ww.dll API Reference

Version 4.0.58

Siriusware, Inc.
302 Camino de la Placita
Taos, NM 87571
575.751.0633
www.siriusware.com/docs
www.siriusware.com/training
google.siriusware.com

Copyright

ww.dll API Reference

Introduction

The Salesware `ww.dll` is a COM object, a `.dll` that is instantiated and connects to the SiriusSQL database. It's typically run inside of Microsoft's COM+ (in order to handle pooling). It's installed on your E-Commerce server and you can make calls to it using Siriusware's XML format to retrieve information or to submit information to the database. Calls are available to, for example:

- Retrieve a list of items and prices
- Retrieve guest information
- Create a guest
- Create a sale

What you can't do right now is edit existing reservations. Once you plug the reservation into the system and it is processed by Sales Host, it can only be edited from within the system (at a Classic salespoint).

The Salesware `ww.dll` is used to provide Internet (TCP/IP) access to the SiriusSQL database. `ww.dll` functions provide an interface to perform many of the operations that you perform from a Classic salespoint: pass and ticket sales, guest lookups and modifications, private lesson reservations, In-House Cards purchases, and so on. The Salesware E-Commerce and Rentals (Self Entry component) modules use `ww.dll` functions, but the `ww.dll` functions are also available to clients who wish to write their own custom Internet applications. `ww.dll` communications with the SiriusSQL database can be across a LAN, or across the web. With `ww.dll`, you can create a small set of web pages to provide very specific functionality, or you can duplicate most of the functions of a Classic salespoint, but with a web interface.

To use this manual, you need a copy of the SiriusSQL Data Dictionary, available from Siriusware. While SiriusSQL tables and fields are defined to some extent in this document (especially in the case of calculated fields that are passed back as the result of execution of a stored procedure), the Data Dictionary is the final word on the structure, makeup, and definition of the SiriusSQL database.

To use this document, you need to understand how a Classic salespoint operates. You then use the `ww.dll` functions to mimic these operations from custom web pages that implement some mechanism for acquiring items, such as a shopping cart. You can learn how a Classic salespoint operates by reading the *Salesware User and Ticketing Guide for Beginning Users*, *Salesware User Guide for Advanced Users* and the other documents available from http://www.siriusware.com/docs. Most developers use ASP.NET and VB.NET from within a Microsoft Visual Studio project to leverage the API and generate the web pages that are then read by an Internet Explorer, Netscape or FireFox browser.

API summary

The following table summarizes the API.

Function	Description
`acceptliability`	Writes a record into the `gst_actv` table to denote that a guest accepted a liability form.
`checkpassword`	Checks the password of the indicated record.
`clearitemres`	Eliminates records from the `item_res` table.
`createaccess`	Convenience call. Calls `createpass` with `<access>1</access>` specified.
`createpass`	Updates the `gst_pass` or `access` table with new pass *or* new access information, depending on whether `<access>1</access>` is specified.

decryptdata	Decrypts the data passed in.
encryptdata	Constructs the key, hashes it, and then encrypts the data.
encryptdata2	Constructs the key, hashes it, and then encrypts the data. Gets client number from database.
execsp	Makes a stored procedure call, essentially of the form `EXECUTE` followed by the `spname` and then the `params`.
getaccessinfo	Calls `getpassinfo` with `<access>1</access>` specified.
getdciavail	Finds available booking times for Private Instruction.
getdwinfo	Returns information about an In-House Cards card.
getguest	Retrieves a guest record, joined to the record(s) for the address(es).
gethash	Constructs a one-way hash of the data passed in.
getitem	Gets an item.
getitemexpanded	Gets an item, including modifiers and other information not available using `getitem`.
getitemtree	Takes the head node, the `<date_time>` of interest, and a list of fields to be returned (optional) and gets the next level of the item tree.
getliabilityinfo	Looks up the description, minimum age to accept, the expiration days/date, and the text of the liability form(s) assigned to a particular item via `<department><category><item>` parameters.
getlog	Causes a log file to be generated.
getmatrixinfo	Returns a table of matrix information for a Retail item.
getmax4saleinfo	Returns Max4Sale information.
getmods	Gets the applicable modifier records for a DCI.
getparents	Gets items that are allowed as parents to the list of modifiers given in the `<hasmod>` tag.
getpassinfo	Retrieves information about new pass/access records and modified pass/access records.
getserverinfo	Returns the path, filename and internal version number of the `.dll`.
getuniquekey	Takes a `<key>` tag with a field name from the `sequence` table and returns a new `address id.`
getverbositylevel	Gets current verbosity level.
getversion	Returns the internal version number of the `.dll.`
getwwsaleid	Retrieves a new `wwsale_id` for use in tracking a sale.
insert	Constructs a SQL `INSERT` statement.
lookupguests	Looks up guests based on the field(s) passed in.
lookuppasses	Looks up passes based on the field(s) passes in.
modifyaccess	Calls `modifypass` with `<access>1</access>` specified.
modifyaddress	Modifies existing guest information.
modifygstrent	Updates information for the guest for the Rentals process.
modifyguest	Modifies information for an existing guest.
modifypass	Adds the ability to modify existing pass/access records to accommodate voids, returns, updated valid and blackout dates, and updated usage.
newaddlink	Creates a new `addlink` record.
newaddress	Creates a new record in the `address` table.
newbooking	Adds a booking to the Private Instruction schedule.
newguest	Creates a new guest record in the `guest` table.
newitemres	Checks Max4Sale limits for an item with real time inventory and creates a record in

	the item_res table if there is sufficient quantity available.
processsale	Takes a description of a sale and queues it for processing asynchronously.
select	Constructs a SQL select statement.
setpassword	Sets the password of the indicated record.
setverbositylevel	Sets the verbosity level.
sleep	Sleeps the indicated number of seconds.
update	Constructs a SQL UPDATE statement.
writefile	Writes a file to a specified filepath containing the specified file data.

Setup

General setup

The ww.dll file needs to be registered. A file called ww.reg is installed in the same directory as the the one where ww.dll resides; this file dictates settings for all instances of the .dll. If you change a setting in ww.reg, you need to double-click it to update the registry with the changed settings, and then re-start ww.dll. For complete information on how to install and configure ww.dll, see the *Salesware E-Commerce* documents. For complete information on how to update ww.dll, see the *Updating Salesware Modules* document.

Note: ww.dll requires the file msvcr71.dll. This file is installed by the Install_Siriusware_CommonFiles_xxxx.msi installer, which is run when you first install Salesware. Other Microsoft-compatible applications install it as well. If the .dll is not present when you attempt to register ww.dll, you get an error message.

ww.dll registry settings

The following table lists and describes the available ww.dll registry settings.

Section in ww.reg file	Setting name	Default value	Description
E-Commerce\CC	AccountID	"" (empty string)	For OCV. Is sent in the <acct_num> field when getting an approval. If the <AccountID> tag is sent from the web page, ww.dll uses that setting instead. Also see altAccountID key documentation in the *Salesware E-Commerce* document.
E-Commerce\CC	DPSAuthSSL	FALSE	Used with DPSAuthSSL (Payment Express). Username and password are provided by DPS. Password is encrypted and rewritten to the registry the first time it is read by ww.dll. The E-Commerce devicetype must be set to 18 for DPSAuthSSL. Using DPSAuthSSL
E-Commerce\CC	DPSAuthSSLUsername	"" (empty string)	
E-Commerce\CC	DPSAuthSSLPassword	"" (empty string)	

			for E-Commerce allows refund matching but Reservation Headers must be enabled both for the E-Commerce Pages and Sales Host to make refund matching work. DPSAuthSSL works both with and without the `PreAuth=FALSE`/`WebPreAuthCompletion=FALSE` settings. Note that the `ww.dll` `PreAuth` setting must match the value of the `WebPreAuthCompletion` setting on Sales Host.
`E-Commerce\CC`	`Enabled`	`FALSE`	Whether communications with the credit card server specified by `Protobase` or `OCV` is enabled. As of release 4.0.56 of the `.dll`, this setting is ignored. Credit card charging is always enabled. Credit card charging can be disabled by entering an incorrect ProtoBase server into the `Protobase` setting.
`E-Commerce\CC`	`OCV`	"" (empty string)	IP address and port for communication with the OCV (Ingenico) server – e.g., `127.0.0.1:3005`.
`E-Commerce\CC`	`Operator`	"" (empty string)	The operator to use when communicating with the credit card server specified by `Protobase`. This is typically the operator or salespoint specified using `DefaultInfo`. Must be 8 characters or less; if it is longer, then it is truncated.
`E-Commerce\CC`	`Protobase`	"" (empty string)	IP address and port for communication with the ProtoBase server – e.g., `Protobase=12.96.199.182:4209`.
`E-Commerce\CC`	`TerminalID`	"" (empty string)	The type of terminal that is to be used in communications with ProtoBase or OCV. A `<termid>` tag inside of the `<settlement>` tag in the `processsale` call can contain a different terminal ID to be passed to ProtoBase, in which case this setting is overwritten. Typically, the ID is `RET`, for Retail.
`E-Commerce\CC`	`Timeout`	`120`	The amount of time, in seconds, before `ww.dll` stops attempting to communicate with the server.
`E-Commerce\Preferences`	`ConnectionString`	"" (empty string)	The string that `ww.dll` uses to communicate with the SiriusSQL server. After it has been entered during installation, `ww.dll` encrypts it so that only `ww.dll` can read it.

E-Commerce\ Preferences	DefaultInfo	"" (empty string)	Used to specify default fields that ww.dll can reference. Typically set to <operator>WEBOP</operator><salespoint>WEBSP</salespoint>.
E-Commerce\ Preferences	EventLogVerbosity	2	Sets event log verbosity. Can be 0, 1, 2, 3, 4, or 5. With this set to 1, only errors are logged. With this set to 2, warnings are also added into the log. At 3, every call is logged. With this set between 3 and 5, additional logging of ww.dll startups and stops, calls made to ww.dll, and OCV and socket opening/closing information is logged. *Note*: Setting this to 4 or 5 should be used only for troubleshooting purposes. A large event log is created and there is the possibility of performance degradation.
E-Commerce\ Preferences	GuestFilter	"" (empty string)	Additional criteria for guest lookup. For example, GuestFilter=role_no<>2. This ensures that only guests without a role_no of 2 are returned.
E-Commerce\ Preferences	LogDir	.. (current directory, where ww.dll is installed)	Specifies location of the log file ww_log.txt. For example, LogDir=c:\ causes ww.dll to write the log file to the root of the c: drive.
E-Commerce\ Preferences	SaveSize	100000	Used to specify how much information is retained by ww.dll. If the log file saved in memory exceeds twice SaveSize, then it is trimmed back to the amount specified by this setting. For example, SaveSize=100500 saves only the last 100,500 bytes.
E-Commerce\ Preferences	Secure	FALSE	Enables/disables encryption (except for the ConnectionString, which is always encrypted).
E-Commerce\ Preferences	SetAppRole	FALSE	If TRUE, sets the SiriusApp role on startup. This should only be set if the sirius user is being used and connection pooling is disabled. See http://support.microsoft.com/support/kb/articles/q229/5/64.asp for information on how to disable connection pooling. The string: ;OLE DB Services= -2 must be added to the ConnectionString for this to work.

E-Commerce\ Preferences	SingleLog	FALSE	Used for troubleshooting purpose. Check with Siriusware Technical Support before using.
E-Commerce\ Preferences	TestCard	"" (empty string)	Credit card to use for testing. ww.dll does not attempt to communicate with the credit card server if a purchase is made using the test card defined here. Typically, 5454545454545454 is used. This setting should not be used if your web Sales Host is processing the sale – if it is, the Sales Host returns an error when it goes to process the sale.
E-Commerce\ Preferences	VerbosityLevel	0	Specifies how much information is written into the log file ww_log.txt. Valid settings are 0 (least verbose) to 5 (most verbose).

General Use

ww.dll needs to be instantiated as ww.main. In VBScript, this looks like:

```
Set objT = CreateObject("ww.main")
```

All calls made go through the invoke method and must be formatted as XML. A <func> tag contains the actual function to execute; the other information passed depends on the call. For example,

```
a=objT.invoke("<func>getversion</func>");
```

Likewise, the return values are all XML with the exception of the first 4 characters, which are always either OK : or ERR:.

OK : is followed by an XML return string.

ERR: is followed by a text string which identifies the error.

Note on the <fields> tag

Many calls allow use of a <fields> tag to specify what information is returned. The log can be used to see the actual SQL statement passed to the database in order to see what fields are passed by default and what aliases are used (e.g., i for items, it for itemtree). This is for advanced users only and this field must be used carefully. Here's an example of how the <fields> tag is used:

```
<func>lookupguests</func><first_name>matt</first_name><fields>g.guest_no, g.first_name,
a.address</fields>
```

For example, when using the `getitemtree` call with the `<calcprice>` tag set, if you specify the `<fields>` tag and do not include pricing fields, the prices are all calculated to be zero. Most of these problems come from the "composite" calculations such as the <`max4sale`> tag and the <`calcprice`> tag.

Note on the <avail> tag

The `<avail>` tag returns information on whether a particular item is available based on Max4Sale or Points4Sale restrictions for a certain date or date range. If the tag is absent or is specified as `<avail>0</avail>`, nothing is calculated. The `<avail>...</avail>` tag can contain a single number or date, multiple numbers or dates each separated by a comma, a range of numbers or dates, or a combination of numbers, dates and ranges using the following guidelines:

- `<avail>1</avail>` means calculate availability for the current day
- `<avail>1-7</avail>` means to calculate for today (`1`) until 6 days from now (`7`)
- `<avail>1,3,7</avail>` means to calculate for today, 2 days from now, and 6 days from now
- `<avail>8/29/2008-9/1/2008,9/15/2008-9/30/2008</avail>` means to calculate for 8/29/2008 through 9/1/2008 and 9/15/2008 through 9/30/2008

Comma separated values and ranges can be mixed:

`<avail>1-4,6,7</avail>` is valid as is `1-7,10/1/2008-10/12/2008`. Also, the identifiers `bm` and `em` are in place and mean beginning-of-month and end-of-month respectively. So, `<avail>bm-em</avail>` gives all values for the current month.

Values are returned in the `avail_info` field and are in an XML list using the Julian Day, e.g.:

```
avail_info='[39793]0[/39793][39794]0[/39794][39797]1[/39797]'
```

A `1` indicates availability while a `0` indicates no availability.

Conventions when using duplicate field names

In some calls, such as `getitemexpanded`, you might have to pass in field names that are the same but that exist in different tables. In these cases, in order to distinguish which field belongs to which table, you need to append the initial letter to the table name when passing in the argument. If you use this convention with a single argument you must use it with all other field names in that call as well, regardless of whether they are duplicate field names. For example:

- `g.guests_no` (`guests` table)
- `i.item_id` (`items` table)
- `t.trans_no` (`transact` table)
- `a. address_id` (`address` table)

In addition, field names such as c60 that do not appear in the data dictionary are automatically generated when there are duplicate field names in joined datasets. The description in the data set tells exactly what the duplicate field is. For example:

```
<s:AttributeType name='c52' rs:name='department' rs:number='53' rs:nullable='true'
                 rs:writeunknown='true'>
           <s:datatype dt:type='string' rs:dbtype='str' dt:maxLength='10'
                 rs:fixedlength='true'/>
</s:AttributeType>
```

Tags in <sale>

The `<sale>` tag is used to process a sale with the `processsale` function. The indentation indicates the nesting of the tags.

<sale>

This contains all the information for a sale.

<res_hdr>

Holds fields for the resrvatn table.

Note: reserv_no should not be filled in.

`<guest_no>` *If provided, first_name, last_name, are filled in.*

The following fields should have a pri_key passed for the value.

<accomodat>
<base_lodge>
<mktg_code>
<wrap_code>
<pickup_loc>
<srce_code>
<user_code1>
<user_code2>
<user_code3>
<res_status>

<settlement>

<device type> specifies the type of charge used. When <card_no> is present, it is assumed to be 15 (ProtoBase) unless otherwise indicated.

<devicetype>	**Process via:**	**Currently supported?**
5	Directnet	No
11	PMS Server	No
14	Tender Retail	No
15	ProtoBase *Note*: When a call is made from `ww.dll` to ProtoBase for a credit card authorization, the tag `<proc_online></proc_online>` is returned in the XML. If the tag contains a `1` it means that ProtoBase was able to communicate with the bank and if it contains a `0` it means that ProtoBase was unable to communicate with the bank. Online transactions from the E-Commerce pages send tran type 22 to the ProtoBase server.	Yes
16	Debitware	Yes

17	Ingenico Australia	Yes
18	Payment Express	Yes

<devicetype>
Typically holds a <swipe> tag plus any other fields for credit card processing, but can also hold a payment nickname (e.g., CASH). Can hold either just the description of the payment type to be used for the web payment, or the following tags are required for processing via ProtoBase.
<pmt*n*>
Used to specify multiple forms of payment. For example, `<pmt1>...`*info*`...</pmt1><pmt2>...`*info*`...</pmt2>`*. Payments can be made with one or more credit cards, one or more In-House Cards items, or a combination of credit cards and In-House Cards items. For sales that are paid for with a combination of credit cards and In-House Cards items, the In-House Cards payment must come first so that it can be unwound correctly if the credit card is declined.*
<card_no> *15 or 16 digit card number.*
<exp_date> *4 digit expiration date in MMYY format.*
<card_addr> *Billing address.*
<card_addr2> *Billing address 2.*
<card_zip> *Billing zip code.*
<ship_zip> *Shipping zip code.*
<card_city> *City.*
<card_cntry> *Country.*
<card_fname> *First name.*
<card_lname> *Last name.*
<card_state> *State.*
<card_cvv2> *3 digit code on the back of the card.*
<auth_no> *Authorization number received from ProtoBase.*
<batch_no> *Batch number received from ProtoBase.*
<BatchSeq> *Batch sequence.*
<host_code> *Host code.*
<merit_cd> *Merit cd.*
<org_date> *Org date.*
<org_time> *Org time.*
<pb_code> *PB code.*
<pr_info> *PR info – encrypted.*
<swipe> *Swipe – encrypted.*
The following fields can be provided, but if they are not included, the item sections of the sale are totaled on <finalprice> and <tax> to obtain these numbers.
<amount> *Total amount (including tax).*
<tax> *Tax only.*
Optional field(s).
<termid> *ProtoBase terminal id (defaults to setting in ww.ini file if absent).*
The following fields are added by the system.
<item_codes> *4 character each, up to 3 code1, code2, code3*
<item_descrip> *dci1,dci2,dci3 (30 characters)*

<salenotes>
Notes about the sale. Contents of this tag are written into the `salenote.notes1` *field.*

<save>
The typical action is to finalize the sale. If this tag is present, the sale will be saved instead.
<first_name>

<last_name>
<phone>
<descrip1>
<descrip2>
<guest_no>

<special>

Sale level special to apply. The following fields can be specified.

<name> *10 character nick name of special.*

<disc_pct> *Percentage discount for special.*

<disc_flat> *Flat discount for special.*

<account>

This is the account that the sale will be applied to. If an <invoice> tag is also passed, a specific invoice must be present. If <invoice> is not passed, a new invoice will be created.

<acct_name>

If this account is not present, it will be created.

<contact><full_name><email><address>...

All the fields in the accounts table can be passed as well. They will override whatever is in the table if present.

<invoice>

The invoice used will always be associated with the account passed (if the <invoice> tag is passed, the <account> tag must also be present).

<account>

This must be filled in with the <acct_name> from the <account> tag.

<invoice_no>

Only passed if a specific invoice is to be selected (neither the account nor the invoice can be new in this case).

<descrip1><descrip2><email><address>...

All the fields in the invoice table are available to be passed. They will override whatever is in the table if present.

<item>

This is the parent tag for an item in a sale.

<dci>

This contains a 30 character string with the exact department, category, and item for the item. This is also used for the item description passed to ProtoBase.

<invent_id>

This is also used for the item code passed to ProtoBase.

<val_info>

This is used to pass validation information. For example:

```
<item><val_info><0><entry>327003010</entry></0></val_info></item>
```

The <0>,<1>, etc., are for the differing quantities. For a quantity of 2, the code would be:

```
<val_info><0><entry>327003010</entry></0>
          <1><entry>328003010</entry></1></val_info>
```

<qty>

Quantity of the item to be sold.

<dw_info>

<dw_info> must be inside of tags indicating which item quantity is referred to:

```
<qty>1</qty><0><dw_info><pass_no>...</dw_info></0>
<qty>2</qty><0><dw_info><pass_no>...</...</dw_info></0>
            <1><dw_info><pass_no>...</...</dw_info></1>
```

<pass_no> *DW pass number.*

<invoice_no> *DW invoice number.*

<crl> *Credit limit.*

<spld> *Spending limit per day.*

<splt> *Spending limit (total).*

<rel> *Reload amount.*

<prel> *Preload amount.*

<special>
<name> *10 character nick name of special.*
<disc_pct> *Percentage discount for special.*
<disc_flat> *Flat discount for special.*
<finalprice>
Price to use for the item – otherwise it will be whatever is calculated for the date-time. (It's with tax.)
<tax>
Tax amount for the item.
<date>
date/time for the item. Format is "MM/DD/YYYY HH:MM PP". So, it could be "06/01/2000 02:24 PM". Otherwise it will be whenever it is sold.
<message>
Line item message to be applied to the item.
<validate>
Used by certain clients to over-ride the specification of the item – 0,1=none 2=reqd 3=opt.
<item_type>
Used to over-ride the specification of the item 0,1=none 2=guest 3=pass 5=rental 6=private 7=pods.
<gst_pass>
If more than one guest in the sale, <guest> is the first guest, then <gst_pass2>, <gst_pass3> etc.
<swipe_no>
Swipe_no to be attached to the pass attached to this item, if applicable.
<guest_no> <pass_no> <passprefix>
Will be copied straight into the transact record produced.
<guest>
Typically only contains a <guest_no> tag.
If more than one guest in the sale, <guest> is the first guest, then <guest2>, <guest3> etc.
<mod>
Contains information (identical to info that can be used inside an <item> tag for a modifier. More than 1 <mod> tag can be used inside 1 item. For example:

```
<item><dci>AAA-MATT  AAA       CANDY</dci><qty>1</qty>
      <mod><dci>AAA-MATT  AAA       SPRINKLES</dci></mod>
      <mod><dci>AAA-MATT  AAA       SYRUP</dci></mod>
</item>
```

<print>
<do_on_sale>

<email>
<to>
<from>
<name> *Display name.*
<address> *Actual address.*
<cc>
<bcc>
<subject>
<body>
<html> *1 or 0. 1 if the body is in html and it's meant to be sent that way.*
<pdf> *Wait until PDF(s) are printed and attach them to the e-mail;this is the number of PDF(s) to wait for.*
<pdf_t>

<ww_tix> *Print At Home Tickets.*
<pdf_t>

<purchaser> *Purchaser.*

Example invocation:

```
SaleText= <settlement>
      <auth_no>03572A</auth_no>
      <batch_no>271</batch_no>
      <BatchSeq>271</BatchSeq>
      <card_addr>14 SURREY LANE</card_addr>
      <card_addr2></card_addr2>
      <card_city>MENDHAM</card_ city>
      <card_cntry>USA</card_cntry>
      <card_fname>ANDREW</card_fname>
      <card_lname>BENEDICT</card_lname>
      <card_state>NJ</card_state>
      <card_zip >07945</card_zip>
      <host_code>00</host_code>
      <merit_cd>00</merit_cd>
      <org_date>062206</org_date>
      <org_time>183722</org_time>
      <pb_code>0 000</pb_code>
      <pr_info>***605098abfc7093126982bd02f710e2dc922d38
47015227b19ebcdddae7a84005a7fc34bf69620ef2a0b7c8f17749e88a7a8c8
020ddd12aee31b4b1e013fb48602635ff0b2c75c26ecd30f1f90f8e921ac6ce5
89a96ddcbfa69cc6be513ede2c896d50cd3c57da3ce736aa6ce865c5bcd0f224
c17a967df9e8524e1fb73d58abba8e5cad436f9c8c080a2f01c1285e50be86fa
45c1d61b653e7a0c75203ffa3b0699ee2ca440a68aa6ed02591fc24f424a09e
2f928c896fce7103341b574f9c5fb07795d583764cd375541f1dc44edfee2949
46c50da89527963ea1b1c74aea41cc04f5b0f333f544ba9140aeb53297f9df8
9456ada8a5a55005520639f3b3b90634c9f03e5654d85fd3b9b658a44a180a
d5075dce3afd966f97ed8337a4ddf57a958b1933e18f2dfda4282ad0c2832be
d6a5a0dc9d752443f9c2333292e4a3f675f611adb7c99c899141cee670167a7 17ef15e15</pr_info>
      <swipe>***7b489aaac548980a6a83bc1bf91ddfe5955
f3c4107516bb89db7dbda</swipe></settlement>
      <item><dci>WPGA G/A PAHWPGA </dci>
            <qty>2</qty>
            <finalprice>29.95</finalprice>
            <date>06-22- 2006</date>
            <message></message>
            <print>T</print>
            <do_on_sale>SELEC TLAST()FORCEFINALIZE()</do_on_sale></item>
      <item><dci>WPCHILD WPCHILD PAHWPCHILD</dci>
            <qty>3</qty>
            <finalprice>24.95</finalprice>
            <date>06- 22- 2006</date>
            <message></message>
            <print>T</print>
            <do_on_sale>SELEC TLAST()FORCEFINALIZE()</do_on_sale></item>
      <email L=1566>
            <to>ABENEDICT@OPTONLINE.NET</to>
            <from><name>Camelback Ski
            Corporation</name><address>estore@skicamelback.com</address></from>
            <bcc>ray@skicamelback.com,estore@skicamelback.com,louise@skicam
            elback.com,easure@skicamelback.com</bcc>
            <subject L=44>Your Print@Home Tickets - Sale # 3689000000</subject>
            <body L=1204>Dear ANDREW BENEDICT Thanks for your purchase. Your order has
            been recieved and will soon be processed and charged. You may be recieving one
            more email once the order has been finalized. Your sale id is 3689000000. To
            retrieve your Print@Home tickets, please click (or copy/paste into your
            browser) the link below: <https://secure.skicamelback.com/GetTix.aspx?req=KjVI%2bth7%
            2fb291HmwEcUBdg%3d%3d> You will need Adobe Acrobat Reader (available free
            from http://www.adobe.com) to view or print your items. Please Note: All
            Print@Home ticket(s) will be emailed to the email address you provided.
            Treasure Packs and Gift Cards will be mailed via USPS to the address you
            provided. Season Passes and Camelcards will be available for pickup at our
            Welcome Center Guest Services Desk. Lift Ticket and Rental Orders can be
            picked up at the "Will Call" window located at our Welcome Center Ticket
```

```
            building. Please have valid identification when picking up your items. If you
            have any questions about your order, don't hesitate to call: 570- 629-1661
            ext. 1109 or email us at: estore@skicamelback.com. Thanks again and we look
            forward to seeing you soon! Camelback & Camelbeach </body>
            <html L=1>0</html>
            <pdf>1</pdf>
            <pdf_t>5</pdf_t></email>
            <ww_tix><pdf>1</ pdf><pdf_t>5</pdf_t></ww_tix>
<purchaser>9739000000</purchaser>
```

Code to deserialize an XML dataset

The following is sample code showing how to deserialize an XML dataset using VB.NET (.Net 1.1).

Note: This code does not work with .Net 2.0.

```
    Public Shared Function deSerializeDS(ByVal xml As String)
        Dim ds As New DataSet
        If Not IsNothing(xml) Then
            Dim XMLRead As New System.IO.StringReader(xml)
            Try
                ds.ReadXml(XMLRead)
                XMLRead.Close()
            Catch ex As Exception
                Return Nothing
            End Try
            XMLRead = Nothing
        End If
        Return ds
    End Function
```

The data you want is in `Table(1)`, with `Table(0)` being kind of a dummy table.

Single quotes that are part of a search query are now supported

Single quotes (e.g., O'Brien) are supported by `ww.dll`.

ww.dll now supports specials

`ww.dll` now supports specials through the use of `<srcecode>` and `<mktgcode>`, which can be passed in through the functions `getitem`, `getitemexpanded`, `getmods` and `getitemtree`. For example, the following `getitem` call:

```
<department>TEST2</department><category>TEST2<
/category><item>MWFTICKET</item><calcprice>1</calcprice><datetime>10-02-2006
</datetime><srcecode>Y06M08D30</srcecode>
```

returns the following `price_xml`:

```
[ext]11.25[/ext][tax]0.00[/tax][fee]0.00[/fee][txa]
0.00[/txa][txb]0.00[/txb][dwp]0.00[/dwp][in]15.00[/in][sp]ONLINE25[/sp]
```

The `ONLINE25` special is configured to apply a 25% discount between 10/01/2006 and 12/31/2006 to the item `MWFTICKET`. Furthermore, the special is authorized for the source code `Y06M08D30`. The item `MWFTICKET` is normally priced at $15.00 (`[in]15.00[/in]`), but with the special applied because of the source code the price gets correctly set to $11.25 (`[ext]11.25[/ext]`).

The `processsale` call must contain the `<special>` tag within the `<item>` tag in order to replicate the special applied via the source code `<srcecode>` or marketing code `<mktgcode>`. For example, here is a portion of a `processsale` call with `<special>` tags correctly embedded:

```
<item><dci>IMAX MOVIES OCEAN0730</dci><qty>2</qty>
<special><name>ONLINE</name><mktgcode>email</mktgcode></special>
<finalprice>4.00</finalprice><date>02-22-2007</date><message></message><mod><dci>IMAX
MODIFIERSSENIOR </dci><date>02-22-
2007</date><finalprice>0</finalprice><val_info></val_info>
<do_on_sale></do_on_sale></mod><val_info></val_info><do_on_sale>
</do_on_sale></item><item><dci>TEST2 TEST2
BIGWEDSURF</dci><qty>2</qty><special>ONLINE</special>
<finalprice>6.25</finalprice><date>02-23-
2007</date><message></message><val_info></val_info>
<do_on_sale></do_on_sale></item><item><dci>TEST TEST REGULAR
</dci><qty>1</qty><finalprice>19.95</finalprice><date>02-19-
2007</date><message></message><val_info></val_info><do_on_sale>
</do_on_sale></item><item><dci>MISC SHIPPING LOWSHIP
</dci><qty>1</qty><finalprice>5</finalprice><message></message>
<val_info></val_info><do_on_sale></do_on_sale></item>
```

Specials get submitted as <special><name>WEB$5OFF</name> <mktgcode>email</mktgcode></special>

Specials now get submitted in the sales string as follows: `<special><name>WEB$5OFF</name> <mktgcode>local</mktgcode></special>` or `<special><name>WEB25%OFF</name><srcecode>goodstuff</srcecode></special>`. Then when Sales Host processes the sale, the correct special name, source or marketing code, and discount amount get correctly written into `tr_save` or `transact`.

How ww.dll reacts to quantity tags

`ww.dll` reacts to quantity tags in the following manner. `<qty></qty>` tags outside of the `<item`*n*`> </item`*n*`>` tags apply to all items in the call unless there are `<qty></qty>` inside of any of the `<item`*n*`> </item`*n*`>` tags, in which case the `<qty></qty>` tags inside of the `<item`*n*`></item`*n*`>` tags override the `<qty></qty>` tags outside of the `<item`*n*`></item`*n*`>` tags.

For example:

```
<func>getitemexpanded</func><qty>10</qty><item1><department>TEST</department><category>TES
T</category><item>DPTEST1</item><hasmod>TEST MODIFIERS DPMOD1
```

```
</hasmod></item1><item2><qty>6</qty><department>IMAX</department><category>MOVIES</categor
y><item>STORM0730</item><hasmod>IMAX MODIFIERS
ADULT</hasmod></item2><item3><qty>12</qty><department>TEST</department><category>TEST
</category><item>CRTGSTWMOD</item><hasmod>TEST MODIFIERS TESTMOD01
</hasmod></item3><item4><department>IMAX</department><category>MOVIES</category><item>OCEA
N0730</item><hasmod>IMAX MODIFIERS CHILD
</hasmod></item4><calcprice>1</calcprice><datetime></datetime>
```

The above uses `qty 10` for `item1` and `item4`, `qty 6` for `item2`, and `qty 12` for `item3`.

Troubleshooting

`ww.dll` supports a registry setting that sets the verbosity for logging of `ww.dll` activity into the event log. This requires the following new setting to go into the `[HKEY_LOCAL_MACHINE\SOFTWARE\Siriusware\E-Commerce\Preferences]` section of the Windows registry: `EventLogVerbosity=n` where n can be `0`, `1`, `2`, `3`, `4`, or `5`. With this set to `1`, only errors are logged. With this set to `2`, warnings are also added into the log. With this set between `3` and `5`, additional logging of `ww.dll` startups and stops, calls made to `ww.dll`, and OCV and socket opening/closing information is logged.

Note: Setting this to `4` or `5` should be used only for troubleshooting purposes. A large event log is created and there is the possibility of performance degradation.

You can also use the `getlog` function for troubleshooting. See getlog.

The `TestCard` setting that is in the Windows registry that `ww.dll` uses (`[HKEY_LOCAL_MACHINE\SOFTWARE\Siriusware\E-Commerce\Preferences]`) functions so that if a valid “test” credit card number is entered (e.g., `5454545454545454`), `ww.dll` bypasses the attempt to get an approval from ProtoBase. `ww.dll` writes a valid sale string into the `ww_sales` table with an approval for the sale so that Sales Host is able to process the sale without error provided it is also using a matching `TestCard` setting (i.e., `TestCard=5454545454545454`) in the `[WebInterface]` section of the `Sales32c.INI` file. For more information on the `TestCard` setting, see the *Salesware .INI Settings Reference*. For a complete theory of operation for `ww.dll`, see the *Salesware User Guide for Advanced Users*.

API descriptions

This section lists and describes every API available in the `.dll`. There are two general categories for APIs: functions with no datababase access, and functions with database access.

Note that the API includes the ability to execute the SQL commands `SELECT`, `UPDATE` and `INSERT`. While it is possible to use these commands to perform common operations against the database, you must use the pre-defined functions wherever possible instead. This is because many operations need to be performed in a certain order to keep consistency with how Classic Sales performs these operations. In many cases, the same code is used for both Classic and Web Sales operations.

Functions are listed in alphabetical order within these categories. The format of the descriptions is as follows:

API name

The name of the call as it appears between the `<func></func>` tags.

Description

A general description of the API.

Input fields

The input fields, if any.

Return fields

The return fields. Sometimes a simply series of XML tags is returned, usually representing fields from a SiriusSQL table. Other times a more elaborate result is required: the XML schema used to represent the result is returned along with the data.

Example

Example invocation:

Example invocation of the function.

Example return string:

Example return string from the function. May be a simple indication of status (OK or ERR), a single or series of XML tags (usually representing fields from a SiriusSQL table), or an XML schema specification along with the data organized as described by the schema. In the later case, a tool such as XML Spy can be used to display the schema and the data in tabular format for easy reference. To conserve space, the return strings in this manual only show the data, not the schema or XML namespace definitions.

See also

Related functions in the API.

Functions with no database access

callez

Description

The `CallEZ` function can connect to functions in any EZ (SalesEZ, BookEZ, etc.). Currently, `tricklepass` and `tricklecard` are the only supported functions. For example:

```
<func>callez</func><ip>127.0.0.1</ip><port>4203</port><ezfunc>tricklepass</ezfunc><tclocat
ion>test</tclocation><tnpassno>288003010</tnpassno><tcprefix>
</tcprefix><tnnumvals>1</tnnumvals><ttscandatetime>2007-1-10 8:00:00</ttscandatetime>
OK :OK
```

The EZ must be running under Pool at the `IP` and `port` specified. `ww.dll` removes `IP`, `port`, and `timeout` and the remaining information is passed along in the call to the EZ. `tcoperator` and `tcsalespoint` are filled in by `ww.dll` if not provided. This function requires that a user be very familiar with the operation of the EZs and is not fully documented.

The `callez` function for `<ezfunc>tricklepass</ezfunc>` also accepts `<additno>` in lieu of a pass number – the first matching record from `access` or `gst_pass` is sent. For example:

```
<func>callez</func><ip>127.0.0.1</ip><port>4203</port><ezfunc>tricklepass</ezfunc><tclocat
ion>test</tclocation><additno>101</additno><tcprefix>A</tcprefix><tnnumvals>1</tnnumvals><
ttscandatetime>2008-1-10 8:00:00</ttscandatetime>
```

When calling `callez` and the function `tricklepass`, an `acc_prefix` and a `pass_prefix` can be specified (in lieu of `tcprefix`) so that the correct prefix gets sent along with the tickets / passes.

Input fields

XML tag	Description
`<IP>`	IP address of computer running Pool.
`<port>`	Port providing communications with the EZ.
`<timeout>`	Timeout in seconds. Defaults to `10` seconds if not specified.
`<ezfunc>`	EZ function.
Function-specific parameters	Contact Siriusware Technical Support.

Return fields

Information returned by EZ function.

Example

Example invocation:

```
<func>callez</func><ip>65.118.84.224</ip><port>4203</port><ezfunc>tricklepass</ezfunc>
<tclocation>test</tclocation><additno>135792468</additno><acc_prefix>A</acc_prefix>
<pass_prefix>P</pass_prefix><tnnumvals>1</tnnumvals><ttscandatetime>2008-2-22
14:00:00</ttscandatetime>
```

Example return string:

```
OK :OK
0-0-OK - PASS_NO: 376101000 - GUEST_NO: 212000000 - NAME: MICHAEL GUTTMAN - WARNINGS: 0 -
PASSTABLE: GST_PASS
```

See also

None.

getserverinfo

Description

`getserverinfo` provides more detailed information than `getversion`. Returns the `<path>` and `<filename>` of the `.dll` in addition to the internal `<version>` number.

Input fields

None.

Return fields

XML tag	Description
`<path>`	Filepath specifying location of `.dll`.
`<filename>`	Filename of `.dll`.
`<version>`	Version description.

Example

Example invocation:

```
<func>getserverinfo<func>
```

Example return string:

```
OK :<path>C:\Program Files\Siriusware\E-Commerce</path><filename>ww.dll</filename>
<version>4.0.53i</version>
```

See also

getversion

getserverversion

Description

`getserverversion` returns the version of `ww.dll`.

Input fields

None.

Return fields

XML tag	Description
`<version>`	Version description.

Example

Example invocation:

```
<func>getserverversion<func>
```

Example return string:

```
OK :<version>4.0.56 [4.0.55w]</version>
```

See also

getversion

getverbositylevel

Description

`getverbositylevel` returns the verbosity level that `ww.dll` is currently set to.

Input fields

None.

Return fields

XML tag	Description
`<level>`	Integer representing the verbosity level.

Example

Example invocation:

```
<func>getverbositylevel<func>
```

Example return string:

```
OK :<level>0</level>
```

See also

setverbositylevel

getversion

Description

`getversion` returns the internal `<version>` of the `.dll` (may not exactly match the file properties).

Input fields

None.

Return fields

XML tag	Description
`<version>`	Version description.

Example

Example invocation:

```
<func>getversion<func>
```

Example return string:

```
OK :<version>4.0.56 [4.0.55w]</version>
```

See also

getserverinfo

setverbositylevel

Description

`setverbositylevel` sets the verbosity level for `ww.dll`.

Input fields

XML tag	Description
`<level>`	Sets the verbosity level.
`<tnnewlevel>`	Sets the verbosity level temporarily. The verbosity remains at the level set in this manner until `ww.dll` is restarted via a reboot in **Component Services**, and then it uses the verbosity as set in the registry.

Return fields

Status only.

Example

Example invocation:

```
<func>setverbositylevel</func><level>1</level>
```

Example return string:

```
OK :
```

See also

getverbositylevel

sleep

Description

`sleep` sleeps the indicated number of seconds. This function is used only for testing. For example, `<func>sleep</func><secs>10</secs>` waits `10` seconds before returning.

Input fields

XML tag	Description
`<secs>`	Amount of time to sleep (in seconds).

Return fields

Status only.

Example

Example invocation:

```
<func>sleep</func><secs>10</secs>
```

Example return string:

```
OK :
```

See also

None.

writefile

Description

`writefile` writes `<filename>` to the specified `<filepath>` containing the `<filedata>`.

Input fields

XML tag	Description
<filepath>	The filepath.
<filename>	The filename.
<filedata>	The file data.

Return fields

Status only.

Example

Example invocation:

```
<func>writefile</func><filepath>c:\siriusware\</filepath><filename>text.txt
</filename><filedata>This is a test</filedata>
```

Example return string:

```
OK :
```

See also

None.

Functions with database access

acceptliability

Description

Liability acceptance can be required per item via SysManager. `acceptliability` is used to write a record into the `gst_actv` table to denote that a guest accepted a liability form.

Input fields

XML tag	Description
<guest_no>	Guest number.

<form_no>	Corresponds to the pri_key field in the l_forms table. Gets written into the ref field in gst_actv.
<expires>	Optional. If <expires> is empty or absent, the expiration date for the liability form is calculated according to how the liability form is configured, and the correct date gets entered into the tag_line field in gst_actv. If the <expires> parameter is populated with a date, then this gets entered into the tag_line field in gst_actv.
<details>	Optional. If the <details> parameter is empty or absent, then the text of the liability form gets entered into the details field in gst_actv. Otherwise whatever is passed populates that field. The activ_type for liability form acceptance in gst_actv is 400.

Return fields

Status only.

Example

Example invocation:

```
<func>acceptliability</func><guest_no>6000010</guest_no><form_no>2</form_no>
```

Example return string:

```
OK :Liability Accepted
```

See also

getliabilityinfo

checkpassword

Description

checkpassword works on the operator, guests, and b_instr tables. The checkpassword function increments the logattmpts field in the operator table the prefs.ologattmpt field is set or the guests table if the g.logattmpt field is set. If a user has been locked out, <force>1</force> may be added to the call to override the lockout.

The password functions require a minimum password length of 8 with at least 1 character and 1 number for each password. They enforce not reusing a certain number of passwords for operators as set with prefs.oreusepswd. They don't allow passwords to be changed until a certain period of time has elapsed (again, for operators as set with prefs.ominpwage). They log the number of failed attempts (if

prefs.ologattmpt and prefs.glogattmpt are set). They can lock out an operator for a period of time (minutes in prefs.olocktime) if prefs.ologattmpt is exceeded. And they can lock out a guest if prefs.glogattmpt is exceeded.

Input fields

XML tag	Description
<table>	Table (the operator, guests, or b_instr table).
<id>	The op_code for the operator table, the guest_no for the guests table, or the instr_id for the b_instr table.
<password>	Password. *Note*: What is entered into the <password></password> tags for both checkpassword and setpassword is case sensitive. However, passwords entered in Sales or SysManager and hashed/stored in the SiriusSQL database for operators and instructors are always upper case. So when using these functions for operators and instructors, always enter characters in the <password></password> tags in upper case. Characters entered into the <password></password> tags for guests can be mixed case as these passwords are hashed/stored in the SiriusSQL database in mixed case.
<force>	Set to 1 to override lockouts.

Return fields

Status only.

Example

Example invocation:

```
<func>checkpassword</func><table>operator</table><id>LARRYL</id><password>BILY111</password>
```

Example return string:

```
OK
```

See also

setpassword

clearitemres

Description

`clearitemres` eliminates records from the `item_res` table. The `item_res` table tracks real-time item reservations (primarily when a user puts an item into his cart). This table is used to provide accurate Max4Sale remaining quantity calculations.

Input fields

XML tag	Description
`<wwsale_id>`	Web sale ID from a web sale. If a `wwsale_id` that is passed already exists in the `ww_sales` table, `ww.dll` returns "ERR:wwsale_id is not unique." Either `<wwsale_id>` or `<itmres_no>` (or both) must be specified.
`<itmres_no>`	Item reservation number. Either `<wwsale_id>` or `<itmres_no>` (or both) must be specified.

Return fields

Status only.

Example

Example invocation:

```
<func>clearitemres</func><wwsale_id>1000000</wwsale_id><itmres_no>
4000000</itmres_no>
```

Example return string:

```
OK :Records deleted
```

See also

`newitemres`

createaccess

Description

`createaccess` is provided for convenience. It simply calls `createpass` with `<access>1</access>` specified. For more information on parameters, see `createpass`.

Input fields

XML tag	Description
<guest_no>	Guest number. (If used, it is ignored.)
<department>	<item> department.
<category>	<item> category.
<item>	Item.
<start_date>	Optional. Start date.
<expires>	Optional. Expiration date.
<other field>	Optional. Any field from the gst_pass or access table.

Return fields

XML tag	Description
<pass_no>	New pass number.

Example

Example invocation:

```
<func>createaccess</func><start_date>06-07-
2007</start_date><department>PASSES</department>
<category>EMPLOYEE</category><item>EMPLOYEE</item>
<amt_paid>399.95</amt_paid><points1>2</points1>
```

Example return string:

```
OK :<pass_no>11000001</pass_no>
```

See also

createpass

createpass

Description

createpass is used to create new pass *or* new access information. It takes the following required parameters: <department>, <category>, <item>. If <access>1</access> is specified, the call creates an access record in the access table. If <access>1</access> is not specified or is set to 0, the call creates a pass record in the gst_pass table. The call takes the following optional parameters: <guest_no>, <start_date>, <expires> and any other fields in the gst_pass and/or access tables

Input fields

XML tag	Description
<guest_no>	Guest number. Optional if the <access> tag is set to 1 (if used, it is ignored). Required if the <access> tag is set to 0 or the <access> tag is not specified.
<department>	<item> department.
<category>	<item> category.
<item>	Item.
<start_date>	Optional. Start date.
<expires>	Optional. Expiration date.
<access>	Optional. Creates an access record. Boolean.
<other field>	Optional. Any field from the gst_pass or access table.

Return fields

XML tag	Description
<pass_no>	New pass number.

Example

Example invocation:

```
<func>createpass</func><guest_no>26000001</guest_no><start_date>06-07-
2007</start_date><department>PASSES</department>
<category>EMPLOYEE</category><item>EMPLOYEE</item>
<amt_paid>399.95</amt_paid><points1>2</points1>
```

To create a new pass record:

```
<func>createpass</func><guest_no>212000000</guest_no<start_date>06-07-
2007</start_date><department>PASSES</department<category>ADULTPASS</category><item>2007ADS
EAS</item><amt_paid>399.95</amt_paid><points1>2</points1>
```

To create a new access record:

```
<func>createpass</func><start_date>06-08-2007</start_date><department>TICKETS</department>
<category>ADULTS</category><item>01DYADTKT</item><total_uses>3</total_uses><amt_paid>19.95
</amt_paid><access>1</access>
```

Example return string:

```
OK :<pass_no>1000001</pass_no>
```

See also

createaccess

decryptdata

Description

Decrypts the data passed in. The key is expected to be a hexadecimal hash.

Input fields

XML tag	Description
<data>	Data to be decrypted.
<key>	Key to use for decryption.

Return fields

The decrypted data.

Example

Example invocation:

```
<FUNC>DECRYPTDATA</FUNC><DATA>4D1A02054518D598E47860DD9E8EC2EB</DATA>
<KEY>289D1BFB42672B7288FE2AAB8170094ECB6A334B48A4CE069709B9F26896BC41</
KEY>
```

Example return string:

```
OK :Test
```

See also

encryptdata
encryptdata2
gethash

encryptdata

Description

Constructs the key, hashes it, and then encrypts the data.

Input fields

XML tag	Description
`<data>`	Data to be encrypted.
`<id>`	Contact Siriusware Technical Support for guidelines about using this field.
`<client>`	Client number.

Return fields

Hexadecimal representation of the encrypted data.

Example

Example invocation:

```
<func>encryptdata</func><data>Test</data><id>1</id><client>5051</client>
```

Example return string:

```
OK :4d1a02054518d598e47860dd9e8ec2eb
```

See also

decryptdata
encryptdata2
gethash

encryptdata2

Description

Constructs the key, hashes it, and then encrypts the data. Differs from `encryptdata` because this function uses the live database connection to get the client number.

Input fields

XML tag	Description
<data>	Data to be encrypted.
<id>	Contact Siriusware Technical Support for guidelines about using this field.

Return fields

Hexadecimal representation of the encrypted data.

Example

Example invocation:

```
<func>encryptdata2</func><data>Test</data><id>1</id>
```

Example return string:

```
OK :4d1a02054518d598e47860dd9e8ec2eb
```

See also

decryptdata
encryptdata
gethash

execsp

Description

Makes a stored procedure call, essentially of the form `EXECUTE` followed by the `spname` and then the `params`. If a recordset is returned, it is converted to XML and returned to the caller.

Input fields

XML tag	Description
<spname>	Stored procedure name.
<params>	Parameters to send.

Return fields

OK with recordset or ERR

Example

Example invocation:

```
<func>execsp</func>
<spname>SIRIUSSP_GETQTYREMAINING_BY_DCI_BASE</spname>
<params>'10-19-2008 08:00:00', 'test', 'test', 'main', 0</params>
```

Example return string:

```
OK :...recordset
```

See also

getuniquekey
insert
select
update

getaccessinfo

Description

`getaccessinfo` is provided for convenience. It simply calls `getpassinfo` with `<access>1</access>` specified and returns information regarding new and modified access records.

Input fields

XML tag	Description
`<new>`	Return new records since the indicated `datetime`. `<new>1</new>` (for new pass/access info) or `<mod>1</mod>` (for modified pass/access info) must be specified in the function call. Boolean.
`<mod>`	Return modified records since the indicated `datetime`. `<new>1</new>` (for new pass/access info) or `<mod>1</mod>` (for modified pass/access info) must be specified in the function call. Boolean.
`<datetime>`	Sets the `datetime`.

Return fields

Field	Description
pass_no	Pass number.
cluster	Cluster.
parent_no	Parent number.
swipe_no	Swipe number.
addit_no	Additional number.
last_use	Last use.
e_message	E-message.
start_date	Start date.
expires	Expires.
good_for	Good for this number of accesses.
validcount	Number of times validated.
dis_count	Like validcount, but can be reset.
total_uses	Total uses.
usesw_left	Uses left in current week.
week_refr	Used to determine when a week has passed.
usest_left	Uses left in current day.
day_refr	Used to determine when a day has passed.
points1	User-defined meaning.
points2	User-defined meaning.
money1	User-defined meaning.
money2	User-defined meaning.
blackout_s	Blackout start.
blackout_e	Blackout end.
warnings	User-defined meaning.
voided_for	Reason for void.
voided_by	Who voided.
shift_ends	Next time pass is validated.
department	Department.
category	Category.
item	Item.
amt_paid	Amount paid.

account	Account.
operator	Operator.
salespoint	Salespoint.
date_time	Date/time.
val_parent	Validate parent record.
trans_no	Transaction number.
cc_tracks	Hold some or all of the track1, 2 and 3 magnetic swipe info from card.
bl_reason	Blackout reason.

Example

Example invocation:

Returns all new access records added since 6/06/2007:

```
<func>getaccessinfo</func><new>1</new><mod>0</mod>
<datetime>6/06/2007</datetime>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row pass_no='1018001' cluster='195' parent_no='1018001'
             swipe_no='                    ' addit_no='0' e_message=''
             start_date='2007-09-30T12:00:00' expires='2007-09-30T12:05:00'
             good_for='1' validcount='0' dis_count='0' total_uses='1'
             usesw_left='0' usest_left='0' points1='0' points2='0'
             money1='0' money2='0' warnings='0' voided_for='' voided_by=''
             department='TICKETS   ' category='ADULT     ' item='1DAY      '
             amt_paid='300' account='*RESRVATN*' operator='ADMIN '
             salespoint='GOLF  ' date_time='2007-09-30T14:57:31' val_parent='False'
             trans_no='58020001' cc_tracks='' bl_reason=''/>
</rs:data>
```

See also

getpassinfo

getdciavail

Description

`getdciavail` finds available booking times for Private Instruction. For example:

```
<func>getdciavail</func><dcilist>privates,skiing,2hrpriv;privates,
skiing,4hrpriv</dcilist><start_time>03/12/2007 9:00 AM</start_time>
<end_time>03/12/2007 2:00 PM</end_time>
<instructor>swilson</instructor>
```

What gets returned is a list of DCIs, `qty_rem` and `book_times`. The list of DCIs is the same as what gets passed in for the call, the `qty_rem` returned is the number of slots available during the timeframe set by the start and end times, and the `book_times` are the start times that are available. The `qty_rem` numbers returned take into account lesson type Max4Sale limits and what's already been booked into these limits. DCI Max4Sale limits enter into this in that if a DCI Max4Sale limit is less than the lesson type Max4Sale and the DCI Max4Sale limit has been used up, then the `qty_rem` returned is empty. So the number returned to `qty_rem` reflects the lesson type Max4Sale remaining until the DCI Max4Sale limit has been used up.

Input fields

XML tag	Description
`<dcilist>`	List of DCIs using commas to separate the department from category and category from item, and using a semicolon to separate complete DCIs.
`<start_time>`	Defines the period to examine.
`<end_time>`	Defines the period to examine.
`<instructor>`	Optional. Instructor.

Return fields

Field	Description
`department`	Department.
`category`	Category.
`item`	Item.
`descrip`	Description.
`ckmax4sale`	Check Max4Sale.
`ckpts4sale`	Check Points4Sale.
`pts4saledp`	Pts4Sale Dynamic Pricing rule.
`qty_rem`	Quantity remaining.
`book_times`	Booking times.
`span`	Time span.
`book_prefs`	Booking preferences.
`startimes`	Start times.
`startime1`	Start time 1.
`startime2`	Start time 2.

startime3	Start time 3.
startime4	Start time 4.
startime5	Start time 5.
startime6	Start time 6.
startime7	Start time 7.
startime8	Start time 8.
startime9	Start time 9.
startime10	Start time 10.

Example

Example invocation:

```
<func>getdciavail</func><dcilist>LSSNS-PRIV,ALPINE,2HOUR</dcilist>
<start_time>11/06/2007 10:00am</start_time><end_time>11/06/2007 12:00pm</end_time>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row department='LSSNS-PRIV' category='ALPINE    '
             item='2HOUR     ' descrip='2 Hour Private Alpline   '
             ckmax4sale='False' ckpts4sale='False' pts4saledp='0' qty_rem='-, -, -, -, -,
-, -, -, -'
             book_times='11/06/2007 10:00:00, 11/06/2007 10:15:00, 11/06/2007 10:30:00,
11/06/2007 10:45:00, 11/06/2007 11:00:00, 11/06/2007 11:15:00, 11/06/2007 11:30:00,
11/06/2007 11:45:00, 11/06/2007 12:00:00'
             span='120' book_prefs='&#x3c;Pref_Lvl1&#x3e;1&#x3c;/Pref_Lvl1&#x3e;
&#x3c;Pref_Lvl2&#x3e;1&#x3c;/Pref_Lvl2&#x3e;
&#x3c;Pref_Lvl3&#x3e;1&#x3c;/Pref_Lvl3&#x3e;
&#x3c;Pref_Lvl4&#x3e;3&#x3c;/Pref_Lvl4&#x3e;
&#x3c;Pref_Lvl5&#x3e;1&#x3c;/Pref_Lvl5&#x3e;
&#x3c;Pref_Lvl6&#x3e;1&#x3c;/Pref_Lvl6&#x3e;
&#x3c;Pref_Lvl7&#x3e;1&#x3c;/Pref_Lvl7&#x3e;
&#x3c;Pref_Lvl8&#x3e;1&#x3c;/Pref_Lvl8&#x3e;
&#x3c;Pref_Lvl9&#x3e;1&#x3c;/Pref_Lvl9&#x3e;
&#x3c;sex&#x3e;1&#x3c;/sex&#x3e;
&#x3c;Criteria1&#x3e;0&#x3c;/Criteria1&#x3e;
&#x3c;Pref_Lvl10&#x3e;1&#x3c;/Pref_Lvl10&#x3e;
&#x3c;Level&#x3e;0&#x3c;/Level&#x3e;
&#x3c;Criteria6&#x3e;0&#x3c;/Criteria6&#x3e;
&#x3c;Pref_Lvl11&#x3e;1&#x3c;/Pref_Lvl11&#x3e;
&#x3c;Criteria7&#x3e;0&#x3c;/Criteria7&#x3e;
&#x3c;Pref_Lvl12&#x3e;1&#x3c;/Pref_Lvl12&#x3e;
&#x3c;p_level&#x3e;.T.&#x3c;/p_level&#x3e;
&#x3c;DisciplineOption&#x3e;1&#x3c;/DisciplineOption&#x3e;
' startimes='False' startime1='        ' startime2='        '
             startime3='        ' startime4='        ' startime5='        '
             startime6='        ' startime7='        ' startime8='        '
             startime9='        ' startime10='        '/>
</rs:data>
```

See also

newbooking

getdwinfo

Description

`getdwinfo` returns information about an In-House Cards card. The information comes from the `siriussp_GetDebitwareApproval` stored procedure combined with fields from the `gst_pass` and `guests` tables.

Input fields

XML tag	Description
`<pass_no>`	The pass number of the pass associated with the In-House Card for which information is being returned.
`<swipe_no>`	The swipe number for the In-House Cards card for which information is to be returned.

Return fields

Field	Description
`invoice_no`	Invoice number.
`inv_bal`	Invoice balance.
`sp_bal`	Total amount spent during the history of this In-House Cards card. (Calculated value from stored procedure.)
`sp_bal_dy`	Total amount spent for the current day for this In-House Cards card. (Calculated value from stored procedure.)
`crlimit`	Credit limit for this In-House Cards card.
`splimit`	Total spending limit for this In-House Cards card.
`splimit_dy`	Daily spending limit for this In-House Cards card.
`pass_no`	Number of pass associated with In-House Cards card.
`swipe_no`	Swipe number.
`start_date`	Start date.
`expires`	Expiration date.
`dw_active`	Indication of whether In-House Cards card is active.
`department`	In-House Cards item department.
`category`	In-House Cards item category.
`item`	In-House Cards item.

voided_by	If voided, name of operator who performed void.
voided_for	If voided, reason for void.
guest_no	Guest number.
first_name	Guest first name.
last_name	Guest last name.
dw_ignrexp	Boolean indicating whether to ignore expiration dates on In-House Cards.

Example

Example invocation:

```
<func>getdwinfo</func><swipe_no>6035241000000139</swipe_no>
```

Example return string:

```
OK :...
<rs:data>
      <z:row invoice_no='5003001' inv_bal='-18.00' sp_bal='2.00'
             sp_bal_dy='.00' crlimit='.00' splimit='.00' splimit_dy='.00'
             approval='162439' pass_no='25003001' swipe_no='6035241000000139      '
             start_date='2006-11-01T00:00:00' expires='2008-01-20T23:59:59'
             dw_active='True' department='INHOUSECRD' category='GIFTCERTS '
             item='GIFTCERT  ' voided_by='' voided_for='' guest_no='4000001'
             first_name='NANI' last_name='GORING'/>
</rs:data>
```

See also

None.

getguest

Description

`getguest` retrieves a guest record, joined to the record(s) for the address(es).

Input fields

XML tag	Description
<guest_no>	Guest number.
<pass_no>	Optional. Pass number.
<gst_rent>	Optional. Automatically merge in the `gst_rent` information (use the alias `r` if you

	want to re-specify the fields to be retrieved other than the defaults). Boolean.
`<fields>`	Optional. See Note on the <fields> tag.
`<getphoto>`	Optional. Boolean indicating whether `getguest` should return a photo.

Return fields

Recordset of all the fields in the `guest`, `address` and `addlink` tables with the exception of `mug_shot`. Mug shots, because they are graphics, tend to be large files. However, `has_photo` is returned with a value of `1` if the `mug_shot` field is populated (and `0` if not), and `<getphoto>0<getphoto>` can be used to retrieve a mug shot if needed.

Field	Description
`guest_no`	Guest number.
`last_mod`	Last modified.
`salespoint`	Salespoint.
`operator`	Operator.
`date_time`	Date/time.
`parent_no`	Parent number.
`addit_no`	Additional number.
`addit_no2`	Additional number 2.
`trans_no`	Transaction number.
`guestgroup`	Guest group.
`salute`	Salutation.
`first_name`	First name.
`last_name`	Last name.
`e_mail`	E-mail address.
`birth_date`	Birth date.
`gender`	Gender.
`cc_swipe`	Credit card swipe.
`cc_number`	Credit card number.
`mug_date`	Mug shot date.
`notes`	Notes.
`e_message`	E-message.
`check_bx1`	Check box 1.
`check_bx2`	Check box 2.

check_bx3	Check box 3.
check_bx4	Check box 4.
check_bx5	Check box 5.
check_bx6	Check box 6.
check_bx7	Check box 7.
check_bx8	Check box 8.
check_bx9	Check box 9.
check_bx10	Check box 10.
check_bx11	Check box 11.
check_bx12	Check box 12.
check_bx13	Check box 13.
check_bx14	Check box 14.
check_bx15	Check box 15.
number_1	Number 1.
number_2	Number 2.
number_3	Number 3.
number_4	Number 4.
number_5	Number 5.
text_1	Text 1.
text_2	Text 2.
text_3	Text 3.
text_4	Text 4.
text_5	Text 5.
text_6	Text 6.
text_7	Text 7.
text_8	Text 8.
memo_1	Memo 1.
memo_2	Memo 2.
memo_3	Memo 3.
date_1	Date 1.
datetime_1	Date/time 1.
gfwdstatus	Forward status.
guest_id	Guest ID.
relation	Relation.

role_no	Role number.
acct_name	Account name.
vipcode	VIP code.
height	Height.
weight	Weight.
height_m	Height in centimeters.
weight_m	Weight in centimeters.
no_mail	No mail.
no_email	No e-mail.
no_phone	No phone.
addlink_id	Add link ID.
address_id	Address ID.
addr_type	Address type.
address	Address.
address2	Address 2.
city	City.
state	State.
zip	Zip.
country	Country.
cntry_cod	Country code.
area_code	Area code.
phone	Phone.
cntry_cod2	Country code 2.
area_cod2	Area code 2.
phone2	Phone 2.
fax_ccode	Fax country code.
fax_acode	Fax area code.
fax_phone	Fax phone.
company	Company.
has_photo	Has photo.

Example

Example invocation:

```
<func>getguest</func><guest_no>26000001</guest_no>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row guest_no='26000001' last_mod='805' salespoint='OFFICE'
             operator='ADMIN ' date_time='2007-10-18T09:54:36' parent_no='26000001'
             addit_no='0' addit_no2='0' trans_no='0' guestgroup=''
             salute='' first_name='BRAD           ' last_name='HARRISON           '
             e_mail='bharrison@siriusware.com
'
             gender=' ' cc_swipe='' cc_number='' notes='' e_message=''
             check_bx1='False' check_bx2='False' check_bx3='False'
             check_bx4='False' check_bx5='False' check_bx6='False'
             check_bx7='False' check_bx8='False' check_bx9='False'
             check_bx10='False' check_bx11='False' check_bx12='False'
             check_bx13='False' check_bx14='False' check_bx15='False'
             number_1='0' number_2='0' number_3='0' number_4='.00'
             number_5='.00' text_1='' text_2='' text_3='' text_4=''
             text_5='' text_6='' text_7='' text_8='' memo_1='' memo_2=''
             memo_3='' gfwdstatus='0' guest_id='0' relation='0' role_no='0'
             acct_name='' vipcode='0' height='0' weight='0' height_m='0'
             weight_m='0' no_mail='False' no_email='False' no_phone='False'
             has_photo='0'/>
</rs:data>
</xml>
```

See also

lookupguests

gethash

Description

Constructs a one-way hash of the data that is passed in.

Note: In version 4.0.56, the `encryptwebpass` function was removed from `ww.dll`. Passwords must now be hashed. The `gethash` function exists for this purpose. Credit card number, CVV2 number, and credit card expiration date are all masked in the `ww.dll` log when the processsale call return code is written. Any logging done by `ww.dll` as well as any calls to encryption functions are clean of the CVV2 number and credit card swipes. Credit card numbers only show the last four digits in the logs and the CVV2 number and the expiration date display as asterisks. For example:

```
<settlement><card_fname>TOM</card_fname><card_lname>THUMB</card_lname><car
d_addr>1337 E GUSDORF
ROAD</card_addr><card_addr2></card_addr2><card_city>TAOS</card_city><card_state
>NM</card_state><card_zip>87571</card_zip><card_cntry>USA</card_cntry><card_no>
************4242</card_no><exp_date>****</exp_date><card_zip>87571</card_zi
p><card_cvv2>***</card_cvv2><termid>WEB</termid><devicetype>15</devicetype><a
mount>0007.44</amount><disctotal>0000.00</disctotal><tax>0000.49</tax></settle
ment>
```

Card numbers, CVV2, and expiration dates are masked in event log entries from the E-Commerce pages as well as in any strings displayed in the `error.aspx` page before being displayed.

Input fields

XML tag	Description
`<data>`	Data to be hashed.

Return fields

Hexadecimal representation of the hashed data.

Example

Example invocation:

```
<func>gethash</func><data>15051secretkey</data>
```

Example return string:

```
OK :289d1bfb42672b7288fe2aab8170094ecb6a334b48a4ce069709b9f26896bc41
```

See also

decryptdata
encryptdata
encryptdata2

getitem

Note: Unless you have a good reason for not doing so, use `getitemexpanded` instead of `getitem`.

Description

`getitem` gets an item. `<calcprice>` uses the current date or the date passed in the `<datetime>` tag to calculate the price. The prices are included in the return string in the `price_info` field, including final price, tax and fee information. These should all be included in each `<item>` section when passed to the `processsale` function. If `<max4sale>` is sent as `true` (`<max4sale>1</max4sale>`), then information is sent back in the `qty_rem` field with the number left to sell at that `<datetime>`. Otherwise, a `-` will be returned in that field.

With regard to the <pass_no> field, if the corresponding pass template contains a special macro (e.g., SELECTLAST()ITEMSPECIAL ("ONLINE ")), then a discounted price is returned in price_xml in the [ext] field. The specials can be any of the following in the macro for the pass template: selectivespecial, globalspecial, itemspecial or special. If the pass is voided or expired, then the discounted price is not returned. For example, the following shows a sample call containing a pass number and what is returned in price_xml:

```
<func>getitem</func><department>TEST</department><category>TEST</
category><item>MAIN</item><pass_no>321003010</pass_no><calcprice>1</calcprice>

price_xml='[ext]98.00[/ext][tax]0.00[/tax][fee]0.00[/fee][txa]0.00[/txa]
[txb]0.00[/txb][dwp]0.00[/dwp][in]100.00[/in]'
```

Input fields

XML tag	Description
<qty>	Optional. Quantity. Price returned is for specified quantity instead of per unit. If no <qty></qty> tags are passed, then ww.dll assumes the quantity is 1 and returns the price for a quantity of 1. See How ww.dll reacts to quantity tags for more information.
<department>	<item> department.
<category>	<item> category.
<item>	Item.
<pass_no>	Optional. Pass number. If the corresponding pass template contains a special macro (e.g., SELECTLAST()ITEMSPECIAL ("ONLINE ")), then a discounted price is returned in price_xml in the [ext] field. The specials can be any of the following in the macro for the pass template: selectivespecial, globalspecial, itemspecial or special. If the pass is voided or expired, then the discounted price is not returned. See main description of this call for an example.
<calcprice>	Optional. Boolean (1 = calculate the price, 0 = don't calculate the price)
<max4sale>	Optional. Boolean (1 = return Max4Sale information, 0 = don't return Max4Sale information)
<calcpoints>	Optional. Boolean. If 1, a point cost of each item (at qty=1) is returned as <pts_cost>. Note that the ckpts4sale (is Points4Sale enforced) and pts4saledp (Dynamic Pricing rule to use to calculate Points4Sale points to deduct) fields from the items table must be obtained in the field list from the item record (e.g., <fields>it.department, it.category, it.item, i.ckmax4sale, ckpts4sale, pts4saledp</fields>). Also note that all item reservation functionality works the same for Points4Sale as it does for Max4Sale except that the total point cost is stored in the item_res.pts4qty field.
<avail>	Optional. Returns information on whether a particular item is available based on Max4Sale or Points4Sale restrictions for a certain date or date range. If the tag is

	absent or is specified as <avail>0</avail>, nothing is calculated. See Note on the <avail> tag.
<account>	Optional. An account nickname. When used, item price information is returned based on any dynamic pricing rule associated with the account. If a user is logged into a group (which is really an account), that group name is passed to ww.dll when getting product prices, and any dynamic pricing rule assigned to that account is applied to the prices. *Note*: In order for dynamic pricing rules assigned to accounts to work correctly in E-Commerce, you must add the following settings to the [Preferences] section of the Sales32c.INI file for the web Sales Host and the salespoint where you recall these sales: [Preferences] ApplyAccountRulesOnRecall=FALSE RecalculatePriceOnRecall=FALSE Without these settings in place, you see incorrect pricing on recall of these sales.
<fields>	Optional. Fields to be returned. See Note on the <fields> tag.
<mktgcode>	Optional. Marketing code. See ww.dll now supports specials.
<srcecode>	Optional. Source code. See ww.dll now supports specials.

Return fields

XML recordset of the matching record from the items table.

Field	Description
department	Department.
category	Category.
item	Item.
descrip	Description.
price_type	Price type.
dp_set_id	The set of Dynamic Pricing rules to apply first.
price_cols	Price columns – 1, 2, or 3.
wknd_start	Weekend start.
wknd_end	Weekend end.
rate_sched	Rate schedule.
daily_pric	Daily price.
monday_0	Default price.
monday_1	Price during special rate period 1.

monday_2	Price during special rate period 2.
monday_3	Price during special rate period 3.
monday_4	Price during special rate period 4.
monday_5	Price during special rate period 5.
tuesday_0	Default price.
tuesday_1	Price during special rate period 1.
tuesday_2	Price during special rate period 2.
tuesday_3	Price during special rate period 3.
tuesday_4	Price during special rate period 4.
tuesday_5	Price during special rate period 5.
weds_0	Default price.
weds_1	Price during special rate period 1.
weds_2	Price during special rate period 2.
weds_3	Price during special rate period 3.
weds_4	Price during special rate period 4.
weds_5	Price during special rate period 5.
thursday_0	Default price.
thursday_1	Price during special rate period 1.
thursday_2	Price during special rate period 2.
thursday_3	Price during special rate period 3.
thursday_4	Price during special rate period 4.
thursday_5	Price during special rate period 5.
friday_0	Default price.
friday_1	Price during special rate period 1.
friday_2	Price during special rate period 2.
friday_3	Price during special rate period 3.
friday_4	Price during special rate period 4.
friday_5	Price during special rate period 5.
saturday_0	Default price.
saturday_1	Price during special rate period 1.
saturday_2	Price during special rate period 2.
saturday_3	Price during special rate period 3.
saturday_4	Price during special rate period 4.
saturday_5	Price during special rate period 5.

sunday_0	Default price.
sunday_1	Price during special rate period 1.
sunday_2	Price during special rate period 2.
sunday_3	Price during special rate period 3.
sunday_4	Price during special rate period 4.
sunday_5	Price during special rate period 5.
tax_rate	Tax rate.
tax_rate_b	Tax rate is defined in defaults table.
cust_tax	Custom tax rate.
cust_tax_b	Custom tax rate is defined in defaults table.
fee_rate	Fee rate.
add_tax	Add tax.
add_tax_b	Add tax b.
time_span	Time span.
span_type	Span type.
admissions	Admissions.
split_type	Split type.
item_type	Item type.
validate	Validate.
validate2	Additional validation.
ckmax4sale	Check Max4Sale.
ckpts4sale	Check Points4Sale.
pts4saledp	DP Rule to use to calculate Points4Sale points to deduct.
novalonret	When a “forced validation” item is returned, should the salepoint make sure the pass is valid prior to returning it?
set_price	Set price.
inventory	Inventory.
invent_id	Inventory ID.
barcode	Barcode.
upc	UPC.
keycode	Keycode.
keydescrip	Key description.
mod_reqd	Modifier required?
multiline	Multiline.

help_info	Help information.
price_info	Price information. Includes final price, tax and fee information.
qty_rem	If <max4sale> is sent as true (<max4sale>1</max4sale>), then information is sent back with the number left to sell at that <datetime>. Otherwise, a - is returned in that field.
pts_cost	If <calcpoints> is sent as true (<calcpoints >1</calcpoints>), then information is sent back with the points cost of each item (at qty=1).
avail_info	Values returned as a result of using the <avail> tag. See Note on the <avail> tag.
price_xml	Price information. If the <pass_no> tag is sent in and the corresponding pass template contains a specials macro (e.g., SELECTLAST()ITEMSPECIAL ("ONLINE ")), then a discounted price is returned in price_xml in the [ext] field. The specials can be any of the following in the macro for the pass template: selectivespecial, globalspecial, itemspecial or special. If the pass is voided or expired, then the discounted price is not returned. See main description of this call for an example.

Example

Example invocation:

```
<func>getitem</func><department>FBBAR</department><category>BEER</category><item>BTL-
COORS</item><calcprice>1</calcprice>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row department='FBBAR     ' category='BEER      '
             item='BTL-COORS ' descrip='Coors - Bottle          '
             price_type='0' dp_set_id='0' price_cols='1' wknd_start='0'
             wknd_end='0' rate_sched='1' daily_pric='1' monday_0='2.5'
             monday_1='0' monday_2='0' monday_3='0' monday_4='0' monday_5='0'
             tuesday_0='0' tuesday_1='0' tuesday_2='0' tuesday_3='0'
             tuesday_4='0' tuesday_5='0' weds_0='0' weds_1='0' weds_2='0'
             weds_3='0' weds_4='0' weds_5='0' thursday_0='0' thursday_1='0'
             thursday_2='0' thursday_3='0' thursday_4='0' thursday_5='0'
             friday_0='0' friday_1='0' friday_2='0' friday_3='0' friday_4='0'
             friday_5='0' saturday_0='0' saturday_1='0' saturday_2='0'
             saturday_3='0' saturday_4='0' saturday_5='0' sunday_0='0'
             sunday_1='0' sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0'
             tax_rate='2' tax_rate_b='1' cust_tax='.0000' cust_tax_b='.0000'
             fee_rate='1' add_tax='False' add_tax_b='False' time_span='0'
             span_type='1' admissions='.00' split_type='1' item_type='1'
             validate='1' validate2='False' ckmax4sale='False' ckpts4sale='False'
             pts4saledp='0' novalonret='False' set_price='False' inventory='False'
```

```
            invent_id='0' barcode='                    ' upc='                    '
            keycode='0' keydescrip='          ' mod_reqd='0' multiline='False'
            help_info='' price_info='2.5000,0.1636,0.0000,0.1636,0.0000'

price_xml='[ext]2.5000[/ext][tax]0.1636[/tax][fee]0.0000[/fee][txa]0.1636[/txa][txb]0.0000
[/txb][dwp]0.0000[/dwp][in]2.5000[/in][sp][/sp]'/>
</rs:data>
```

See also

[getitemexpanded](#)
[getitemtree](#)
[getmods](#)
[processsale](#)

getitemexpanded

Note: Siriusware recommends that you use this call instead of `getitem` when possible.

Description

`getitemexpanded` adds modifiers to `getitem`. It takes the `<hasmod>` field and expands all DCIs involved to get the proper pricing for the main (parent) item and all of its modifiers. These are returned in a recordset. The recordset always has the parent item first. (This is accomplished by the presence of a parent field which is `1` for the parent item and `0` for all the modifiers).

For example:

```
<func>getitemexpanded</func><department>AAA-MATT  </department>
<category>AAA</category><item>WIDGET</item><calcprice>1</calcprice>
<HASMOD>AAA-MATT  AAA       BLING     AAA-MATT  AAA      JANKY</HASMOD>
```

returns a recordset with three records and prices for the main item and the modifiers.

This is different from `getitem`, which returns the price of the item passed, but doesn't include the modifiers in the record set.

If you use `getitemexpanded` (using the above example), you get three records (the main one and two modifiers). If you use `getitem`, you get one record (just the main item, but the price "knows" about the modifiers for the Dynamic Pricing rules).

`<calcprice>` uses the current date or the date passed in the `<datetime>` tag to calculate the price. The prices are included in the return string in the `price_info` field, including final price, tax and fee information. These should all be included in each `<item>` section when passed to the `processsale` function. If `<max4sale>` is sent as `true` (`<max4sale>1</max4sale>`), then information is sent back in the `qty_rem` field with the number left to sell at that `<datetime>`. Otherwise, a `-` will be returned in that field.

An enhanced `getitemexpanded` call now includes the ability to use the `<item1>` and `<item2>` tags to pass in multiple DCIs. For example:

```
<func>getitemexpanded</func><item1><department>TEST</department
><category>TEST</category><item>DPTEST1</item><hasmod>TEST MODIFIERS DPMOD1
</hasmod></item1><item2><department>IMAX</department><category
>MOVIES</category><item>STORM0730</item><hasmod>IMAX MODIFIERS ADULT
</hasmod></item2><calcprice>1</calcprice><datetime></datetime><ite
m3><department>TEST</department><category>TEST</category><item>
CRTGSTWMOD</item><hasmod>TEST MODIFIERS TESTMOD01
</hasmod></item3><item4><department>IMAX</department><category
>MOVIES</category><item>OCEAN0730</item><hasmod>IMAX MODIFIERS CHILD
</hasmod></item4><calcprice>1</calcprice><datetime></datetime>
```

The enhanced function returns all items with modifiers in the order received by the function (parent followed by its modifier[s] and then the next parent followed by its modifier[s], etc.). Also, in addition to the `ord` field, which gives the overall numerical order of the items returned starting with `0`, a parent field contains a `1` for parent items and a `0` for children (or modifiers).

As a result of this work, the `getitemexpanded` call now correctly handles modifiers. In addition, the `getitemexpanded` call now preserves the same order for items and modifiers being passed in as for those being returned.

With regard to the `<pass_no>` field, if the corresponding pass template contains a special macro (e.g., `SELECTLAST()ITEMSPECIAL ("ONLINE ")`), then a discounted price is returned in `price_xml` in the `[ext]` field. The specials can be any of the following in the macro for the pass template: `selectivespecial`, `globalspecial`, `itemspecial` or `special`. If the pass is voided or expired, then the discounted price is not returned. For an example, see getitem.

Input fields

XML tag	Description
`<qty>`	Optional. Quantity. Price returned is for specified quantity instead of per unit. If no `<qty></qty>` tags are passed, then `ww.dll` assumes the quantity is `1` and returns the price for a quantity of `1`. See How ww.dll reacts to quantity tags for more information.
`<item1>`	Optional. Provides the ability to pass in multiple DCIs. See example in main description of this API.
`<item2>`	Optional. Provides the ability to pass in multiple DCIs. See example in main description of this API.
`<department>`	`<item>` department.
`<category>`	`<item>` category.
`<item>`	Item.
`<pass_no>`	Optional. Pass number. If the corresponding pass template contains a special macro (e.g., `SELECTLAST()ITEMSPECIAL ("ONLINE ")`), then a discounted price

	is returned in `price_xml` in the `[ext]` field. The specials can be any of the following in the macro for the pass template: `selectivespecial`, `globalspecial`, `itemspecial` or `special`. If the pass is voided or expired, then the discounted price is not returned. For an example, see getitem.
`<calcprice>`	Optional. Boolean (`1` = calculate the price, `0` = don't calculate the price)
`<max4sale>`	Optional. Boolean (`1` = return Max4Sale information, `0` = don't return Max4Sale information)
`<calcpoints>`	Optional. Boolean. If `1`, a point cost of each item (at `qty=1`) is returned as `<pts_cost>`. Note that the `ckpts4sale` (is `Points4Sale` enforced) and `pts4saledp` (Dynamic Pricing rule to use to calculate Points4Sale points to deduct) fields from the items table must be obtained in the field list from the item record (e.g., `<fields>it.department, it.category, it.item, i.ckmax4sale, ckpts4sale, pts4saledp</fields>`). Also note that all item reservation functionality works the same for Points4Sale as it does for Max4Sale except that the total point cost is stored in the `item_res.pts4qty` field.
`<avail>`	Optional. Returns information on whether a particular item is available based on Max4Sale or Points4Sale restrictions for a certain date or date range. If the tag is absent or is specified as `<avail>0</avail>`, nothing is calculated. See Note on the <avail> tag.
`<datetime>`	Optional. Date/time of interest.
`<template>`	Optional. Gets template information for the item (if available).

`<account>`	Optional. An account nickname. When used, item price information is returned based on any dynamic pricing rule associated with the account. If a user is logged into a group (which is really an account), that group name is passed to `ww.dll` when getting product prices, and any dynamic pricing rule assigned to that account is applied to the prices.
	
	Note: In order for dynamic pricing rules assigned to accounts to work correctly in E-Commerce, you must add the following settings to the `[Preferences]` section of the `Sales32c.INI` file for the web Sales Host and the salespoint where you recall these sales:
	`[Preferences]`
	`ApplyAccountRulesOnRecall=FALSE`
	`RecalculatePriceOnRecall=FALSE`
	
	Without these settings in place, you see incorrect pricing on recall of these sales.

`<fields>`	Optional. Fields to be returned. See Note on the <fields> tag. Template fields do not need to be specified (and should not be) however in the `<fields>` tag.
`<hasmod>`	Optional. Modifier. See example in main description of this API.
`<mktgcode>`	Optional. Marketing code. See ww.dll now supports specials.
`<srcecode>`	Optional. Source code. See ww.dll now supports specials.

Return fields

XML recordset of the matching record from the items table. See `getitem`.

Note: Field names such as c60 that do not appear in the data dictionary are automatically generated when there are duplicate field names in joined datasets. These field names are not defined in the following table because their meanings vary. The description in the data set tells exactly what the duplicate field is. For example:

```
<s:AttributeType name='c52' rs:name='department' rs:number='53' rs:nullable='true'
                rs:writeunknown='true'>
          <s:datatype dt:type='string' rs:dbtype='str' dt:maxLength='10'
                rs:fixedlength='true'/>
</s:AttributeType
```

Field	Description
`parent`	Parent.
`department`	Department.
`category`	Category.
`item`	Item.
`item_id`	Item ID.
`descrip`	Description.
`gl_no`	General ledger number.
`user_code`	User code.
`type_1`	Type 1.
`type_2`	Type 2.
`type_3`	Type 3.
`type_4`	Type 4.
`type_5`	Type 5.
`type_6`	Type 6.
`type_7`	Type 7.
`type_8`	Type 8.
`type_9`	Type 9.
`type_10`	Type 10.
`security`	Security level.
`help_info`	Help information.
`start_show`	Start date/time to show.
`end_show`	End date/time to show.
`blackout1s`	Blackout 1 start.

blackout1e	Blackout 1 end.
blackout2s	Blackout 2 start.
blackout2e	Blackout 2 end.
blackout3s	Blackout 3 start.
blackout3e	Blackout 3 end.
blackout4s	Blackout 4 start.
blackout4e	Blackout 4 end.
start_time	Start time.
end_time	End time.
show_mon	Special usable on Mondays?
show_tue	Special usable on Tuesdays?
show_wed	Special usable on Wednesdays?
show_thu	Special usable on Thursdays?
show_fri	Special usable on Fridays?
show_sat	Special usable on Saturdays?
show_sun	Special usable on Sundays?
price_type	Price type.
dp_set_id	The set of dynamic pricing rules to apply first.
price_cols	Price columns – 1, 2 or 3
wknd_start	Weekend start.
wknd_end	Weekend end.
rate_sched	Rate schedule.
daily_pric	Daily price.
monday_0	Monday 1.
monday_1	Price during special rate period 1.
monday_2	Price during special rate period 2.
monday_3	Price during special rate period 3.
monday_4	Price during special rate period 4.
monday_5	Price during special rate period 5.
tuesday_0	Default price.
tuesday_1	Price during special rate period 1.
tuesday_2	Price during special rate period 2.
tuesday_3	Price during special rate period 3.
tuesday_4	Price during special rate period 4.

tuesday_5	Price during special rate period 5.
weds_0	Default price.
weds_1	Price during special rate period 1.
weds_2	Price during special rate period 2.
weds_3	Price during special rate period 3.
weds_4	Price during special rate period 4.
weds_5	Price during special rate period 5.
thursday_0	Default price.
thursday_1	Price during special rate period 1.
thursday_2	Price during special rate period 2.
thursday_3	Price during special rate period 3.
thursday_4	Price during special rate period 4.
thursday_5	Price during special rate period 5.
friday_0	Default price.
friday_1	Price during special rate period 1.
friday_2	Price during special rate period 2.
friday_3	Price during special rate period 3.
friday_4	Price during special rate period 4.
friday_5	Price during special rate period 5.
saturday_0	Default price.
saturday_1	Price during special rate period 1.
saturday_2	Price during special rate period 2.
saturday_3	Price during special rate period 3.
saturday_4	Price during special rate period 4.
saturday_5	Price during special rate period 5.
sunday_0	Default price.
sunday_1	Price during special rate period 1.
sunday_2	Price during special rate period 2.
sunday_3	Price during special rate period 3.
sunday_4	Price during special rate period 4.
sunday_5	Price during special rate period 5.
tax_rate	Tax rate.
tax_rate_b	Tax rate is defined in defaults table.
cust_tax	Custom tax rate.

cust_tax_b	Custom tax rate is defined in defaults table.
fee_rate	Fee rate.
add_tax	Add tax.
add_tax_b	Add tax b.
time_span	Time span.
span_type	Span type.
admissions	Admissions.
admprconly	Whether value in admissions field is only for price calculations and not for actual admissions.
min_qty	Minimum quantity to sell.
max_qty	Maximum quantity to sell.
prn_tkt1	Type of layout for ticket 1.
one_tkt1	Print one ticket 1, regardless of quantity.
prn_tkt2	Type of layout for ticket 2.
one_tkt2	Print one ticket 2, regardless of quantity.
prn_vouch1	Type of layout for voucher 1.
one_vch1	Print one voucher 1, regardless of quantity.
prn_vouch2	Type of layout for voucher 2.
one_vch2	Print one voucher 2, regardless of quantity.
prn_recpt	Print a receipt for this item?
split_type	Profit center split type – percentage or fill method.
pr_ctr_1	Profit center 1.
pcsplit_1	Profit center split 1.
pr_ctr_2	Profit center 2.
pcsplit_2	Profit center split 2.
pr_ctr_3	Profit center 3.
pcsplit_3	Profit center split 3.
pr_ctr_4	Profit center 4.
pcsplit_4	Profit center split 4.
pr_ctr_5	Profit center 5.
pcsplit_5	Profit center split 5.
pr_ctr_6	Profit center 6.
pcsplit_6	Profit center split 6.
item_type	Item type.

validate	Validate.
validate2	Additional validation.
last_mod	Last modified.
stockt1	Number of pieces of ticket 1 stock.
stockt2	Number of pieces of ticket 2 stock.
stockv1	Number of pieces of voucher 1 stock.
stockv2	Number of pieces of voucher 2 stock.
ckmax4sale	Check Max4Sale.
ckpts4sale	Check Points4Sale.
pts4saledp	DP Rule to use to calculate Points4Sale points to deduct.
ckm4s_rt	Perform real-time inventory checking in addition to regular Max4Sale checking?
dw_act	In-House Cards (Debitware) action.
do_on_sale	String of Sales32 commands that can be run at the salespoint when this item is sold.
novalonret	When a “forced validation” item is returned, should the salespoint make sure the pass is valid prior to returning it?
set_price	Set price.
inventory	Inventory.
invent_id	Inventory ID.
barcode	Barcode.
upc	UPC.
keycode	Keycode.
keydescrip	Key description.
vendor_1	Vendor 1.
vendor_2	Vendor 2.
reorder_pt	Re-order point.
target_qty	Target quantity.
max_tod	Number of minutes since midnight that a rental item must be returned by.
min_age	Minimum age.
max_age	Maximum age.
gf_cc	Operator is forced to enter credit card swipe for guest when selling this product.
gf_address	Guest address.

gf_zip	Guest zip.
gf_areacod	Guest area code.
gf_phone	Guest phone.
gf_birthda	Guest birthday.
gf_notes	Guest notes.
gf_number1	Guest number 1.
gf_number2	Guest number 2.
gf_number3	Guest number 3.
gf_number4	Guest number 4.
gf_number5	Guest number 5.
gf_text1	Guest text 1.
gf_text2	Guest text 2.
gf_text3	Guest text 3.
gf_text4	Guest text 4.
gf_text5	Guest text 5.
gf_date1	Guest date 1.
gf_dtime1	Guest date/time 1.
gf_memo1	Guest memo 1.
gf_city	Guest city.
gf_state	Guest state.
revnu_type	Link to the rev_type table for use in the revenue program.
rev_days	For revenue reporting module.
rev_money	For revenue reporting module.
rfnd_cond	Refund condition.
i_matrix	If this is an inventory item, does it use a matrix?
gstno_ret	Collect guest number when doing a return so that the return can be associated with the guest.
remote_p1	Should this item be printed on remote printer #1?
remote_p2	Should this item be printed on remote printer #2?
remote_p3	Should this item be printed on remote printer #3?
remote_p4	Should this item be printed on remote printer #4?
rtnvaloptn	Return validation option.
vpartno	Vendor part number.

role_no	Role number.
prevalidat	Pre-validation.
subclass	Subclass.
season	Season.
hidden	Hidden.
meet_reqd	Meeting location required.
gf_height	Guest height.
gf_weight	Guest weight.
mod_reqd	Modifier required?
mod_min	Minimum number of modifiers required to sell.
mod_max	Maximum number of modifiers required to sell.
multiline	Multiline.
do_on_web	String of Sales32 commands that can be run at the salespoint when this item is sold on a web sale.
web_rule	Web rule.
allow_pah	Allow Print At Home Tickets.
price_info	Price information. Includes final price, tax and fee information.
qty_rem	If <max4sale> is sent as true (<max4sale>1</max4sale>), then information is sent back with the number left to sell at that <datetime>. Otherwise, a - is returned in that field.
avail_info	Values returned as a result of using the <avail> tag. See Note on the <avail> tag.
pts_cost	If <calcpoints> is sent as true (<calcpoints>1</calcpoints>), then information is sent back with the points cost of each item (at qty=1).
price_xml	Price information. If the <pass_no> tag is sent in and the corresponding pass template contains a specials macro (e.g., SELECTLAST()ITEMSPECIAL("ONLINE ")), then a discounted price is returned in price_xml in the [ext] field. The specials can be any of the following in the macro for the pass template: selectivespecial, globalspecial, itemspecial or special. If the pass is voided or expired, then the discounted price is not returned. The [i2] tag returns the price of an item excluding taxes and before any discounts are applied. See main description of this getitem for an example.

Example

Example invocation:

```
<func>getitemexpanded</func><department>FBBAR</department><category>BEER</category><item>B
TL-COORS</item><calcprice>1</calcprice>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row ord='0' parent='1' department='FBBAR     ' category='BEER      '
             item='BTL-COORS ' item_id='475' descrip='Coors - Bottle          '
             gl_no='            ' user_code='                '
             type_1='True' type_2='True' type_3='True' type_4='True'
             type_5='True' type_6='True' type_7='True' type_8='True'
             type_9='True' type_10='True' security='0' help_info=''
             start_show='2006-10-18T00:00:00' end_show='2999-01-01T00:00:00'
             blackout1s='1900-01-01T00:00:00' blackout1e='1900-01-01T00:00:00'
             blackout2s='1900-01-01T00:00:00' blackout2e='1900-01-01T00:00:00'
             blackout3s='1900-01-01T00:00:00' blackout3e='1900-01-01T00:00:00'
             blackout4s='1900-01-01T00:00:00' blackout4e='1900-01-01T00:00:00'
             start_time='00:00' end_time='23:59' show_mon='True' show_tue='True'
             show_wed='True' show_thu='True' show_fri='True' show_sat='True'
             show_sun='True' price_type='0' dp_set_id='0' price_cols='1'
             wknd_start='0' wknd_end='0' rate_sched='1' daily_pric='1'
             monday_0='2.5' monday_1='0' monday_2='0' monday_3='0'
             monday_4='0' monday_5='0' tuesday_0='0' tuesday_1='0'
             tuesday_2='0' tuesday_3='0' tuesday_4='0' tuesday_5='0'
             weds_0='0' weds_1='0' weds_2='0' weds_3='0' weds_4='0'
             weds_5='0' thursday_0='0' thursday_1='0' thursday_2='0'
             thursday_3='0' thursday_4='0' thursday_5='0' friday_0='0'
             friday_1='0' friday_2='0' friday_3='0' friday_4='0' friday_5='0'
             saturday_0='0' saturday_1='0' saturday_2='0' saturday_3='0'
             saturday_4='0' saturday_5='0' sunday_0='0' sunday_1='0'
             sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0' tax_rate='2'
             tax_rate_b='1' cust_tax='.0000' cust_tax_b='.0000' fee_rate='1'
             add_tax='False' add_tax_b='False' time_span='0' span_type='1'
             admissions='.00' admprconly='False' min_qty='1' max_qty='999'
             prn_tkt1='1' one_tkt1='False' prn_tkt2='1' one_tkt2='False'
             prn_vouch1='1' one_vch1='False' prn_vouch2='1' one_vch2='False'
             prn_recpt='False' split_type='1' pr_ctr_1='7' pcsplit_1='100'
             pr_ctr_2='13' pcsplit_2='0' pr_ctr_3='13' pcsplit_3='0'
             pr_ctr_4='13' pcsplit_4='0' pr_ctr_5='13' pcsplit_5='0'
             pr_ctr_6='13' pcsplit_6='0' item_type='1' validate='1'
             validate2='False' last_mod='1120' stockt1='0' stockt2='0'
             stockv1='0' stockv2='0' ckmax4sale='False' ckpts4sale='False'
             pts4saledp='0' ckm4s_rt='False' dw_act='1'
do_on_sale='IGNOREERRORS()SPECIAL(&#x22;HAPPYHOUR &#x22;)
'
             novalonret='False' set_price='False' inventory='False'
             invent_id='0' barcode='                    ' upc='                    '
             keycode='0' keydescrip='          ' vendor_1='0' vendor_2='0'
             reorder_pt='.00' target_qty='.00' max_tod='0' min_age='.00'
             max_age='.00' gf_cc='False' gf_address='False' gf_zip='False'
             gf_areacod='False' gf_phone='False' gf_birthda='False'
             gf_notes='False' gf_number1='False' gf_number2='False'
             gf_number3='False' gf_number4='False' gf_number5='False'
             gf_text1='False' gf_text2='False' gf_text3='False' gf_text4='False'
             gf_text5='False' gf_date1='False' gf_dtime1='False' gf_memo1='False'
             gf_city='False' gf_state='False' revnu_type='0' rev_days='0'
             rev_money='0' rfnd_cond='' i_matrix='False' i_tmpnm_id='0'
             gstno_ret='False' remote_p1='False' remote_p2='False'
```

```
            remote_p3='False' remote_p4='False' rtnvaloptn='0' vpartno='
'
            role_no='0' prevalidat='False' subclass='          ' season='
'
            hidden='False' meet_reqd='False' gf_height='False' gf_weight='False'
            mod_reqd='0' mod_min='0' mod_max='99' multiline='False'
            do_on_web='' web_rule='' c202='True' c203='True' c204='True'
            c205='True' c206='True' c207='True' c208='True' allow_pah='False'
            price_info='2.5000,0.1636,0.0000,0.1636,0.0000'
price_xml='[ext]2.5000[/ext][tax]0.1636[/tax][fee]0.0000[/fee][txa]0.1636[/txa][txb]0.0000
[/txb][dwp]0.0000[/dwp][in]2.5000[/in][sp][/sp]'/>
</rs:data>
```

See also

[getitem](#)
[getitemtree](#)
[getmods](#)
[processsale](#)

getitemtree

Description

`getitemtree` takes the head node, the `<date_time>` of interest, and a list of fields to be returned (optional) and gets the next level of the item tree. Note that this function does *not* promote from lower levels (like the salespoint does when the `.INI` setting is activated).

`<calcprice>` uses the current date or the date passed in the `<date_time>` tag to calculate the price. The prices are included in the return string in the `price_info` field, including final price, tax and fee information. These should all be included in each `<item>` section when passed to the `processsale` function. If `<max4sale>` is sent as `true` (`<max4sale>1</max4sale>`), then information is sent back in the `qty_rem` field with the number left to sell at that `<datetime>`. Otherwise, a `-` will be returned in that field.

With regard to the `<pass_no>` field, if the corresponding pass template contains a special macro (e.g., `SELECTLAST()ITEMSPECIAL ("ONLINE ")`), then a discounted price is returned in `price_xml` in the `[ext]` field. The specials can be any of the following in the macro for the pass template: `selectivespecial`, `globalspecial`, `itemspecial` or `special`. If the pass is voided or expired, then the discounted price is not returned. For example, the following shows a sample call containing a pass number and what is returned in `price_xml`:

```
<func>getitem</func><department>TEST</department><category>TEST</
category><item>MAIN</item><pass_no>321003010</pass_no><calcprice>1</calcprice>

price_xml='[ext]98.00[/ext][tax]0.00[/tax][fee]0.00[/fee][txa]0.00[/txa]
[txb]0.00[/txb][dwp]0.00[/dwp][in]100.00[/in]'
```

Input fields

XML tag	Description
<qty>	Optional. Quantity. Price returned is for specified quantity instead of per unit. If no <qty></qty> tags are passed, then ww.dll assumes the quantity is 1 and returns the price for a quantity of 1. See How ww.dll reacts to quantity tags for more information.
<head> (or <headdesc>)	Head node, or 0 for the root.
<date_time>	Date/time of interest.

<account>	Optional. An account nickname. When used, item price information is returned based on any dynamic pricing rule associated with the account. If a user is logged into a group (which is really an account), that group name is passed to ww.dll when getting product prices, and any dynamic pricing rule assigned to that account is applied to the prices.
	
	Note: In order for dynamic pricing rules assigned to accounts to work correctly in E-Commerce, you must add the following settings to the [Preferences] section of the Sales32c.INI file for the web Sales Host and the salespoint where you recall these sales:
	`[Preferences]`
	`ApplyAccountRulesOnRecall=FALSE`
	`RecalculatePriceOnRecall=FALSE`
	
	Without these settings in place, you see incorrect pricing on recall of these sales.

<pass_no>	Optional. Pass number. If the corresponding pass template contains a special macro (e.g., SELECTLAST()ITEMSPECIAL ("ONLINE ")), then a discounted price is returned in price_xml in the [ext] field. The specials can be any of the following in the macro for the pass template: selectivespecial, globalspecial, itemspecial or special. If the pass is voided or expired, then the discounted price is not returned. For an example, see getitem.
<calcprice>	Boolean. 1 = calculate the price. 0 = don't calculate the price.
<max4sale>	Optional. Boolean. If Max4Sale is sent as true (<max4sale>1</max4sale>), then a tag is returned in the qty_rem field with the number left to sell at that <date>. Otherwise, a - is returned in that field.
<calcpoints>	Optional. Boolean. If 1, a point cost of each item (at qty=1) is returned as <pts_cost>. Note that the ckpts4sale (is Points4Sale enforced) and pts4saledp (Dynamic Pricing rule to use to calculate Points4Sale points to deduct) fields from the items table must be obtained in the field list from the item record (e.g., <fields>it.department, it.category, it.item, i.ckmax4sale, ckpts4sale, pts4saledp</fields>). Also note that all item reservation functionality works the same for Points4Sale as it does for Max4Sale except that the total point cost is stored in the item_res.pts4qty field.`
<avail>	Optional. Returns information on whether a particular item is available based on Max4Sale or Points4Sale restrictions for a certain date or date range. If the tag is absent or is specified as <avail>0</avail>, nothing is calculated. See Note on

	the <avail> tag.
<getchildnodes>	Optional. Returns all child items and groups as well. Boolean.
<fields>	Optional. Fields to be returned. Note on the <fields> tag.
<type_?>	Optional. Boolean for the type_1 to type_10 fields. Any fields that need to be matched can be specified. For example, if you only want matches where the type_10 is set to true, use <type_10>1</type_10>. If you wanted only matches where all types are 0 except for type 10, specify: <type_1>0</type_1><type_2>0</type_2> <type_3>0</type_3><type_4>0</type_4> <type_5>0</type_5><type_6>0</type_6> <type_7>0</type_7><type_8>0</type_8> <type_9>0</type_9><type_10>1</type_10>
<mktgcode>	Optional. Marketing code. See ww.dll now supports specials.
<srcecode>	Optional. Source code. See ww.dll now supports specials.

Return fields

XML recordset of a join between the itemtree table and the items table.

Note: For more information on the price_info, qty_rem and price_xml fields that are returned, see getitem, getitemexpanded and getmods.

Example

Example invocation:

```
<head>0</head><datetime>12-19-2006 07:00:00</datetime>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row descrip='**MISC**                ' department='          '
             category='          ' item='          ' node_id='621'
             parent_id='0' sort_order='1'/>
      <z:row descrip='**TIPS**                ' department='          '
             category='          ' item='          ' node_id='651'
             parent_id='0' sort_order='2'/>
      <z:row descrip='Golf - Tee Time Sched.  ' department='          '
             category='          ' item='          ' node_id='925'
             parent_id='0' sort_order='3'/>
      <z:row descrip='Retail                  ' department='          '
             category='          ' item='          ' node_id='1112'
             parent_id='0' sort_order='4'/>
      <z:row descrip='Food Service - Bar      ' department='          '
             category='          ' item='          ' node_id='9015'
             parent_id='0' sort_order='5'/>
      <z:row descrip='Food Service - Cafe/Coffe' department='          '
             category='          ' item='          ' node_id='1301'
             parent_id='0' sort_order='6'/>
```

```
      <z:row descrip='Food Service - Table Serv' department='          '
             category='          ' item='          ' node_id='1300'
             parent_id='0' sort_order='7'/>
      <z:row descrip='Admissions                ' department='          '
             category='          ' item='          ' node_id='1450'
             parent_id='0' sort_order='8'/>
      <z:row descrip='Kiosk                     ' department='          '
             category='          ' item='          ' node_id='1853'
             parent_id='0' sort_order='9'/>
      <z:row descrip='WEB                       ' department='          '
             category='          ' item='          ' node_id='1170'
             parent_id='0' sort_order='10'/>
      <z:row descrip='Rentals                   ' department='          '
             category='          ' item='          ' node_id='7521'
             parent_id='0' sort_order='11'/>
      <z:row descrip='Call Center               ' department='          '
             category='          ' item='          ' node_id='8350'
             parent_id='0' sort_order='12'/>
      <z:row descrip='TEST VISTA                ' department='          '
             category='          ' item='          ' node_id='18443'
             parent_id='0' sort_order='13'/>
</rs:data>
```

See also

[getitem](#)
[getitemexpanded](#)
[getmods](#)
[processsale](#)

getliabilityinfo

Description

`getliabilityinfo` looks up the description, minimum age to accept, the expiration days/date, and the text of the liability form(s) assigned to a particular item via `<department><category><item>` parameters. If the item specified does not have a liability form associated with it, `ww.dll` returns: “ERR:Function getliabilityinfo did not return information.” By including the optional `<guest_no>` parameter in the call, `ww.dll` returns either `1` (`true`) or `0` (`false`) in a column called “accepted” to denote whether the guest has accepted the liability form and whether it is still valid. If the form has expired for the guest, `ww.dll` returns `0` (`false`).

Input fields

XML tag	Description
`<department>`	`<item>` department.
`<category>`	`<item>` category.
`<item>`	Item.
`<guest_no>`	Optional. By including the optional `<guest_no>` parameter in the call, `ww.dll` returns either `1` (`true`) or `0` (`false`) in a column called “accepted” to denote whether the guest has accepted the liability form and whether it is still valid. If the

	form has expired for the guest, `ww.dll` returns `0` (`false`).

Return fields

If the item specified does not have a liability form associated with it, `ww.dll` returns: `ERR:Function getliabilityinfo did not return information.` By including the optional `<guest_no>` parameter in the call, `ww.dll` returns either `1` (`true`) or `0` (`false`) in a field called `accepted` to denote whether the guest has accepted the liability form and whether it is still valid. If the form has expired for the guest, `ww.dll` returns `0` (`false`).

Field	Description
`pri_key`	Primary key.
`department`	Department.
`category`	Category.
`item`	Item.
`liability`	Liability.
`sortorder`	Sort order.
`last_mod`	Last modified.
`descrip`	Description.
`liab_text`	Liability text.
`exp_static`	`0`=expires `exp_days` days after acceptance, `1`=expires on `exp_date`
`exp_days`	See `exp_static`.
`exp_date`	See `exp_static`.
`accept_age`	Guest age required to accept the form.
`hidden`	Hidden.
`accepted`	Boolean to denote whether the guest has accepted the liability form and whether it is still valid. Only returned if optional `<guest_no>` parameter is included in the call.

Example

Example invocation:

```
<func>getliabilityinfo</func><department>pufrentals</department>
<category>pufrntpkgs</ category ><item>bascskipkg</item><guest_no>577000000</guest_no>
```

Example return string:

```
OK :…recordset
<rs:data>
            <z:row pri_key='4000000' department='PUFRENTALS' category='PUFRNTPKGS'
               item='BASCSKIPKG' liability='1000000' sortorder='1' last_mod='25'
               c7='1000000' descrip='SKIING                    ' liab_text='Skiing is
               dangerous so be careful.'
               exp_static='False' exp_days='365' exp_date='2007-06-06T23:59:59'
               accept_age='18' hidden='False' c15='17' accepted='0'/>
</rs:data>
```

See also

`acceptliability`

getlog

Description

`getlog` causes a log file to be generated. You need to set `"VerbosityLevel"="5"` in the `ww.reg` file to obtain full output. (The `ww.reg` file is usually located in the same directory as `ww.dll`, though sometimes administrators delete it for security reasons – see the *Salesware E-Commerce* document for more information.)

Note: After you set `"VerbosityLevel"="5"` in the `ww.reg` file, be sure to double-click on the `ww.reg` file to update the registry. In addition, also be sure to shut down **wwSales** in **Administrative Tools** > **Component Services** > **Computers** > **My Computer** > **COM+ Applications**, which forces a "reboot" of `ww.dll` the next time it is invoked by a call to `ww.dll`.

Input fields

None.

Return fields

Logfile output.

Example

Example invocation:

```
<func>getlog</func>
```

Example return string:

```
OK :CIniFile/GetString: Key ConnectionString in Section Preferences Read as
Provider=SQLOLEDB.1;Password=genev1;Persist Security Info=True;User ID=siriusweb;Initial
Catalog=SiriusSQL;Data Source=INSPIRATION
CIniFile/SetString: Key ConnectionString in Section Preferences Added to INI File.
ConnectionString found and encrypted.
CIniFile/GetString: Key Protobase in Section CC Read as 000.000.000.000:4209
CIniFile/GetInt: Key Timeout in Section CC Read as 120
Unable to read registry: The system cannot find the file specified.

CIniFile/GetString: Key TerminalID in Section CC Read as RET
Unable to read registry: The system cannot find the file specified.

Unable to read registry: The system cannot find the file specified.

CIniFile/GetLogical: Key Enabled in Section CC Read as TRUE
CIniFile/GetString: Key Operator in Section CC Read as Webware
CIniFile/GetString: Key DefaultInfo in Section Preferences Read as
<operator>WEBOP1</operator><salespoint>WEBSP</salespoint>
CIniFile/GetString: Key TestCard in Section Preferences Read as 5454545454545454
Unable to read registry: The system cannot find the file specified.

Unable to read registry: The system cannot find the file specified.

Unable to read registry: The system cannot find the file specified.

Unable to read registry: The system cannot find the file specified.

Unable to read registry: The system cannot find the file specified.

MainEngine Initialization complete
CMainEngine::ProcessCall: Entering: <FUNC>GETLOG</FUNC>
CRecSet::Open Cmd=select site_no from prefs
CRecSet::Open returned 1 records
Record Retrieved: <site_no>1</site_no>
CRecSet::Open Cmd=select * from prefs_sl
CRecSet::Open returned 1 records
Record Retrieved: <ask_reprnt>.F.</ask_reprnt><cashdr_txt>Siriusware,
Inc.</cashdr_txt><closedrwrs>3</closedrwrs><conf_esubj>Your Reservation
Details</conf_esubj><conf_etxt>Thank you for choosing Siriusware, Inc.  We look forward to
seeing you!  Please review the attachment  regarding the details of your
reservation.</conf_etxt><conf_text></conf_text><fee_flat_1>0</fee_flat_1><fee_flat_2>0</fe
e_flat_2><fee_pct_1>1.5</fee_pct_1><fee_pct_2>2</fee_pct_2><finv_lay><|CENTER(dtoc(date())
,80)|>
<|CENTER(defaults->C_name,80)|>
<|CENTER(defaults->address,80)|>
<|CENTER(alltrim(defaults->city)+'  '+defaults->state+'  '+defaults->zip,80)|>
<|CENTER(defaults->area_code+defaults->phone,80)|>
<|CENTER(left(time(),5)+'     ACCOUNT  INVOICE     '+sale_hdr->salespoint,80)|>
<|NEWLINE()|>
<|CENTER(' Account Name: '+sale_hdr->acct_name,80)|>
<|CENTER('Invoice Number: '+ALLTRIM(STR(sale_hdr->invoice_no,16,0)),80)|>
<|CENTER('  Sale Number: '+ALLTRIM(STR(sale_hdr->sale_no,16,0)),80)|>
<|NEWLINE()|>
<|'Qty     Dept        Item         Special       Disc          Price        Ext'|>
<|'--------------------------------------------------------------------------------'|>
<|iterate_over(transact, Sale_no, Sale_hdr->Sale_no,STR(quantity,3,0)+'    '
+iif(transact->Item="**TRANS**","Account Transaction",Department+'   '+Item)+'
'+Transact->Special+'   '+STR(Transact->Disc_amt,6,2)+'     '+STR(transact-
>init_price,9,2)+'  '+STR(transact->extension,10,2),NEWLINE())|>
<|JUSTRIGHT('--------------------',80)|>
<|JUSTRIGHT('SUB TOTAL: '+STR(Utility->sale_sub,10,2),80)|>
<|JUSTRIGHT(ALLTRIM(defaults->taxname1)+': '+STR(Utility->sale_taxa,10,2),80)|>
<|JUSTRIGHT(ALLTRIM(defaults->taxname2)+': '+STR(Utility->sale_taxb,10,2),80)|>
<|JUSTRIGHT('--------------------',80)|>
<|JUSTRIGHT('CHARGE TO ACCOUNT: '+STR(Utility->sale_ext,10,2),80)|>
<|NEWLINE()|>
<|JUSTRIGHT('PAYMENT: '+STR(Utility->acct_pay,10,2),80)|>
<|JUSTRIGHT(FOP(),80)|>
<|JUSTRIGHT('--------------------',80)|>
```

```
<|JUSTRIGHT('INVOICE TOTAL: '+STR(Utility->inv_tot,10,2),80)|>
<|JUSTRIGHT('ACCOUNT TOTAL: '+STR(Utility-
>acct_tot,10,2),80)|></finv_lay><frecpt_lay><|CENTER(defaults->C_name,40)|>
<|CENTER(defaults->address,40)|>
<|CENTER(alltrim(defaults->city)+'  '+defaults->state+'  '+defaults->zip,40)|>
<|NEWLINE()|>
<|CENTER('SALES RECEIPT',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR
(sale_hdr->mastersale,16,0)),40)|>
<|CENTER(sale_hdr->operator+'     '+sale_hdr->salespoint,40)|>
<|CENTER(DSTR2(sale_hdr->Date_time),40)|>
<|NEWLINE()|>
<|'Qty  Item                      Price   '|>
<|'     Special                   Discount'|>
<|'----------------------------------------'|>
<|Details(IIF(Item="**TRANS**",'     Account Transaction
'+STR(extension,8,2),STR(quantity,3,0)+'  '+Items->descrip+'  '+IIF(extension+disc_amt=0,'
',STR(quantity*init_price,8,2))+IIF(Items->I_Matrix,NEWLINE()+'
'+ALLTRIM(I_Items->Descrip),'')+IIF(disc_amt=0,NEWLINE()+ALLTRIM('
'),NEWLINE()+'     '+IIF(ALLTRIM(UPPER(special))="CUSTOM","Custom Special
",specials->descrip)+'  '+STR(-1*disc_amt,8,2)+NEWLINE())|>
<|JUSTRIGHT('-------------------------',40)|>
<|JUSTRIGHT('       SUB TOTAL:'+STR(Utility->sale_sub,10,2),40)|>
<|JUSTRIGHT('             TAX:'+STR(Utility->sale_tax,10,2),40)|>
<|JUSTRIGHT('-------------------------',40)|>
<|JUSTRIGHT('          TOTAL:'+STR(Utility->sale_ext,10,2),40)|>
<|JUSTRIGHT('          PAYMENTS:'+STR(Utility->amt_paid,10,2),40)|>
<|NEWLINE()|>
<|JUSTRIGHT('          BALANCE DUE:'+STR(Utility->bal_due,10,2),40)|>
<|NEWLINE()|>

<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|Printers->cut_code|>
<|NEWLINE()|>
<|NEWLINE()|></frecpt_lay><full_inv>.F.</full_inv><get_zipcod>0</get_zipcod><golf_tree>0</
golf_tree><invoic_lay><|CENTER(dtoc(date()),40))|>
<|CENTER(defaults->C_name,40)|>
<|CENTER(defaults->address,40)|>
<|CENTER(alltrim(defaults->city)+'  '+defaults->state+'  '+defaults->zip,40)|>
<|CENTER(defaults->area_code+defaults->phone,40)|>
<|CENTER(left(time(),5)+'     ACCOUNT  INVOICE     '+sale_hdr->salespoint,40)|>
<|NEWLINE()|>
<|CENTER(' Account Name: '+sale_hdr->acct_name,40)|>
<|CENTER('Invoice Number: '+ALLTRIM(STR(sale_hdr->invoice_no,16,0)),40)|>
<|CENTER('  Sale Number: '+ALLTRIM(STR(sale_hdr->sale_no,16,0)),40)|>
<|NEWLINE()|>
<|'  Admit     Dept     Item           Qty'|>
<|'Special     Disc     Price          Ext'|>
<|'----------------------------------------'|>
<|iterate_over(transact, Sale_no, Sale_hdr->Sale_no,STR(admissions*quantity,3,0)+'      '
+iif(transact->Item="**TRANS**","Account Transaction",Department+' '+Item)+'      '+
STR(quantity,3,0)+NEWLINE()+Transact->Special+STR(Transact->Disc_amt,6,2)+STR(transact-
>init_price,9,2)+'     '+STR(transact->extension,10,2),NEWLINE())|>
<|JUSTRIGHT('--------------------',40)|>
<|JUSTRIGHT('SUB TOTAL: '+STR(Utility->sale_sub,10,2),40)|>
<|JUSTRIGHT(ALLTRIM(defaults->taxname1)+': '+STR(Utility->sale_taxa,10,2),40)|>
<|JUSTRIGHT(ALLTRIM(defaults->taxname2)+': '+STR(Utility->sale_taxb,10,2),40)|>
<|JUSTRIGHT('--------------------',40)|>
<|JUSTRIGHT('CHARGE TO ACCOUNT: '+STR(Utility->sale_ext,10,2),40)|>
<|NEWLINE()|>
<|JUSTRIGHT('PAYMENT: '+STR(Utility->acct_pay,10,2),40)|>
<|JUSTRIGHT(FOP(),40)|>
<|JUSTRIGHT('--------------------',40)|>
<|JUSTRIGHT('INVOICE TOTAL: '+STR(Utility->inv_tot,10,2),40)|>
```

```
<|JUSTRIGHT('ACCOUNT TOTAL: '+STR(Utility-
>acct_tot,10,2),40)|></invoic_lay><invt_def><ADJUSTFOCUS>DeptCombo</ADJUSTFOCUS><DETAILFIL
TER>1</DETAILFILTER><DAMAGEFOCUS>DeptCombo</DAMAGEFOCUS><TALLYLINEEDIT_FOCUS>txt_UPC</TALL
YLINEEDIT_FOCUS></invt_def><keepsspric>.F.</keepsspric><last_mod>154</last_mod><max4_saved
>2</max4_saved><pole_msg1></pole_msg1><pole_msg2></pole_msg2><pole_msg3></pole_msg3><pole_
msg4></pole_msg4><pole_msg5></pole_msg5><prefs_ini></prefs_ini><pri_key>1</pri_key><recpt_
lay><|CENTER(defaults->C_name,40)|>
<|CENTER(defaults->address,40)|>
<|CENTER(alltrim(defaults->city)+'  '+defaults->state+'  '+defaults->zip,40)|>
<|NEWLINE()|>
<|CENTER('SALES RECEIPT ',40)|>
<|CENTER('Sale Number: '+IIF(sale_hdr->sale_no=sale_hdr->mastersale,ALLTRIM(STR(sale_hdr-
>mastersale,16,0)),ALLTRIM(STR(sale_hdr->sale_no))),40)|>
<|CENTER(sale_hdr->operator+'     '+sale_hdr->salespoint,40)|>
<|CENTER(DSTR2(sale_hdr->Date_time),40)|>
<|NEWLINE()|>
<|'Qty  Item                      Price   '|>
<|'     Special                   Discount'|>
<|'--------------------------------------'|>
<|Details(IIF(Item="**TRANS**",'     Account Transaction
'+STR(extension,8,2),IIF(MODIFIER(),"
",STR(quantity,3,0)+'  '+Items->descrip+'  '+IIF(MASKPRICE(),'    -
',STR(quantity*init_price,8,2))))+NEWLINE()+IIF(disc_amt=0,ALLTRIM('
'),'     '+IIF(ALLTRIM(UPPER(special))="CUSTOM","Custom Special          ",specials-
>descrip)+'  '+STR(-1*disc_amt,8,2))+NEWLINE()+IIF(guest_no=0,ALLTRIM('
'),alltrim(guests.first_name)+' '+alltrim(guests.last_name))+IIF(pass_no=0,ALLTRIM('
'),'     Starting Number: '+ALLTRIM(STR(pass_no,16,0))+ NEWLINE()+NEWLINE()))|>


<|JUSTRIGHT('--------------------------',40)|>
<|JUSTRIGHT('       SUB TOTAL:'+STR(Utility->sale_sub,10,2),40)|>
<|JUSTRIGHT('             TAX:'+STR(Utility->sale_tax,10,2),40)|>
<|JUSTRIGHT('--------------------------',40)|>
<|JUSTRIGHT('          TOTAL:'+STR(Utility->sale_ext,10,2),40)|>
<|NEWLINE()|>
<|JUSTRIGHT('          PAYMENTS:'+STR(Utility->amt_paid,10,2),40)|>
<|JUSTRIGHT('--------------------------',40)|>
<|JUSTRIGHT('          BALANCE DUE:'+STR(Utility->bal_due,10,2),40)|>
<|NEWLINE()|>
<|JUSTRIGHT('CHANGE: '+STR(Utility->change,10,2),40)|>
<|NEWLINE()|>
<|JUSTRIGHT('Payment Type: ',40)|>
<|JUSTRIGHT(FOP(),40)|>
<|NEWLINE()|>
<|JUSTRIGHT(IIF(cc_trans->gc_balance=0.0000,'
','DEBITWARE BALANCE:'+STR(cc_trans->gc_balance,10,2)),40)|>
<|iterate_over(tr_info,sale_no,sale_hdr->sale_no,IIF(info_type=70,'Ticket Number:
'+ALLTRIM(info_num),''),NEWLINE())|>
<|iterate_over(tr_info,sale_no,sale_hdr->sale_no,IIF(info_type=20,'Pass Number:
'+ALLTRIM(info_num),''),NEWLINE())|>
<|NEWLINE()|>
<|Printers->cut_code|>
<|NEWLINE()|>
<|NEWLINE()|></recpt_lay><return_rct>.F.</return_rct><rmtlay2all><|alltrim(printers-
>color_1)+CENTER('KITCHEN RECEIPT',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(utility->sale_no,16,0)),40)|>
<|NEWLINE()|>
<|CENTER('Order Number: '+ALLTRIM(STR((utility->sale_no-500000)/1000000,16,0)),40)|>
<|NEWLINE()|>
<|CENTER(dtoc(date())+' '+time(),40)|>
<|NEWLINE()|>
<|'Qty - Item'|>
<|'--------------------------------------'|>
<|Details(IIF(MODIFIER(),alltrim(printers->color_2)+NEWLINE()+'       ATTN:
'+ALLTRIM(alltrans->descrip)+' '+ALLTRIM(alltrans->message),alltrim(printers-
>color_1)+NEWLINE()+STR(DeltaQty,3,0)+' - ' +ALLTRIM(alltrans->descrip)+ALLTRIM(alltrans-
>message)+NEWLINE())+NEWLINE())|></rmtlay2all><rmtlay2new><|alltrim(printers-
>color_1)+CENTER('KITCHEN RECEIPT',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(utility->sale_no,16,0)),40)|>
```

```
<|NEWLINE()|>
<|CENTER('Order Number: '+ALLTRIM(STR((utility->sale_no-500000)/1000000,16,0)),40)|>
<|NEWLINE()|>
<|CENTER(dtoc(date())+' '+time(),40)|>
<|NEWLINE()|>
<|'Qty - Item'|>
<|'----------------------------------------'|>
<|Details(IIF(MODIFIER(),alltrim(printers->color_2)+NEWLINE()+'       ATTN:
'+ALLTRIM(alltrans->descrip)+' '+ALLTRIM(alltrans->message),alltrim(printers-
>color_1)+NEWLINE()+STR(DeltaQty,3,0)+' - ' +ALLTRIM(alltrans->descrip)+ALLTRIM(alltrans-
>message)+NEWLINE())+NEWLINE(),TRUE)|></rmtlay2new><rmtlay3all></rmtlay3all><rmtlay3new></
rmtlay3new><rmtlay4all></rmtlay4all><rmtlay4new></rmtlay4new><rmtlay5all><|CENTER('KITCHEN
RECEIPT ALL',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(utility->sale_no,16,0)),40)|>
<|CENTER('Table Number: '+ALLTRIM(utility->last_name),40)|>
<|CENTER(iif(ALLTRIM(utility->first_name)="",'','Split Number: '+ALLTRIM(utility-
>first_name)),40)|>
<|CENTER(dtoc(date())+' '+time(),40)|>
<|NEWLINE()|>
<|'Qty - Item'|>
<|'----------------------------------------'|>
<|Details(IIF(MODIFIER(),'       ATTN: '+alltrans->Descrip+' '+ALLTRIM(alltrans-
>Message),NEWLINE()+STR(DeltaQty,3,0)+' - ' +iif(Item="**TRANS**","Account
Transaction",alltrans->descrip+' '+ALLTRIM(alltrans-
>message)))+NEWLINE())|></rmtlay5all><rmtlay5new><|alltrim(printers-
>color_1)+CENTER('KITCHEN RECEIPT ADDED',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(utility->sale_no,16,0)),40)|>
<|NEWLINE()|>
<|CENTER('Table Number: '+ALLTRIM(utility->first_name),40)|>
<|CENTER(iif(ALLTRIM(utility->first_name)="",'','Split Number: '+ALLTRIM(utility-
>first_name)),40)|>
<|CENTER('Server: '+ALLTRIM(utility->operator),40)|>
<|CENTER('Description: '+ALLTRIM(utility->descrip1),40)|>
<|NEWLINE()|>
<|CENTER(dtoc(date())+' '+time(),40)|>
<|NEWLINE()|>
<|'Qty - Item'|>
<|'----------------------------------------'|>
<|Details(IIF(MODIFIER(),alltrim(printers->color_2)+NEWLINE()+'       ATTN:
'+ALLTRIM(alltrans->descrip)+' '+ALLTRIM(alltrans->message),alltrim(printers-
>color_1)+NEWLINE()+STR(DeltaQty,3,0)+' - ' +ALLTRIM(alltrans->descrip)+ALLTRIM(alltrans-
>message))+NEWLINE(),TRUE)|></rmtlay5new><rmtlay6all></rmtlay6all><rmtlay6new></rmtlay6new
><rmtlay7all></rmtlay7all><rmtlay7new></rmtlay7new><rmtlay8all><|CENTER(defaults-
>C_name,40)|>
<|CENTER(defaults->address,40)|>
<|CENTER(alltrim(defaults->city)+'  '+defaults->state+'  '+defaults->zip,40)|>
<|NEWLINE()|>
<|CENTER('SALES RECEIPT',40)|>
<|CENTER(operator->first_name+'     '+sh_save->salespoint,40)|>
<|CENTER(DSTR2(sh_save->Date_time),40)|>
<|NEWLINE()|>
<|CENTER('Sale Number: '+ALLTRIM(STR(sh_save->sale_no,16,0)),40)|>
<|CENTER(ALLTRIM(utility->first_name),40)|>
<|CENTER('PLEASE PRESENT TO PICK UP YOUR GIFT CERTIFICATE',40)|>
<|CENTER('ANYTIME AFTER MAY 1 2007',40)|>
<|NEWLINE()|>
<|'Qty  Item                      Price   '|>
<|'     Special                   Discount'|>
<|'----------------------------------------'|>
<|Details(IIF(Item="**TRANS**",'     Account Transaction
'+STR(extension,8,2),STR(quantity,3,0)+'  '+Items->descrip+'
'+STR(quantity*init_price,8,2))+NEWLINE()+IIF(disc_amt=0,ALLTRIM('
'),'     '+IIF(ALLTRIM(UPPER(special))="CUSTOM","Custom Special          ",specials-
>descrip)+'  '+STR(-1*disc_amt,8,2)+NEWLINE()))|>
<|JUSTRIGHT('-------------------------',40)|>
<|JUSTRIGHT('       SUB TOTAL:'+STR(Utility->sale_sub,10,2),40)|>
<|JUSTRIGHT('             TAX:'+STR(Utility->sale_tax,10,2),40)|>
<|JUSTRIGHT('-------------------------',40)|>
<|JUSTRIGHT('          TOTAL:'+STR(Utility->sale_ext,10,2),40)|>
```

```
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|Printers->cut_code|>
<|NEWLINE()|>
<|NEWLINE()|></rmtlay8all><rmtlay8new></rmtlay8new><rmtlay_all><|alltrim(printers-
>color_1)+CENTER('KITCHEN RECEIPT ALL',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(utility->sale_no,16,0)),40)|>
<|NEWLINE()|>
<|CENTER('Table Number: '+ALLTRIM(utility->first_name),40)|>
<|CENTER(iif(ALLTRIM(utility->first_name)="",'','Split Number: '+ALLTRIM(utility-
>first_name)),40)|>
<|CENTER('Server: '+ALLTRIM(utility->operator),40)|>
<|CENTER('Description: '+ALLTRIM(utility->descrip1),40)|>
<|CENTER('# of Guests: '+ALLTRIM(str(utility->descrip2)),40)|>
<|NEWLINE()|>
<|'Qty - Item'|>
<|'----------------------------------------'|>
<|Details(IIF(MODIFIER(),alltrim(printers->color_2)+NEWLINE()+'       ATTN:
'+ALLTRIM(alltrans->descrip)+' '+ALLTRIM(alltrans->message),alltrim(printers-
>color_1)+NEWLINE()+STR(DeltaQty,3,0)+' - ' +ALLTRIM(alltrans->descrip)+ALLTRIM(alltrans-
>message))+NEWLINE())|></rmtlay_all><rmtlay_new><|alltrim(printers-
>color_1)+CENTER('KITCHEN RECEIPT ADDED',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(utility->sale_no,16,0)),40)|>
<|NEWLINE()|>
<|CENTER('Table Number: '+ALLTRIM(utility->first_name),40)|>
<|CENTER(iif(ALLTRIM(utility->first_name)="",'','Split Number: '+ALLTRIM(utility-
>first_name)),40)|>
<|CENTER('Server: '+ALLTRIM(utility->operator),40)|>
<|CENTER('Description: '+ALLTRIM(utility->descrip1),40)|>
<|CENTER('# of Guests: '+ALLTRIM(str(utility->descrip2)),40)|>
<|NEWLINE()|>
<|CENTER(dtoc(date())+' '+time(),40)|>
<|NEWLINE()|>
<|'Qty - Item'|>
<|'----------------------------------------'|>
<|Details(IIF(MODIFIER(),alltrim(printers->color_2)+NEWLINE()+'       ATTN:
'+ALLTRIM(alltrans->descrip)+' '+ALLTRIM(alltrans->message),alltrim(printers-
>color_1)+NEWLINE()+STR(DeltaQty,3,0)+' - ' +ALLTRIM(alltrans->descrip)+ALLTRIM(alltrans-
>message))+NEWLINE(),TRUE)|></rmtlay_new><saved_lay><|CENTER(defaults->C_name,40)|>
<|CENTER(defaults->address,40)|>
<|CENTER(alltrim(defaults->city)+'  '+defaults->state+'  '+defaults->zip,40)|>
<|NEWLINE()|>
<|CENTER('SALES RECEIPT',40)|>
<|CENTER('Sale Number: '+ALLTRIM(STR(sh_save->sale_no,16,0)),40)|>
<|CENTER('Table Number: '+ALLTRIM(utility->first_name),40)|>
<|CENTER(operator->first_name+'     '+sh_save->salespoint,40)|>
<|CENTER(DSTR2(sh_save->Date_time),40)|>
<|NEWLINE()|>
<|'Qty  Item                      Price   '|>
<|'     Special                   Discount'|>
<|'----------------------------------------'|>
<|Details(IIF(Item="**TRANS**",'     Account Transaction
'+STR(extension,8,2),STR(quantity,3,0)+'  '+Items->descrip+'
'+STR(quantity*init_price,8,2))+NEWLINE()+IIF(disc_amt=0,ALLTRIM('
'),'     '+IIF(ALLTRIM(UPPER(special))="CUSTOM","Custom Special          ",specials-
>descrip)+'  '+STR(-1*disc_amt,8,2)+NEWLINE()))|>
<|JUSTRIGHT('--------------------------',40)|>
<|JUSTRIGHT('       SUB TOTAL:'+STR(Utility->sale_sub,10,2),40)|>
<|JUSTRIGHT('             TAX:'+STR(Utility->sale_tax,10,2),40)|>
<|JUSTRIGHT('--------------------------',40)|>
<|JUSTRIGHT('          TOTAL:'+STR(Utility->sale_ext,10,2),40)|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
<|NEWLINE()|>
```

```
<|Printers->cut_code|>
<|NEWLINE()|>
<|NEWLINE()|></saved_lay><sched1_p1e>01/02/2007
23:59:59</sched1_p1e><sched1_p1s>12/22/2006 00:00:00</sched1_p1s><sched1_p2e>01/15/2007
23:59:59</sched1_p2e><sched1_p2s>01/15/2007 00:00:00</sched1_p2s><sched1_p3e>02/19/2007
23:59:59</sched1_p3e><sched1_p3s>02/19/2007 00:00:00</sched1_p3s><sched4_p1e>12/31/2006
23:59:59</sched4_p1e><sched4_p1s>12/26/2006 00:00:00</sched4_p1s><sched4_p2e>02/23/2007
23:59:59</sched4_p2e><sched4_p2s>02/17/2007 00:00:00</sched4_p2s><sched4_p3e>01/31/2007
23:59:59</sched4_p3e><sched4_p3s>01/01/2007 00:00:00</sched4_p3s><sp_ini><defaults>

[Preferences]
InventoryLookupTimeout=30
ReindexItemsOnStartup=FALSE
VoidAfterOrder=FALSE
ValidateZip=TRUE
LimitLocalItems=FALSE
ItemTreePromotion=FALSE
AlwaysSure=FALSE
SalesSummary=TRUE
SpecialConfirmation=FALSE
ScanCharacterInterval=300
GuestAnywhere=TRUE
CountryCode=USA
CardActionDigits=6
AutoRenew=TRUE
SortPmtType=TRUE
ReasonsForRefunds=TRUE
ReasonsForSaleRefund=TRUE
AutoFillCloseOut=TRUE
CloseoutPmtBtn=TRUE
PartialFinalize=TRUE
ChangeLogForAll=TRUE


DWLineItemLink=FALSE
PassSwipeUnique=TRUE
AutoDebitwareInvoiceLookup=TRUE

ShowAdditionalNumber=TRUE
CardOnFile=TRUE



ItemDesc=<items->descrip>     <guests->first_name> <guests->last_name>   <date>
<ifspecial><specials->descrip> -<disc_amt></ifspecial>     <ifacct><accounts-
>full_name></ifacct>
ItemDescMod=<items->descrip>     <guests->first_name>     <date>
<ifspecial><cr><specials->descrip></ifspecial>     <ifacct><accounts->full_name></ifacct>
ItemDescFull=<items->descrip>     <guests->first_name>     <date>
<ifspecial><specials->descrip> -$<disc_amt></ifspecial>     <ifacct><accounts-
>full_name></ifacct>

SetDateOnRecall=FALSE
;SetDateOnRecall=TRUE
;For ASC -- The default is TRUE (that when a sale is recalled, the "big" date time button
is always set.)  Setting this to FALSE ensures that it will not be set unless changed
explicitly before saving.

RequireLiabilityOnSave=FALSE


;PodLabel=Facilitiy Bookings
TBDPrivateBooking=FALSE
;BookingSeriesNotRequests=TRUE
FinalizeLockedBookings=FALSE
;BookEZReconnectThreshold=1.5
;BookEZReconnectInterval=10
```

```
;SettleByOperator=TRUE
;Lumps batches together by operator instead of salespoint for Cardware reports in
ReportManager and sends operator nickname to protobase instead of salespoint nickname
(again for reporting or grouping in PBAdmin).  Salespoint nickname is the default setting.
;ForwardAuthOnReconnect=TRUE (Used with Protobase to allow the operator to forward cc
auths at the next finalize)
AutoPopulateAVS=TRUE
UseNewSockeType=TRUE

;These are sample hot key configuration...
;F11Text="%E"
;F11Track=2
;F11Prompt="Please Swipe Discount Card"

;F12Text="%A"
;F12Track=1
;F12Prompt="Please Enter Voucher Number"

F12Text=%AAAA

;RECAP SETTINGS
RecapTextBegin=Enter any system wide text you want right here such as instructions to your
operators... <cr>_____________________________________________________________

ResDescRecap=Guest:  <first_name> <last_name><cr>Status:  <sale_status><cr>Reservation#:
<reserv_no><cr>Confirmation#:  <user_resno><cr>Arrival Date:  <start_date><cr>Pickup
Location:  <pickup_loc><cr>Base Lodge: <base_lodge><cr>Accommodations:
<accommodat><cr>Print Tickets:  <Print_due><cr>Final Payment Due:
<final_due><cr>Tracking#:  <tracking1., <tracking2>

;ItemDescRecap option - shows the price per for both the main item and the modifiers
;ItemDescRecap=_____________________________________________________<cr><ret_stat><
cr>Qty:  <qty>     Item:  <items->descrip>     <ifspecial>Special: <specials->descrip>
Discount Amount: <tr_save->disc_amt></ifspecial>    Cost:
$<ext>each<cr><cr><ifmods>Activities:<cr></ifmods><ifmods><formods>Qty:  <qty>     <items-
>descrip>     <ifspecial>Special: <specials->descrip>     Discount Amount: <tr_save-
>disc_amt></ifspecial>     Cost:
$<ext>each<cr></formods></ifmods><cr>___________<cr><cr>Total:
$<exttotal><cr>___________<cr><cr>Start Date:  <date><cr><cr><ifguest><forguests>Guest:
<guests->first_name> <guests-
>last_name><cr></forguests></ifguest><cr><ifpod><forbooks>Resource: <b_sched->podname>
Booking Date/Time: <b_sched-
>start_time><cr></forbooks></ifpod><ifprivate><forbooks>Instructor: <b_sched->instrname>
Lesson Date/Time: <b_sched->start_time><cr>Booking Number:  <b_sched-
>booking_ID><cr><b_sched->loc_id></forbooks></ifprivate><cr><iftee><fortee>Tee-time:
<res_schd->start_time></fortee></iftee><cr>Meeting Location:
<meet_loc><cr><iffinalized>Printed On:  <tr_save->date_time></iffinalized>
<ifpass><forpass>Pass Number: <gst_pass->pass_no><cr>Pass Valid Dates: <gst_pass-
>start_date> through <gst_pass->expires></forpass></ifpass>

;ItemDescRecap option - eliminates the cost on mods and rolls everything up to the total
line item cost
ItemDescRecap=_____________________________________________________<cr><ret_stat><c
r>Qty:  <qty>     Item:  <items->descrip>     <ifspecial>Special: <specials->descrip>
Discount Amount: <tr_save->disc_amt></ifspecial>    Total:
$<exttotal><cr><cr><ifmods>Includes:<cr></ifmods><ifmods><formods>     <items->descrip>
<r_days>   <r_daytype>     <ifspecial>Special: <specials->descrip>     Discount Amount:
<tr_save->disc_amt></ifspecial><cr></formods></ifmods><cr><cr>Start Date:
<date><cr><cr><ifguest><forguests>Guest: <guests->first_name> <guests-
>last_name><cr></forguests></ifguest><cr><ifpod><forbooks>Resource: <b_sched->podname>
Booking Date/Time: <b_sched-
>start_time><cr></forbooks></ifpod><ifprivate><forbooks>Instructor: <b_sched->instrname>
Lesson Date/Time: <b_sched->start_time><cr>Booking Number:  <b_sched-
>booking_ID><cr><b_sched->loc_id></forbooks></ifprivate><cr><iftee><fortee>Tee-time:
<res_schd->start_time></fortee></iftee><cr>Meeting Location:
<meet_loc><cr><iffinalized>Printed On:  <tr_save->date_time></iffinalized>
<ifpass><forpass>Pass Number: <gst_pass->pass_no><cr>Pass Valid Dates: <gst_pass-
>start_date> through <gst_pass->expires></forpass></ifpass>
```

```
ResDescRecap2=_____<cr>Payment
Summary<cr><cr>Reservation Grand Total: <res_total><cr>Total Payments Made:
<amt_paid><cr>Balance Due: <bal_due><cr>Forfeited Amount: <fft_total><cr>Refunded Amount:
<ref_total><cr>_____<cr>Payment
Detail<cr><cr><forpmts>Payment Date: <pmt->Date_time><cr>Amount: <pmt->amount><cr>Payment
Type: <pmt->Pmt_type><cr><iscard>Card Number: <pmt->Card_id></iscard><cr>At Salespoint:
<pmt->Salespoint><cr>By Operator: <pmt->Operator><cr><cr></forpmts>


RecapTextEnd=_____<cr>Enter more
system wide text here, such as a cancellation policy or anything else your operators
should communicate to your guests...

;Ticket Header Samples for Batch Ticket Printing
;Blaster
;BatchTicketLayout=<?Style=1?>! 0 100 1050 1<cr>JUSTIFY LEFT<cr>U A30 (2,1,0) 275 120
First Name: <!first_name><cr>U A30 (2,1,0) 275 150 Last Name: <!last_name><cr><cr>U A30
(2,1,0) 275 190 Reservation Number: <!reserv_no><cr>U A30 (2,1,0) 275 220 Confirmation
Number: <!user_resno><cr><cr>U A30 (2,1,0) 275 270 Arrival Date: <!start_date><cr>U A30
(2,1,0) 275 300 Time: <!resrvatn.user_code1><cr><cr>U A30 (2,1,0) 275 330 Pickup Location:
<!pickup_loc><cr>U A30 (2,1,0) 275 370 Accommodations: <!accommodat><cr>U A30 (2,1,0) 275
400 Base: <!base_lodge><cr><cr>U A30 (2,1,0) 275 430 Type: <!resrvatn.user_code2><cr>U A30
(2,1,0) 275 460 Lunch: <!resrvatn.user_code3><cr>END

;Boca
;BatchTicketLayout=<?Style=1?><|alltrim("<NR><RC200,200><F9><HW4,2>")+alltrim("<!first_nam
e> <!last_name>")|><cr><|alltrim("<NR><RC300,200><F8><HW2,1>")+alltrim("<!reserv_no>
<!pickup_loc>")|><cr><|alltrim("<NR><RC400,200><F8><HW2,1>")+alltrim("<!start_date>
<!resrvatn.user_code1>")|><cr><|alltrim("<p>")|>

ForceRentalCheckIn=TRUE

[Rental]
FormIsTrans=TRUE
LockFormNumber=FALSE
SelfEntry=TRUE


[Fonts]
BigFontHeight=18
BigFontWeight=SEMIBOLD
BigFontFace=Segoe UI
BigFont2Height=18
BigFont2Weight=BOLD
BigFont2Face=Segoe UI
StdFontHeight=18
StdFontWeight=SEMIBOLD
StdFontFace=Segoe UI
SmlFontHeight=14
SmlFontWeight=SEMIBOLD
SmlFontFace=Segoe UI
SmlFont2Height=18
SmlFont2Weight=REGULAR
SmlFont2Face=Segoe UI
TxtFontHeight=16
TxtFontWeight=REGULAR
TxtFontFace=Segoe UI
LineItemFontHeight=20
LineItemFontWeight=REGULAR
LineItemFontFace=Segoe UI


;The choices for the font weight are (case doesn't matter):
;THIN
;EXTRALIGHT
;ULTRALIGHT
;LIGHT
;NORMAL
;REGULAR
```

```
;MEDIUM
;SEMIBOLD
;DEMIBOLD
;BOLD
;EXTRABOLD
;ULTRABOLD
;HEAVY
;BLACK


[Interface]
ShowImages=TRUE
PickWidth=45
Buttons=TRUE
ButtonCols=3
ButtonRows=10
MainButton1=FIN
MainButton2=CLR
MainButton3=PRN
MainButton4=SCH
MainButton5=SAV
MainButton6=GST
MainButton7=LOG
MainButton8=CLS
MainButton9=TAX
MainButton10=INV
MainButton11=TLS
MainButton12=DWR
QuickSaveRows=1
QuickSaveColumns=5

;[SerialReader]
;Type=1
;ComPort=4
;BaudRate=38400
;Parity=N
;DataBits=8
;StopBits=1
;PollInterval=200
;SwipePrefix=S


;[Layouts]
;ReprintHeader="<cr>            *** REPRINT ***<cr><cr>"



</defaults>



<Reservations>

[Preferences]
ReservationHeaders=TRUE
RequireResHeader=FALSE
AutoCloseGuestRes=TRUE
GuestRequired=TRUE
ClearAtFinalize=FALSE
GlobalDateTime=FALSE
;AutoDatePop=TRUE
ResDesc=<ifnconf>RES#: <reserv_no></ifnconf> <ifconf>CNF#:<user_resno></ifconf>  Guest:
<first_name> <last_name>  Arrival:<start_date>
RequiredGuestFields=First_name,Last_name,Birth_date,Address,City,State,Zip,Area_code,Phone
,E-Mail
PrintPassOnFinalize=FALSE


[Reservations]
```

```
Enabled=1
TimeInterval=10
AllowBegin=FALSE
RequirePayment=TRUE


;[Layouts]
;CCBottom=
;CCTop=
;Invoice=
;Receipt=
;Remote1All=
;Remote1New=
;Remote2All=
;Remote2New=
;Remote3All=
;Remote3New=
;Remote4All=
;Remote4New=
;SavedSale=
;Summary=

;[Server]
;StartupListener=TRUE
;ListenerPort=4215
;Both of these are for CTI integration screen pops

[Interface]
MainButton1=SAV
MainButton2=CLR
MainButton3=TSK
MainButton4=SCH
MainButton5=TEE
MainButton6=REC
MainButton7=GST
MainButton8=TLS
MainButton9=NSW
MainButton10=REF
MainButton11=UPD
MainButton12=LOG


</Reservations>
```

See also

Troubleshooting

getmatrixinfo

Description

`getmatrixinfo` takes `<department>`, `<category>`, `<item>` and the optional `<fields>` parameters and returns a table of matrix information for a Retail module item that has been configured as a matrix item. The result set returned is ordered according to the order defined in the inventory matrix template.

Input fields

XML tag	Description
`<department>`	`<item>` department.
`<category>`	`<item>` category.
`<item>`	Item.
`<fields>`	Optional. See Note on the <fields> tag. Available columns for the `<fields>` tag include all of the columns in the `i_items` (`i_it` prefix) and `i_invent` (`i_in` prefix) tables. If the `<fields>` tag is not present, the function returns: `i_it.descrip`, `i_it.invent_id` and `i_in.quantity` for the given DCI.

Return fields

Table of matrix information for the matrix item. If the DCI passed is not a matrix item, the function returns: `ERR:Function getmatrixinfo did not return information`.

Field	Description
`invent_id`	Inventory ID.
`quantity`	Sum of all the quantities in `i_invent` and should equal the quantity on hand.
`value1`	Value for the matrix attributes for the item; primary axis of the matrix.
`value2`	Value for the matrix attributes for the item.
`label1`	Names of the attributes corresponding to `value1`.
`label2`	Names of the attributes corresponding to `value2`.

Example

Example invocation:

```
<func>getmatrixinfo</func><department>retail</department><category>menscloth</category><item>qkslvshirt</item>
```

Example return string (formatted):

invent_id	Quantity	Value1	Value2	Label1	Label2
81	6.00	Surf Print	Small	COLOR	SIZE
82	8.00	Surf Print	Medium	COLOR	SIZE
83	9.00	Surf Print	Large	COLOR	SIZE
84	8.00	Surf Print	Extra Large	COLOR	SIZE
85	6.00	Black	Small	COLOR	SIZE
86	8.00	Black	Medium	COLOR	SIZE
87	10.00	Black	Large	COLOR	SIZE
88	8.00	Black	Extra Large	COLOR	SIZE
89	6.00	Orange	Small	COLOR	SIZE
90	7.00	Orange	Medium	COLOR	SIZE
91	10.00	Orange	Large	COLOR	SIZE
92	8.00	Orange	Extra Large	COLOR	SIZE
93	6.00	Red	Small	COLOR	SIZE
94	8.00	Red	Medium	COLOR	SIZE
95	10.00	Red	Large	COLOR	SIZE
96	8.00	Red	Extra Large	COLOR	SIZE
97	6.00	Dark Blue	Small	COLOR	SIZE
98	8.00	Dark Blue	Medium	COLOR	SIZE
99	10.00	Dark Blue	Large	COLOR	SIZE
100	8.00	Dark Blue	Extra Large	COLOR	SIZE
101	6.00	Dark Green	Small	COLOR	SIZE
102	8.00	Dark Green	Medium	COLOR	SIZE
103	10.00	Dark Green	Large	COLOR	SIZE
104	8.00	Dark Green	Extra Large	COLOR	SIZE

See also

None.

getmax4saleinfo

Description

`getmax4saleinfo` returns Max4Sale information for anything that satisfies the selection criteria.

Input fields

XML tag	Description
`<dciselect>`	DCI. For example, `<dciselect>department='aaa-matt' and item = 'teetime'</dciselect>`.
`<dates>`	The dates specified must be discrete dates (no ranges).

<fields>	Optional. See Note on the <fields> tag.

Return fields

Max4Sale information for the one matching DCI for the two <dates> listed.

Example

Example invocation:

```
<func>getmax4saleinfo</func><dates>10/24/2007 10 am, 10/25/2007
8am</dates><dciselect>department='IMAX-EXHIB' and item='DPSEA1000'</dciselect>
```

Example return string:

Max4Sale information for the one matching DCI for the two <dates> listed.

See also

getmods

getmods

Description

getmods gets the applicable modifier records for a DCI. An SQL join is performed between the dci_mods and items tables in order to get this information.

<calcprice> uses the current date or the date passed in the <date> tag to calculate the price. The prices are included in the return string in the price_info field, including final price, tax and fee information. These should all be included in each <item> section when passed to the processsale function.

The call respects all restrictions (at the item level) for the modifier items, and responds to the <type_?> tags in the same manner as getitemtree. (See getitemtree.) If Max4Sale is sent as true (<max4sale>1</max4sale>), then a tag will be in the qty_rem field with the number left to sell at that <date>. Otherwise, a - will be returned in that field.

Input fields

XML tag	Description
<qty>	Optional. Quantity. Price returned is for specified quantity instead of per unit. If no <qty></qty> tags are passed, then ww.dll assumes the quantity is 1 and

	returns the price for a quantity of `1`. See How ww.dll reacts to quantity tags for more information.
`<department>`	`<item>` department.
`<category>`	`<item>` category.
`<item>`	Item.
`<pass_no>`	Optional. Pass number. If the corresponding pass template contains a special macro (e.g., `SELECTLAST()ITEMSPECIAL ("ONLINE ")`), then a discounted price is returned in `price_xml` in the `[ext]` field. The specials can be any of the following in the macro for the pass template: `selectivespecial`, `globalspecial`, `itemspecial` or `special`. If the pass is voided or expired, then the discounted price is not returned. For an example, see getitem.
`<calcprice>`	Boolean. `1` = calculate the price. `0` = don't calculate the price.
`<max4sale>`	Optional. Boolean. If Max4Sale is sent as true (`<max4sale>1</max4sale>`), then a tag is returned in the `qty_rem` field with the number left to sell at that `<date>`. Otherwise, a `-` is returned in that field.
`<calcpoints>`	Optional. Boolean. If `1`, a point cost of each item (at `qty=1`) is returned as `<pts_cost>`. Note that the `ckpts4sale` (is `Points4Sale` enforced) and `pts4saledp` (Dynamic Pricing rule to use to calculate Points4Sale points to deduct) fields from the items table must be obtained in the field list from the item record (e.g., `<fields>it.department, it.category, it.item, i.ckmax4sale, ckpts4sale, pts4saledp</fields>`). Also note that all item reservation functionality works the same for Points4Sale as it does for Max4Sale except that the total point cost is stored in the `item_res.pts4qty` field.
`<avail>`	Optional. Returns information on whether a particular item is available based on Max4Sale or Points4Sale restrictions for a certain date or date range. If the tag is absent or is specified as `<avail>0</avail>`, nothing is calculated. See Note on the <avail> tag.
`<date>`	Optional. Date/time of interest.

`<account>`	Optional. An account nickname. When used, item price information is returned based on any dynamic pricing rule associated with the account. If a user is logged into a group (which is really an account), that group name is passed to `ww.dll` when getting product prices, and any dynamic pricing rule assigned to that account is applied to the prices.
	
	Note: In order for dynamic pricing rules assigned to accounts to work correctly in E-Commerce, you must add the following settings to the `[Preferences]` section of the `Sales32c.INI` file for the web Sales Host and the salespoint where you recall these sales:
	`[Preferences]`
	`ApplyAccountRulesOnRecall=FALSE`
	`RecalculatePriceOnRecall=FALSE`
	
	Without these settings in place, you see incorrect pricing on recall of these sales.

<fields>	Optional. Fields to be returned. Note on the <fields> tag.
<type_?>	Optional. Boolean for the type_1 to type_10 fields. Any fields that need to be matched can be specified. For example, if you only want matches where the type_10 is set to true, use <type_10>1</type_10>. If you wanted only matches where all types are 0 except for type 10, specify: <type_1>0</type_1><type_2>0</type_2> <type_3>0</type_3><type_4>0</type_4> <type_5>0</type_5><type_6>0</type_6> <type_7>0</type_7><type_8>0</type_8> <type_9>0</type_9><type_10>1</type_10>
<mktgcode>	Optional. Marketing code. See ww.dll now supports specials.
<srcecode>	Optional. Source code. See ww.dll now supports specials.

Return fields

XML recordset of the matching record from the items table.

Field	Description
department	Department.
category	Category.
item	Item.
descrip	Description.
price_type	Price type.
dp_set_id	The set of Dynamic Pricing rules to apply first.
price_cols	Price columns – 1, 2, or 3.
wknd_start	Weekend start.
wknd_end	Weekend end.
rate_sched	Rate schedule.
daily_pric	Daily price.
monday_0	Default price.
monday_1	Price during special rate period 1.
monday_2	Price during special rate period 2.
monday_3	Price during special rate period 3.
monday_4	Price during special rate period 4.
monday_5	Price during special rate period 5.
tuesday_0	Default price.
tuesday_1	Price during special rate period 1.
tuesday_2	Price during special rate period 2.

tuesday_3	Price during special rate period 3.
tuesday_4	Price during special rate period 4.
tuesday_5	Price during special rate period 5.
weds_0	Default price.
weds_1	Price during special rate period 1.
weds_2	Price during special rate period 2.
weds_3	Price during special rate period 3.
weds_4	Price during special rate period 4.
weds_5	Price during special rate period 5.
thursday_0	Default price.
thursday_1	Price during special rate period 1.
thursday_2	Price during special rate period 2.
thursday_3	Price during special rate period 3.
thursday_4	Price during special rate period 4.
thursday_5	Price during special rate period 5.
friday_0	Default price.
friday_1	Price during special rate period 1.
friday_2	Price during special rate period 2.
friday_3	Price during special rate period 3.
friday_4	Price during special rate period 4.
friday_5	Price during special rate period 5.
saturday_0	Default price.
saturday_1	Price during special rate period 1.
saturday_2	Price during special rate period 2.
saturday_3	Price during special rate period 3.
saturday_4	Price during special rate period 4.
saturday_5	Price during special rate period 5.
sunday_0	Default price.
sunday_1	Price during special rate period 1.
sunday_2	Price during special rate period 2.
sunday_3	Price during special rate period 3.
sunday_4	Price during special rate period 4.
sunday_5	Price during special rate period 5.
tax_rate	Tax rate.

tax_rate_b	Tax rate is defined in defaults table.
cust_tax	Custom tax rate.
cust_tax_b	Custom tax rate is defined in defaults table.
fee_rate	Fee rate.
add_tax	Add tax.
add_tax_b	Add tax b.
time_span	Time span.
span_type	Span type.
admissions	Admissions.
split_type	Split type.
item_type	Item type.
validate	Validate.
validate2	Additional validation.
ckmax4sale	Check Max4Sale.
ckpts4sale	Check Points4Sale.
pts4saledp	DP Rule to use to calculate Points4Sale points to deduct.
novalonret	When a “forced validation” item is returned, should the salepoint make sure the pass is valid prior to returning it?
set_price	Set price.
inventory	Inventory.
invent_id	Inventory ID.
barcode	Barcode.
upc	UPC.
keycode	Keycode.
keydescrip	Key description.
mod_reqd	Modifier required?
multiline	Multiline.
help_info	Help information.
price_info	Price information. Includes final price, tax and fee information.
qty_rem	If <max4sale> is sent as true (<max4sale>1</max4sale>), then information is sent back with the number left to sell at that <datetime>. Otherwise, a - is

	returned in that field.
pts_cost	If <calcpoints> is sent as true (<calcpoints >1</calcpoints>), then information is sent back with the points cost of each item (at qty=1).
avail_info	Values returned as a result of using the <avail> tag. See Note on the <avail> tag.
price_xml	Price information. If the <pass_no> tag is sent in and the corresponding pass template contains a specials macro (e.g., SELECTLAST()ITEMSPECIAL ("ONLINE")), then a discounted price is returned in price_xml in the [ext] field. The specials can be any of the following in the macro for the pass template: selectivespecial, globalspecial, itemspecial or special. If the pass is voided or expired, then the discounted price is not returned. See main description of this getitem for an example.

Example

Example invocation:

```
<func>getmods</func><department>imax-exhib</department><category>i-deepsea3</category>
<item>dpsea1000</item>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row department='IMAX-EXHIB' category='MODIFIERS '
             item='ADULT     ' descrip='Adult Entrance          '
             price_type='0' dp_set_id='0' price_cols='0' wknd_start='0'
             wknd_end='0' rate_sched='0' daily_pric='0' monday_0='0'
             monday_1='0' monday_2='0' monday_3='0' monday_4='0' monday_5='0'
             tuesday_0='0' tuesday_1='0' tuesday_2='0' tuesday_3='0'
             tuesday_4='0' tuesday_5='0' weds_0='0' weds_1='0' weds_2='0'
             weds_3='0' weds_4='0' weds_5='0' thursday_0='0' thursday_1='0'
             thursday_2='0' thursday_3='0' thursday_4='0' thursday_5='0'
             friday_0='0' friday_1='0' friday_2='0' friday_3='0' friday_4='0'
             friday_5='0' saturday_0='0' saturday_1='0' saturday_2='0'
             saturday_3='0' saturday_4='0' saturday_5='0' sunday_0='0'
             sunday_1='0' sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0'
             tax_rate='0' tax_rate_b='0' cust_tax='.0000' cust_tax_b='.0000'
             fee_rate='0' add_tax='False' add_tax_b='False' time_span='0'
             span_type='1' admissions='.00' split_type='0' item_type='1'
             validate='1' validate2='False' ckmax4sale='False' ckpts4sale='False'
             pts4saledp='0' novalonret='False' set_price='False' inventory='False'
             invent_id='0' barcode='' upc='' keycode='0' keydescrip=''
             mod_reqd='0' multiline='False' help_info=''/>
```

```
<z:row department='IMAX-EXHIB' category='MODIFIERS '
       item='CHILD     ' descrip='Child Entrance          '
       price_type='0' dp_set_id='0' price_cols='0' wknd_start='0'
       wknd_end='0' rate_sched='0' daily_pric='0' monday_0='0'
       monday_1='0' monday_2='0' monday_3='0' monday_4='0' monday_5='0'
       tuesday_0='0' tuesday_1='0' tuesday_2='0' tuesday_3='0'
       tuesday_4='0' tuesday_5='0' weds_0='0' weds_1='0' weds_2='0'
       weds_3='0' weds_4='0' weds_5='0' thursday_0='0' thursday_1='0'
       thursday_2='0' thursday_3='0' thursday_4='0' thursday_5='0'
       friday_0='0' friday_1='0' friday_2='0' friday_3='0' friday_4='0'
       friday_5='0' saturday_0='0' saturday_1='0' saturday_2='0'
       saturday_3='0' saturday_4='0' saturday_5='0' sunday_0='0'
       sunday_1='0' sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0'
       tax_rate='0' tax_rate_b='0' cust_tax='.0000' cust_tax_b='.0000'
       fee_rate='0' add_tax='False' add_tax_b='False' time_span='0'
       span_type='1' admissions='.00' split_type='0' item_type='1'
       validate='1' validate2='False' ckmax4sale='False' ckpts4sale='False'
       pts4saledp='0' novalonret='False' set_price='False' inventory='False'
       invent_id='0' barcode='' upc='' keycode='0' keydescrip=''
       mod_reqd='0' multiline='False' help_info=''/>
<z:row department='IMAX-EXHIB' category='MODIFIERS '
       item='MEMBERDISC' descrip='Member Disc. Entrance    '
       price_type='0' dp_set_id='0' price_cols='0' wknd_start='0'
       wknd_end='0' rate_sched='0' daily_pric='0' monday_0='0'
       monday_1='0' monday_2='0' monday_3='0' monday_4='0' monday_5='0'
       tuesday_0='0' tuesday_1='0' tuesday_2='0' tuesday_3='0'
       tuesday_4='0' tuesday_5='0' weds_0='0' weds_1='0' weds_2='0'
       weds_3='0' weds_4='0' weds_5='0' thursday_0='0' thursday_1='0'
       thursday_2='0' thursday_3='0' thursday_4='0' thursday_5='0'
       friday_0='0' friday_1='0' friday_2='0' friday_3='0' friday_4='0'
       friday_5='0' saturday_0='0' saturday_1='0' saturday_2='0'
       saturday_3='0' saturday_4='0' saturday_5='0' sunday_0='0'
       sunday_1='0' sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0'
       tax_rate='0' tax_rate_b='0' cust_tax='.0000' cust_tax_b='.0000'
       fee_rate='0' add_tax='False' add_tax_b='False' time_span='0'
       span_type='1' admissions='.00' split_type='0' item_type='1'
       validate='2' validate2='False' ckmax4sale='False' ckpts4sale='False'
       pts4saledp='0' novalonret='False' set_price='False' inventory='False'
       invent_id='0' barcode='' upc='' keycode='0' keydescrip=''
       mod_reqd='0' multiline='False' help_info=''/>
<z:row department='IMAX-EXHIB' category='MODIFIERS '
       item='MEMBERFREE' descrip='Member Free Entrance    '
       price_type='0' dp_set_id='0' price_cols='0' wknd_start='0'
       wknd_end='0' rate_sched='0' daily_pric='0' monday_0='0'
       monday_1='0' monday_2='0' monday_3='0' monday_4='0' monday_5='0'
       tuesday_0='0' tuesday_1='0' tuesday_2='0' tuesday_3='0'
       tuesday_4='0' tuesday_5='0' weds_0='0' weds_1='0' weds_2='0'
       weds_3='0' weds_4='0' weds_5='0' thursday_0='0' thursday_1='0'
       thursday_2='0' thursday_3='0' thursday_4='0' thursday_5='0'
       friday_0='0' friday_1='0' friday_2='0' friday_3='0' friday_4='0'
       friday_5='0' saturday_0='0' saturday_1='0' saturday_2='0'
       saturday_3='0' saturday_4='0' saturday_5='0' sunday_0='0'
       sunday_1='0' sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0'
       tax_rate='0' tax_rate_b='0' cust_tax='.0000' cust_tax_b='.0000'
       fee_rate='0' add_tax='False' add_tax_b='False' time_span='0'
       span_type='1' admissions='.00' split_type='0' item_type='1'
       validate='2' validate2='False' ckmax4sale='False' ckpts4sale='False'
       pts4saledp='0' novalonret='False' set_price='False' inventory='False'
       invent_id='0' barcode='' upc='' keycode='0' keydescrip=''
       mod_reqd='0' multiline='False' help_info=''/>
<z:row department='IMAX-EXHIB' category='MODIFIERS '
       item='SENIOR    ' descrip='Senior Entrance         '
       price_type='0' dp_set_id='0' price_cols='0' wknd_start='0'
       wknd_end='0' rate_sched='0' daily_pric='0' monday_0='0'
       monday_1='0' monday_2='0' monday_3='0' monday_4='0' monday_5='0'
       tuesday_0='0' tuesday_1='0' tuesday_2='0' tuesday_3='0'
       tuesday_4='0' tuesday_5='0' weds_0='0' weds_1='0' weds_2='0'
       weds_3='0' weds_4='0' weds_5='0' thursday_0='0' thursday_1='0'
       thursday_2='0' thursday_3='0' thursday_4='0' thursday_5='0'
```

```
            friday_0='0' friday_1='0' friday_2='0' friday_3='0' friday_4='0'
            friday_5='0' saturday_0='0' saturday_1='0' saturday_2='0'
            saturday_3='0' saturday_4='0' saturday_5='0' sunday_0='0'
            sunday_1='0' sunday_2='0' sunday_3='0' sunday_4='0' sunday_5='0'
            tax_rate='0' tax_rate_b='0' cust_tax='.0000' cust_tax_b='.0000'
            fee_rate='0' add_tax='False' add_tax_b='False' time_span='0'
            span_type='1' admissions='.00' split_type='0' item_type='1'
            validate='1' validate2='False' ckmax4sale='False' ckpts4sale='False'
            pts4saledp='0' novalonret='False' set_price='False' inventory='False'
            invent_id='0' barcode='' upc='' keycode='0' keydescrip=''
            mod_reqd='0' multiline='False' help_info=''/>
</rs:data>
```

See also

getitem
getitemexpanded
getitemtree
getmods
processsale

getparents

Description

`getparents` behaves in a similar manner to `getmods`. `getparents` gets items that are allowed as parents to the list of modifiers given in the `<hasmod>` tag. Pricing information, point cost and Max4Sale/Points4Sale availability can also be returned.

Input fields

XML tag	Description
`<qty>`	Optional. Quantity. Price returned is for specified quantity instead of per unit. If no `<qty></qty>` tags are passed, then `ww.dll` assumes the quantity is `1` and returns the price for a quantity of `1`. See How ww.dll reacts to quantity tags for more information.
`<hasmod>`	Modifiers.
`<calcprice>`	Boolean (`1` = calculate the price, `0` = don't calculate the price)
`<max4sale>`	Boolean. (`1` = return Max4Sale information, `0` = don't return Max4Sale information)

<calcpoints>	Optional. Boolean. If 1, a point cost of each item (at qty=1) is returned as <pts_cost>. Note that the ckpts4sale (is Points4Sale enforced) and pts4saledp (Dynamic Pricing rule to use to calculate Points4Sale points to deduct) fields from the items table must be obtained in the field list from the item record (e.g., <fields>it.department, it.category, it.item, i.ckmax4sale, ckpts4sale, pts4saledp</fields>). Also note that all item reservation functionality works the same for Points4Sale as it does for Max4Sale except that the total point cost is stored in the item_res.pts4qty field.
<datetime>	Optional. Date/time of interest.
<calcpoints>	Boolean. (1 = return points cost information, 0 = don't return points cost information)

Return fields

XML recordset of matching records from the items table.

Example

Example invocation:

```
<func>getparents</func><hasmod>imax modifiers adult imax modifiers child
imax modifiers youth imax modifiers senior
</hasmod><max4sale>1</max4sale><calcprice>1</calcprice><calcpoints>1</calcp
oints><datetime>2008-02-05</datetime>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row department='IMAX ' category='MOVIES '.......
```

See also

getmods

getpassinfo

Description

getpassinfo retrieves information about new pass/access records and modified pass/access records. getpassinfo offers the ability to query the SiriusSQL data in order to retrieve information regarding new pass sales/access records and modified pass/access records.

Note: Either <new>1</new> (for new pass/access info) or <mod>1</mod> (for modified pass/access info) must be specified in the function call.

Input fields

XML tag	Description
<new>	Return new records since the indicated datetime. <new>1</new> (for new pass/access info) or <mod>1</mod> (for modified pass/access info) must be specified in the function call. Boolean.
<mod>	Return modified records since the indicated datetime. <new>1</new> (for new pass/access info) or <mod>1</mod> (for modified pass/access info) must be specified in the function call. Boolean.
<datetime>	Sets the datetime.
<access>	Used to check the access table instead of the gst_pass table, default is to check gst_pass. Boolean.

Return fields

Selected fields from records in either the gst_pass or access table.

Example

Example invocation:

Returns all new passes purchased since 6/04/2007:

```
<func>getpassinfo</func><new>1</new><mod>0</mod><datetime>6/04
/2007</datetime><access>0</access>
```

Returns all new passes purchased since 6/06/2007:

```
<func>getpassinfo</func><new>1</new><mod>0</mod>
<datetime>6/06/2007</datetime><access>0</access>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row pass_no='5020001' guest_no='1020001' masterpass='5020001'
             swipe_no='' addit_no='0' val_parent='False' valprnttyp='1'
             printcount='0' start_date='2006-11-01T00:00:00' expires='2007-04-30T00:00:00'
             validcount='0' dis_count='0' total_uses='1' usesw_left='0'
             usest_left='0' points1='0' points2='0' money1='0' money2='0'
             warnings='0' voided_for='' voided_by='' trans_no='37020001'
```

```
            mastertran='37020001' department='PASSES    ' category='LIMITED   '
            item='WEEKDAY   ' amt_paid='200' account='BSCOUTS436'
            operator='SUZY  ' salespoint='RESERV' date_time='2007-09-30T14:25:26'
            totalcomp='0' invoice_no='0' cc_tracks='' crlimit='0'
            crlimit_dy='0' dw_active='False' splimit='0' splimit_dy='0'
            last_mod='74' card_id='0' bl_reason='          ' importpass='SPAS5020001'
            level_chg='0' purch_chg='0'/>
</rs:data>
```

See also

getaccessinfo

getuniquekey

Description

`getuniquekey` takes a `<key>` tag with a field name from the `sequence` table. Using `getuniquekey`, you get a new `address_id` of, for example, `19000010`, with which you can call a function like `insert`:

```
<func>insert</func><params>into addlink (addlink_id, guest_no, address_id) VALUES
(19000010, 1000000, 1000000)</params>
```

Input fields

XML tag	Description
`<key>`	Field name from the `sequence` table. For example, `k_address`, `k_guests`, `k_addlink`, `ww_sale`, `k_transact`, and `k_sale_hdr`.

Return fields

A unique key returned within an XML tag that corresponds to the field name passed to the call.

Example

Example invocation:

```
<func>getuniquekey</func><key>k_address</key>
```

Example return string:

```
OK :<k_address>20000010</k_address>
```

See also

insert
select
update

getwwsaleid

Description

`getwwsaleid` retrieves a new `wwsale_id` for use in tracking a sale. This is required if you are using real time inventory tracking.

Input fields

None.

Return fields

XML tag	Description
`<wwsale_id>`	Web sale ID.

Example

Example invocation:

```
<func>getwwsaleid</func>
```

Example return string:

```
OK : <wwsale_id>1000000<wwsale_id>
```

See also

None.

insert

Description

`insert` constructs a SQL `INSERT` statement using whatever is contained in `params`.

Note: In order to ensure compatibility with other Salesware applications, you should not use this call. Compatibility is ensured only if you use the pre-defined calls that write data to the SiriusSQL database.

Input fields

XML tag	Description

`params`	`INSERT` SQL statement, minus the `INSERT`. Can also use two functions:
	`**SUB(operator)`
	`**SUB(salespoint)`
	in order to populate the `operator` and `salespoint` fields.

Return fields

Status only.

Example

Example invocation:

```
<func>insert</func><params>into addlink (addlink_id, guest_no, address_id) VALUES
(19000010, 1000000, 1000000)</params>
```

Example return string:

```
OK :
```

See also

getuniquekey
select
update

lookupguests

Description

`lookupguests` looks up guests in the database based on the fields passed. All fields are optional, but you need to use at least one in order to get any results. All the fields are used to filter the `guests` table, with the exception of `<fields>`, which can contain a different list of fields. In the case of the `<fields>`, tag, fields should be identified using `g` for the `guests` table, `a` for `address` table, and `l` for the `addlink` table, as shown in the following example:

```
<func>lookupguests</func><first_name>matt</first_name><fields>g.guest_no, g.first_name,
a.address</fields>
```

If the `<fields>` tag is not passed, the default is returned, as described below.

Note: If there are multiple address matches, a record is returned for each match. The `area_code` and `phone` are used to search both phone numbers of a particular address.

Note: When a photo is returned, the format is an encoded binary string that must be decoded to produce a JPEG.

`lookupguests` also supports lookup by second guests, and the `<crit>` tag. For example:

```
<func>lookupguests</func><firstname2>marge</firstname2><lastname2>simpson</lastname2>
```

returns the following:

```
<rs:data><z:row guest_no='5001000' first_name='HOMER' last_name='SIMPSON'
birth_date='1960-01-01T00:00:00' address_id='3001000' area_code='(505)' phone='751-0633'
area_cod2='( )'
phone2='')) 'address='1337 E GUSDORF ROAD' city='TAOS' state='NM '
zip='87571'/><z:row guest_no='5001000' first_name='HOMER' last_name='SIMPSON'
birth_date='1960-01-01T00:00:00' address_id='736000000'area_code='505 ' phone='751-0633'
area_cod2=' ' phone2=' ' address='1337 E GUSDORF ROAD' city='TAOS' state='NM '
zip='87571'/><z:row guest_no='655000000' first_name='HOMER' last_name='SIMPSON'
birth_date='1954-01-01T00:00:00' address_id='165000000' area_code='(417)' phone='856-1234'
area_cod2=' ' phone2=' ' address='1234 Evergreen Terrace' city='Springfield' state='MO '
zip='65807'/></rs:data>
```

But the same call with an added `<crit></crit>` field,

```
<func>lookupguests</func><firstname2>marge</firstname2><lastname2>simpson</lastname2><crit
><e_mail>homer@siriusware.com</e_mail></crit>
```

returns the following:

```
<rs:data><z:row guest_no='655000000' first_name='HOMER'last_name='SIMPSON'
birth_date='1954-01-01T00:00:00' address_id='165000000'area_code='(417)' phone='856-1234'
area_cod2=' ' phone2=' ' address='1234 Evergreen Terrace' city='Springfield' state='MO '
zip='65807'/></rs:data>
```

Note: The`<crit>` functionality gets parsed and added to the statement without much processing or checking. If you pass `<crit><badfield>1</badfield></crit>`, `badfield='1'` is added to the SQL statement, which fails.

Input fields

XML tag	Description

<first_name>	Optional. Guest first name.
<last_name>	Optional. Guest last name.
<firstname2>	Optional. Used to look up guests by the second guest in the guest record.
<lastname2>	Optional. Used to look up guests by the second guest in the guest record.
<crit>	Optional. Used to pass in additional fields from the `guests` and/or `address` tables.
<birth_date>	Optional. Guest birth date. Must be in *YYYY-mm-dd* format.
<address>	Optional. Guest address line 1.
<address2>	Optional. Guest address line 2.
<city>	Optional. Guest city.
<state>	Optional. Guest state.
<zip>	Optional. Guest zip.
<area_code>	Optional. Guest area code. Must be in `(`*xxx*`)` format – e.g., `(303)`.
<fields>	Optional. Other fields to be returned. Note on the <fields> tag.
<getphoto>	Optional. Boolean indicating whether `lookupguests` should return a photo.

Return fields

The fields shown in the table are returned by default.

Field	Description
guest_no	Guest number.
first_name	Guest first name.
last_name	Guest last name.
birth_date	Guest birth date.
address_id	Guest address ID.
area_code	Guest area code.
phone	Guest phone.
area_cod2	Guest area code 2.
phone2	Guest phone 2.
address	Guest address.
city	Guest city.
state	Guest state.
zip	Guest zip.

Example

Example invocation:

```
<func>lookupguests</func><first_name>matt</first_name>
<last_name>messinger</last_name>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row guest_no='14000001' first_name='MATT' last_name='MESSINGER'
             birth_date='1970-01-01T00:00:00' address_id='1000001'
             area_code='(505)' phone='751-0633' area_cod2='(   )' phone2='   -    '
             address='122 N MAIN STREET TEST' city='GUNNISON' state='CO '
             zip='81230'/>
</rs:data>
```

See also

getguest

lookuppasses

Description

`lookuppasses` looks up passes given a particular `guest_no`.

Input fields

XML tag	Description
`<pass_no>`	Pass number. Either a pass number, guest number, swipe number or additional number must be passed, at a minimum.
`<guest_no>`	Guest number. Either a pass number, guest number, swipe number or additional number must be passed, at a minimum.
`<swipe_no>`	Swipe number. For example, `<func>lookuppasses</func><swipe_no>6035241234569040 </swipe_no>` Either a pass number, guest number, swipe number or additional number must be passed, at a minimum.
`<addit_no>`	Additional number. Usually refers to number printed on pass or ticket stock. Either a pass number, guest number, swipe number or additional number must be passed, at a minimum.
`<date_time>`	Optional. Date/time of interest.

<check_voids>	Optional. Can be passed (as 1) and voided passes are excluded. Boolean.
<fields>	Optional. Can be used to specify only certain fields to return using the aliases t for template and p for gst_pass. Note on the <fields> tag.

Return fields

Record set in XML format of gst_pass and template (all fields by default). If the date_time field is passed, then the start_time and expires fields are checked (to eliminate expired passes).

Note: Field names such as c60 that do not appear in the data dictionary are automatically generated when there are duplicate field names in joined datasets. These field names are not defined in the following table because their meanings vary. The description in the data set tells exactly what the duplicate field is. For example:

```
<s:AttributeType name='c52' rs:name='department' rs:number='53' rs:nullable='true'
                 rs:writeunknown='true'>
          <s:datatype dt:type='string' rs:dbtype='str' dt:maxLength='10'
                 rs:fixedlength='true'/>
</s:AttributeType>
```

Field	Description
pass_no	Pass number.
guest_no	Guest number.
masterpass	Master pass.
swipe_no	Swipe number.
addit_no	Additonal number.
val_parent	Validate parent record.
valprnttyp	Validate parent type.
last_use	Last use.
printcount	Number of times printed.
start_date	Start date.
expires	Expiration date.
validcount	Number of times validated.
dis_count	Like validcount, but can be reset.
total_uses	Total uses.
usesw_left	Uses left in current week.
week_refr	Used to determine when a week has passed.
usest_left	Uses left in current day.
day_refr	Used to determine when a day has passed.
points1	User-defined meaning.

points2	User-defined meaning.
money1	User-defined meaning.
money2	User-defined meaning.
blackout_s	Blackout start date.
blackout_e	Blackout end date.
warnings	User-defined meaning.
voided_for	Reason for void.
voided_by	Who voided.
shift_ends	Next time pass is validated.
trans_no	Transaction number.
mastertran	Master transaction.
department	Department.
category	Category.
item	Item.
amt_paid	Amount paid.
account	Account.
operator	Operator.
salespoint	Salespoint.
date_time	Date/time.
totalcomp	Total # of comps given to a guest who forgot his pass.
invoice_no	Invoice number.
cc_tracks	Hold some or all of the track1, 2 and 3 magnetic swipe info from card.
crlimit	Credit limit.
crlimit_dy	Credit limit per day.
dw_active	Indication of whether In-House Cards card is active.
splimit	Total spending limit for this In-House Cards card.
splimit_dy	Daily spending limit for this In-House Cards card.
last_mod	Last modified.
card_id	Card ID.
bl_reason	Blackout reason.
importpass	Import pass ID (for funding interface)
level_chg	Level change (for funding interface)
purch_chg	Purchase change to pass (for funding interface)

layout	Layout.
pref_prntr	Preferred printer.
one_access	Used internally by Sales.
t_wk_use	Uses allowed per week.
t_day_use	Uses allowed per day.
t_bl_out1s	Start of pass blackout period 1.
t_bl_out1e	End of pass blackout period 1.
t_bl_out2s	Start of pass blackout period 2.
t_bl_out2e	End of pass blackout period 2.
t_bl_out3s	Start of pass blackout period 3.
t_bl_out3e	End of pass blackout period 3.
t_bl_out4s	Start of pass blackout period 4.
t_bl_out4e	End of pass blackout period 4.
t_mon	Can pass be used on Mondays?
t_tue	Can pass be used on Tuesdays?
t_wed	Can pass be used on Wednesdays?
t_thu	Can pass be used on Thursdays?
t_fri	Can pass be used on Fridays?
t_sat	Can pass be used on Saturdays?
t_sun	Can pass be used on Sundays?
t_assignno	Get assignment number.
t_startime	Item is valid only between t_startime and t_endtime in a day.
t_endtime	Item is valid only between t_startime and t_endtime in a day.
t_ck_time	Enforce validation of the time portion of the start_date and expires fields?
t_asnotype	Assign number type.
t_newdnact	Create a new DirectNet account (in Sales) when this pass is created?
t_uvalue1	Are points1/money1 to be used for a unit value program?
t_uvalue2	Are points2/money2 to be used for a unit value program?
v_tot_use	What to do to total uses counter in pass when validated.

v_wk_use	What to do to uses per week counter in pass when validated.
v_day_use	What to do to uses per day counter in pass when validated.
v_points1	What to do to points1 counter in pass when validated.
v_points2	What to do to points2 counter in pass when validated.
v_money1	What to do to money1 counter in pass when validated.
v_money2	What to do to money2 counter in pass when validated.
v_delay	How much time should go by before the pass can be used again?
v_no_dscnt	Do not increment the discount counter when processing the validation.
a_autodep1	Department to use for automated sale in default time period.
a_autocat1	Category to use for automated sale in default time period.
a_autoitm1	Item to use for automated sale in default time period.
a_auto2	Time that second “auto” dept takes effect.
a_autodep2	Department to use for automated sale in second alternate time period.
a_autocat2	Category to use for automated sale in second alternate time period.
a_autoitm2	Item to use for automated sale in first second time period.
a_auto3	Time that third “auto” dept takes effect.
a_autodep3	Department to use for automated sale in third alternate time period.
a_autocat3	Category to use for automated sale in third alternate time period.
a_autoitm3	Item to use for automated sale in first third time period.
a_auto4	Time that fourth “auto” dept takes effect.
a_autodep4	Department to use for automated sale in fourth alternate time period.
a_autocat4	Category to use for automated sale in fourth alternate time period.
a_autoitm4	Item to use for automated sale in first fourth time period.
a_discnt1	When pass “discount” counter hits this level, use

	v_d1_flat and v_d1_pct to automatically discount auto sale item.
a_d1_flat	See a_discnt1.
a_d1_pct	See a_discnt1.
a_discnt2	When pass “discount” counter hits this level, use v_d2_flat and v_d2_pct to automatically discount auto sale item.
a_d2_flat	See a_discnt2.
a_d2_pct	See a_discnt2.
a_discnt3	When pass “discount” counter hits this level, use v_d3_flat and v_d3_pct to automatically discount auto sale item.
a_d3_flat	See a_discnt3.
a_d3_pct	See a_discnt3.
a_discnt4	When pass “discount” counter hits this level, use v_d4_flat and v_d4_pct to automatically discount auto sale item.
a_d4_flat	See a_discnt4.
a_d4_pct	See a_discnt4.
a_discnt5	When pass “discount” counter hits this level, use v_d5_flat and v_d5_pct to automatically discount auto sale item.
a_d5_flat	See a_discnt5.
a_d5_pct	See a_discnt5.
a_resetcnt	Reset discount counter in pass record to 0 after hitting discount level.
a_resetamt	Can be used to subtract a certain number from discount count instead of setting it to 0.
a_shiftype	Validation shift type.
a_sh_tot	If *not* first used in shift, change total uses counter in pass?
a_sh_wk	If *not* first used in shift, change total uses counter in pass?
a_sh_day	If *not* first used in shift, change uses per day counter in pass?
a_sh_pts1	If *not* first used in shift, change points1 counter in pass?
a_sh_pts2	If *not* first used in shift, change points2 counter in pass?

a_sh_mny1	If *not* first used in shift, change money1 counter in pass?
a_sh_mny2	If *not* first used in shift, change money2 counter in pass?
a_sh_disc	If *not* first used in shift, change discount in pass?
a_sh_tran	If *not* first used in shift, do automated sale?
a_sh_log	If *not* first used in shift, generate use_log record?
a_flexdate	Flexible date.
a_no_loc_s	Start of range for locations not accessible by this type of pass.
a_no_loc_e	End of range for locations not accessible by this type of pass.
a_no_loc_l	Comma delimited list of locations not accessible by this type of pass.
a_timespan	Used to calculate expires field if flex scanning is used to set start_date and expires upon first scan.
a_autofinl	When doing an auto-validation at a point of sale, should the auto-sale finalize itself, or wait for user input.
cc_pid	Used to create valid credit/debit card numbers (DirectNet).
levelofval	Level of validation to be performed
do_on_val	String of SalesEZ commands that can be run at the salespoint on validaton.
dw	Is debitware allowed for this pass?
dw_crl	Debitware default amount of credit limit.
dw_crlf	Debitware credit limit cannot be modified at salespoint when activated.
dw_spl	Debitware total spending limit.
dw_splf	Debitware spending limit cannot be modified at salespoint when activated.
dw_spld	Debitware daily spending limit.
dw_spldf	Debitware daily spending limit cannot be modified at salespoint when activated.
dw_acc	Debitware name of account where invoice is to be created.
dw_prel	Debitware amount that will automatically get put onto the debitware invoice.

dw_ignrexp	Ignore expiration dates on Debitware cards.
log_chkpt	When validating, create a log entry in gst_ckpt table as well.
trklnoauto	Do not autosale if pass is invalid when trickle validating.
resortchrg	Is this pass product a resort charge capable product?
prescedenc	When validating passes, pass types can be ranked to determine which to validate first.

Example

Example invocation:

```
<func>lookuppasses</func><guest_no>26000001</guest_no><check_voids>1</check_voids>
```

Example return string:

```
OK :…recordset
<rs:data>
      <z:row pass_no='5000001' guest_no='26000001' masterpass='0'
             swipe_no='' addit_no='0' val_parent='False' valprnttyp='1'
             printcount='0' start_date='2006-11-01T00:00:00' expires='2008-04-
             30T00:00:00'
             validcount='0' dis_count='0' total_uses='0' usesw_left='0'
             usest_left='0' points1='2' points2='0' money1='0' money2='0'
             warnings='0' voided_for='' voided_by='' trans_no='0' mastertran='0'
             department='PASSES' category='EMPLOYEE' item='EMPLOYEE'
             amt_paid='399.95' account='' operator='WEBOP1' salespoint='WEBSP '
             date_time='2007-10-18T10:57:31' totalcomp='0' invoice_no='0'
             cc_tracks='' crlimit='0' crlimit_dy='0' dw_active='False'
             splimit='0' splimit_dy='0' last_mod='75' card_id='0' bl_reason='          '
             importpass='SPAS5000001' level_chg='0' purch_chg='0' c52='PASSES    '
             c53='EMPLOYEE  ' c54='EMPLOYEE  ' layout='CURRENTPASSEMP.FRX          '
             pref_prntr='' c57='False' c58='1' one_access='False' c60='2006-11-
             01T00:00:00'
             c61='2008-04-30T00:00:00' c62='0' c63='3' c64='4' c65='0'
             c66='0' c67='0' t_wk_use='0' t_day_use='0' t_mon='True'
             t_tue='True' t_wed='True' t_thu='True' t_fri='True' t_sat='True'
             t_sun='True' t_assignno='True' t_startime='     ' t_endtime='     '
             t_ck_time='False' t_asnotype='0' t_newdnact='False' t_uvalue1='False'
             t_uvalue2='False' v_tot_use='-1' v_wk_use='0' v_day_use='0'
             v_points1='0' v_points2='0' v_money1='0' v_money2='0'
             v_delay='0' v_no_dscnt='False' a_autodep1='          '
             a_autocat1='          ' a_autoitm1='          ' a_auto2='     '
             a_autodep2='' a_autocat2='          ' a_autoitm2='          '
             a_auto3='     ' a_autodep3='' a_autocat3='          '
             a_autoitm3='          ' a_auto4='     ' a_autodep4=''
             a_autocat4='          ' a_autoitm4='          ' a_discnt1='0'
             a_d1_flat='0' a_d1_pct='.00' a_discnt2='0' a_d2_flat='0'
             a_d2_pct='.00' a_discnt3='0' a_d3_flat='0' a_d3_pct='.00'
             a_discnt4='0' a_d4_flat='0' a_d4_pct='.00' a_discnt5='0'
             a_d5_flat='0' a_d5_pct='.00' a_resetcnt='1' a_resetamt='0'
             a_shiftype='1' a_sh_tot='False' a_sh_wk='False' a_sh_day='False'
             a_sh_pts1='False' a_sh_pts2='False' a_sh_mny1='False'
             a_sh_mny2='False' a_sh_disc='False' a_sh_tran='False'
             a_sh_log='True' a_flexdate='False' a_no_loc_s='0' a_no_loc_e='0'
             a_no_loc_l='' a_timespan='0' a_autofinl='False' cc_pid='   '
```

```
            levelofval='8' do_on_val='IGNOREERRORS()SELECTIVESPECIAL(&#x22;EMPRETAIL
            &#x22;)IGNOREERRORS()SELECTIVESPECIAL(&#x22;EMPFOOD   &#x22;)'
            dw='True' dw_crl='0' dw_crlf='True' dw_spl='0' dw_splf='True'
            dw_spld='0' dw_spldf='True' dw_acc='D-EMPPURCH' dw_prel='0'
            dw_ignrexp='False' log_chkpt='False' trklnoauto='False'
            c166='451' resortchrg='False' prescedenc='63'/>
</rs:data>
```

modifyaccess

Description

`modifyaccess` is provided for convenience. It simply calls `modifypass` with `<access>1</access>` specified.

Input fields

XML tag	Description
`<pass_no>`	Pass number.
`<other field>`	Any field from the `gst_pass` or `access` table.

Return fields

Status only.

Example

Example invocation:

```
<func>modifyaccess</func><pass_no>5000001</pass_no>
<voided_for>THEFT</voided_for><voided_by>MATT</voided_by>
```

Example return string:

```
OK :
```

See also

`modifypass`

modifyaddress

Description

`modifyaddress` modifies existing guest information.

Input fields

XML tag	Description
`<address_id>`	Guest address ID.
`<other field>`	Any other field that exists in the `address` table – e.g., `<city>`. The contents of this tag are written into the corresponding field.

Return fields

Status only.

Example

Example invocation:

```
<func>modifyaddress</func><address_id>16099000</address_id>
<city>Gunnison</city>
```

Example return string:

```
OK :Function modifyaddress did not return information
```

See also

`newaddress`
`newaddlink`

modifygsrent

Description

`modifygsrent` updates information for the guest for the rental process. Use with the Rentals module only.

Input fields

XML tag	Description
`<guest_no>`	Guest number.
`<type_skier>`	Type of skier. See the *Salesware Rentals* document for more information.

<shoesize>	Shoe size. See the *Salesware Rentals* document for more information.

Return fields

Status only. Confirmation of changes.

XML tag	Description
<type_skier>	Type of skier.
<shoesize>	Shoe size.

Example

Example invocation:

```
<func>ModifyGstRent</func><guest_no>26000001</guest_no><type_skier>3</type_skier><shoesize
>10.5</shoesize><type_skier>3</type_skier><shoesize>10.5</shoesize>
```

Example return string:

```
OK :<shoesize>10.5</shoesize><type_skier>3</type_skier>
```

See also

None.

modifyguest

Description

`modifyguest` modifies information for an existing guest.

Accepts `password_pt`, which is a plain text password that is correctly hashed and added to the passwords field in the `guests` table. The functions `newguest` and `modifyguest` also accept `cc_swipe_pt` and `cc_number_pt`. These accept plain text which is correctly hashed and added to the `cc_swipe` and `cc_number` fields, respectively, in the `guests` table.

Input fields

XML tag	Description

<guest_no>	Guest number.
<*other field*>	Any other field that exists in the guests table – e.g., <parent_no>. The contents of this tag are written into the corresponding field.

Return fields

Status only.

Example

Example invocation:

```
<func>modifyguest</func><guest_no>26000001</guest_no>
<parent_no>16099001</parent_no>
```

Example return string:

```
OK :Function modifyguest did not return information
```

See also

None.

modifypass

Description

modifypass adds the ability to modify existing pass/access records to accommodate voids, returns, updated valid and blackout dates, and updated usage. It takes <pass_no> and the following optional parameters: any field from the gst_pass or access table, and <access>1</access> if this is to modify an access record.

Input fields

XML tag	Description
<pass_no>	Pass number.
<*other field*>	Any field from the gst_pass or access table.
<access>	Optional. If true (1), modify access record. Boolean.

Return fields

Status only.

Example

Example invocation:

This call adds the specified blackout date to the specified pass record:

```
<func>modifypass</func><pass_no>5000001</pass_no><blackout_s>10/24/2007 12:00:00
AM</blackout_s><blackout_e>10/24/2007 11:59:50 PM</blackout_e>
```

This call voids the specified access record:

```
<func>modifypass</func><pass_no>5000001</pass_no>
<voided_for>THEFT</voided_for><voided_by>MATT</voided_by>
<access>1</access>
```

Example return string:

```
OK :
```

See also

modifyaccess

newaddlink

Description

`newaddlink` creates a new `addlink` record. The `addlink` table is needed because there is a many-to-many relationship between guests and addresses (a single guest can have many addresses, or a single address can be associated with many guests). Passing a `guest_no` and an `address_id` is required by the database.

Input fields

XML tag	Description
<guest_no>	Guest number.
<address_id>	Address ID.
<other field>	Optional. Any other field that exists in the `addlink` table – e.g., <city>. The contents of this tag are written into the corresponding field.

Return fields

XML tag	Description
<addlink_id>	Address link ID.

Example

Example invocation:

```
<func>newaddlink</func><address_id>3024001</address_id><guest_no>26000001</guest_no>
```

Example return string:

```
OK :<addlink_id>29000001</addlink_id><address_id>3024001</address_id>
<guest_no>26000001</guest_no>
```

See also

modifyaddress
newaddress

newaddress

Description

newaddress creates a new record in the address table. Passing a guest_no is required by the database. Any fields specified in the DefaultInfo setting in the registry (added to the registry using the ww.reg file – see the *Salesware E-Commerce* document) are included in what's sent to the server. These fields can be overridden by passing them into this function.

Input fields

XML tag	Description
<guest_no>	Guest number.
<other field>	Optional. Any other field that exists in the address table – e.g., <city>. The contents of this tag are written into the corresponding field.

Return fields

XML tag	Description
<address_id>	Address ID of the new record.

Example

Example invocation:

```
<func>newaddress</func><address>110 S. Lashley Lane</address>
<guest_no>2600000</guest_no>
```

Example return string:

```
OK :<address>110 S. Lashley
Lane</address><address_id>30000001</address_id><guest_no>2600000</guest_no><operator>WEBOP
1</operator><salespoint>WEBSP</salespoint>
```

See also

modifyaddress
newaddlink

newbooking

Description

`newbooking` adds the booking to the privates schedule by passing in `<department>`, `<category>`, `<item>`, `<guest_no>`, `<start_time>`, `<end_time>` and, optionally, `<instructor>`, `<request>` and `<max4sale>`. The following are examples. The first example uses a requested regular instructor and the second uses a TBD instructor.

```
<func>newbooking</func><guest_no>212000000</guest_no>
<department>PRIVATES</department><category>SKIING</category>
<item>2HRPRIV</item><start_time>03/12/2007 11am</start_time>
<end_time>03/12/20071 pm</end_time><instructor>SWILSON</instructor>
<request>SWILSON</request>

<func>newbooking</func><guest_no>212000000</guest_no>
<department>privates</department><category>skiing</category>
<item>2hrpriv</item><start_time>03/13/2007 9am</start_time>
<end_time>03/13/2007 11am</end_time>
<instructor>TBDALL01</instructor>
```

A `<booking_id>` is returned if the booking is added to the schedule. The booking is scheduled if it doesn't have a conflict in the instructor schedule or exceed lesson type Max4Sale limits. If `<max4sale>` is passed in with the call, then DCI Max4Sale limits are checked to verify that these limits have not been used up.

Note: Currently, if you want to book into the TBD schedule, you need to specify a TBD instructor as in the example shown above.

To submit a sale with a booking via the `processsale` function, the booking is specified using the `booking_id`. The following is an example of a portion of the `processsale` call:

```
<func>processsale</func><sale><save><guest_no>14001000</guest_no>
<first_name>SAM</first_name><last_name>SPADE</last_name>
<phone>(505) 555-2468</phone></save><settlement>CASH</settlement>
<item><dci>PRIVATES SKIING 2HRPRIV</dci><qty>1</qty><finalprice>75.00</finalprice>
<guest><renew><guest_no>14001000</guest_no></renew></guest>
<date>03-12-2007</date><do_on_sale></do_on_sale><pvt_book>
<booking_id>1409000000</booking_id></pvt_book></item></sale>
```

When the sale is processed by Sales Host, the `trans_no` is added to the booking and other pertinent information is populated into the data.

Input fields

XML tag	Description
`<department>`	`<item>` department.
`<category>`	`<item>` category.
`<item>`	Item.
`<guest_no>`	Guest number.
`<start_time>`	Start time for booking.
`<end_time>`	End time for booking.
`<instructor>`	Optional. Instructor.
`<request>`	Optional. Request for instructor.
`<max4sale>`	Optional. Max4Sale limits are checked to verify that these limits have not been used up. Boolean.

Return fields

XML tag	Description
`<booking_id>`	The ID of the successful booking.

Example

Example invocation:

```
<func>newbooking</func><guest_no>212000000</guest_no>
<department>privates</department><category>skiing</category>
<item>2hrpriv</item><start_time>03/13/2007 9am</start_time>
<end_time>03/13/2007 11am</end_time>
<instructor>TBDALL01</instructor>
```

Example return string:

```
OK :<booking_id>86000001</booking_id>
```

See also

getdciavail

newguest

Description

newguest creates a new guest record in the guest table. Any fields specified in the DefaultInfo setting in the registry (added to the registry using the ww.reg file – see the *Salesware E-Commerce* document) are included in what's sent into the server. These fields can be overridden by passing them into this function.

Accepts password_pt, which is a plain text password that is correctly hashed and added to the passwords field in the guests table. The functions newguest and modifyguest also accept cc_swipe_pt and cc_number_pt. These accept plain text which is correctly hashed and added to the cc_swipe and cc_number fields, respectively, in the guests table.

Input fields

XML tag	Description
<other field>	Optional. Any field that exists in the guests table – e.g., <first_name>. The contents of this tag are written into the corresponding field.

Return fields

XML tag	Description
<guest_no>	Number of the newly-created guest.

Example

Example invocation:

```
<func>newguest</func><first_name>NEVILLE</first_name><last_name>HARSON</last_name>
```

Example return string:

```
OK :<first_name>NEVILLE</first_name><guest_no>28000001</guest_no>
<last_name>HARSON</last_name><operator>WEBOP1</operator><parent_no>28000001</parent_no>
<salespoint>WEBSP</salespoint>
```

See also

modifyguest

newitemres

Description

`newitemres` checks Max4Sale limits for an item with real time inventory and creates a record in the `item_res` table if there is sufficient quantity available. The `item_res` table tracks real-time item reservations, primarily when a user puts an item in his cart in web sales. The table is used to give accurate Max4Sale remaining quantity calculations.

Input fields

XML tag	Description
`<department>`	`<item>` department.
`<category>`	`<item>` category.
`<item>`	Item.
`<quantity>`	Quantity reserved or placed in cart.
`<wwsale_id>`	Web sale ID. If a `wwsale_id` that is passed in the call already exists in the `ww_sales` table, `ww.dll` returns “ERR:wwsale_id is not unique.”
`<start_date>`	Start date for the `<item>`.
`<hold_time>`	Optional. In minutes. Defaults to 5 minutes. (This is how long the user has before the `processsale` call is made.)

Return fields

XML tag	Description
`<itmres_no>`	Item reservation number.

Example

Example invocation:

```
<func>newitemres</func><department>AAA-MATT
</department><category>AAA</category><item>CANDY</item><quantity>1</quantity>
<start_date>03-17-2004 07:00:00</start_date><wwsale_id>1000000</wwsale_id>
```

Example return string:

```
OK :<itmres_no>4000000</itmres_no>
```

See also

clearitemres

processsale

Description

processsale takes a description of a sale and queues it for processing asynchronously. Synchronously: approvals are obtained for credit cards Asynchronously: The sale is "read" by sales32c and credit cards are activated in the batch. For a description of the <sale> tag and an extended example, see Overview of tags in <sale>. The following is a simple example.

```
<sale><settlement><swipe>*378340467012013^1201~</swipe></settlement> <item><dci>PASSES
COPPER    SPR1      </dci><qty>1</qty>
<guest><renew><guest_no>15010000</guest_no></renew></guest></item></sale>
```

A sub-function called verifysale is called by processsale to help cut down on bad sale strings getting passed to Sales Host for processing. This function runs a variety of checks on the sale to make sure that it is OK to be processed. verifysale checks payment amounts, quantities of items in the sale, the number of modifiers and the existence of templates, as well as guest information (if required).

Input fields

XML tag	Description
<sale>	See Overview of tags in <sale>.

Return fields

XML tag	Description
<wwsale_id>	Web sale ID. If a wwsale_id that is passed already exists in the ww_sales table, ww.dll returns “ERR:wwsale_id is not unique.”

Example

Example invocation:

```
<func>processsale</func>
   <sale>
     <res_hdr><guest_no>16099000</guest_no></res_hdr>
     <save><first_name>MATT</first_name><last_name>MESSINGER</last_name></save>
     <settlement><card_no>4300000000000009</card_no><exp_date>0505</exp_date>
       <card_addr>202 Nowhere Ln.</card_addr><card_zip>02061</card_zip>
       <ship_zip>80302</ship_zip>
       <amount>40.05</amount>
       <tax>0.00</tax>
```

```
        <card_cvv2>101</card_cvv2>
      </settlement>
      <item><dci>AAA-MATT  AAA       CANDY</dci>
        <qty>1</qty><price>40.05</price><date>01/18/2004</date>
      </item>
    </sale>
```

Example return string:

```
OK: <wwsale_id>1000000<wwsale_id>
```

See also

None.

select

Description

`select` simply constructs a SQL `select` statement using `select` and whatever is contained in `params`.

Input fields

XML tag	Description
`<params>`	`SELECT` SQL statement, minus the `SELECT`.

Return fields

XML record set.

Example

Example invocation:

```
<func>select</func><params>* from prefs_cc</params>
```

Example return string:

```
OK :<xml xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'  ..... >
```

See also

getuniquekey
insert
update

setpassword

Description

Sets the password of the indicated record. The `setpassword` function works on the `operator`, `guests`, and `b_instr` tables. The `force` parameter only ignores the `ominpwage` setting in `prefs` for an operator. The number of passwords that cannot be reused for an operator (`oreusepswd`) is always respected. If successful, `setpassword` re-writes the `passwords` field of the indicated record.

The password functions require a minimum password length of 8 with at least 1 character and 1 number for each password. They enforce not reusing a certain number of passwords for operators as set with `prefs.oreusepswd`. They don't allow passwords to be changed until a certain period of time has elapsed (again, for operators as set with `prefs.ominpwage`). They log the number of failed attempts (if `prefs.ologattmpt` and `prefs.glogattmpt` are set). They can lock out an operator for a period of time (minutes in `prefs.olocktime`) if `prefs.ologattmpt` is exceeded. And they can lock out a guest if `prefs.glogattmpt` is exceeded.

Input fields

XML tag	Description
`<table>`	Table (the `operator`, `guests`, or `b_instr` table).
`<id>`	The `op_code` for the `operator` table, the `guest_no` for the `guests` table, or the `instr_id` for the `b_instr` table.
`<password>`	Password. *Note*: What is entered into the `<password></password>` tags for both `checkpassword` and `setpassword` is case sensitive. However, passwords entered in Sales or SysManager and hashed/stored in the SiriusSQL database for operators and instructors are always upper case. So when using these functions for operators and instructors, always enter characters in the `<password></password>` tags in upper case. Characters entered into the `<password></password>` tags for guests can be mixed case as these passwords are hashed/stored in the SiriusSQL database in mixed case.
`<force>`	Set to `1` to override lockouts.

Return fields

Status only.

Example

Example invocation:

```
<func>setpassword</func><table>guests</table><id>212037000</id><password>Time2Eat</password>
```

Example return string:

```
OK
```

See also

checkpassword

update

Description

`update` constructs a SQL `UPDATE` statement using whatever is contained in `params`.

Note: In order to ensure compatibility with other Salesware applications, you should not use this call. Compatibility is ensured only if you use the pre-defined calls that write data to the SiriusSQL database.

Input fields

XML tag	Description
`params`	`UPDATE` SQL statement, minus the `UPDATE`.

Return fields

Example

Example invocation:

```
<func>update</func><params>guests set first_name='JOSEPH' where guest_no=4168078001</params>
```

Example return string:

```
OK :System Info Read, System Info Read, Function update did not return information
```

See also

getuniquekey
insert
select