

Feature not_prod

Modules which implement this feature are notified when a product is created or deleted.

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

5.70 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 category 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

5.70 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 category being deleted

Return Value

Ignored

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.

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_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 may 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 may 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 may report invalid fields through **ReportModule_Invalid** or by calling the **FieldError** function.

APPENDIX A

Deprecated Elements

This appendix lists the elements that were deprecated in PR8.

Deprecated Functions

The following functions were deprecated and will never be called for API version 5.71 and above:

- **ShippingModule_Label_Field**
- **ShippingModule_Label_Fields**
- **ShippingModule_Label_Generate**
- **ShippingModule_Label_Invalid**
- **ShippingModule_Label_Prompt**
- **ShippingModule_Label_Validate**

Index of Functions

B

I

M

P

R

S