Home > 8 ShellManager API Guide: Integration with External Systems

8 ShellManager API Guide: Integration with External Systems

Introduction

This document describes Web Service Interface (WSI), which is a ShellManager API and data schema for exchanging orders between ShellManager and third-party software, e.g. an ERP system.

A production environment without WSI requires orders to be created twice, first in the customer's ERP system and then in RegClient. To avoid this redundancy, using WSI can automate this process, enabling the ERP system to transfer orders to ShellManager automatically.

WSI is an extension to ShellManager, allowing orders to be submitted and retrieved by external systems. It encapsulates the core ShellManager database, thus enabling future upgrades of ShellManager, EarMouldDesigner and ShellDesigner with minimum impact.

Requirements

WSI can be installed as a Windows service or as an application under Internet Information Services (IIS) (see the chapter Audio Data Services Setup). To be able to install and run WSI, the following applications and components are required:

- ShellManager
- Microsoft IIS 6.0 (7.0 or higher recommended)
- Windows Server 2003/2008/2012 or Windows 7/8/8.1 (32-bit/64-bit)

See also:
WSI, Structure, Input and Output, Command List
External IDs, WSI Callbacks and Error Codes

Home > 8 ShellManager API Guide: Integration with External Systems > 8.1 WSI, Structure, Input and Output, Command List

8.1 WSI, Structure, Input and Output, Command List

About WSI

Orders are created and handled in the customers own ERP/Order system (e.g. Navision, SAP, Bespoke, etc.) and sent to the ShellManager through the order interface. ShellManager controls the actual daily engineering and manufacturing of hearing instruments via scanning, modelling, printing, milling, reporting, quality control and mounting. At any time the ERP/order system can retrieve the details and current status of the orders.

Technically, the API consists of the Web Service Interface (WSI) application (.NET application) hosted either on Microsoft Internet Information Server or as a Windows service. WSI acts like a proxy between ShellManager and third-party systems and allows the interaction between them. WSI directly interfaces with ShellManager Audio Data Services.

WSI has a single page for processing commands (`http://<server>:<port>/WSIQueryService/ProcessCommandsPage`) that takes an input XML string as a parameter and returns an output XML string as a result.

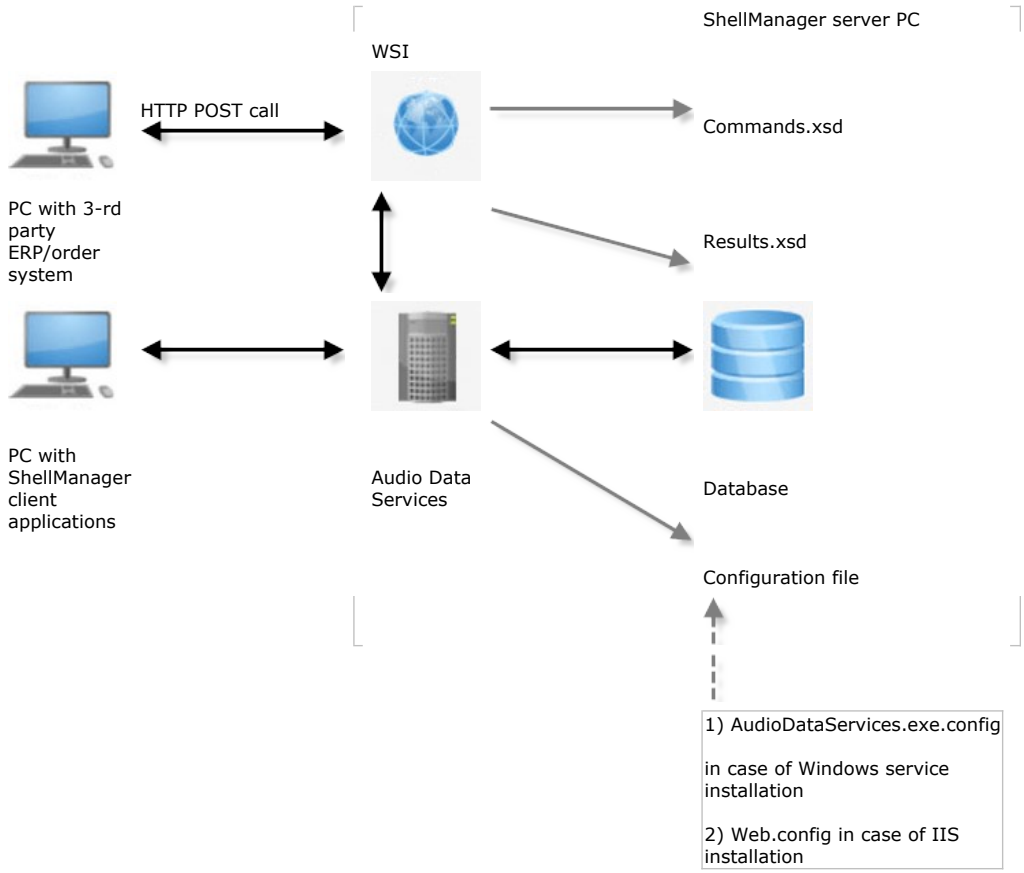
The input XML must be valid to the Commands.xsd schema. The result XML must be valid to the Results.xsd schema. Please find additional information about input commands and resulting XML described in Commands.xsd and Results.xsd schema files, both located in the C:\Program Files\3Shape\AudioDataServices\Implementation\XSD folder.

The input XML string sent to WSI can contain a set of commands, e.g. you can specify three commands, with the result containing answers for all three. The returned XML string contains the results of the operation. Supported operations can be categorized into the following:

- Orders (create, read, update, remake and delete)
- Groups (create, close and open)
- List (get lists of models, vents, wax guards, faceplates, etc.)

To distinguish commands (and corresponding results), the ID attribute of Command type (see the Commands.xsd schema file) must be used. Set ID to 1 for the first command, 2 for the second and so on, and between returned results, you will need to find the one with ID equal to 1, 2 respectively.

> Note: The XML schemas used in WSI have been made according to W3C standards. For introduction to XML schemas, we suggest reading chapters 1-3 of this W3C manual: http://www.w3.org/TR/xmlschema-0/.



ShellManager integration connectivity diagram

Accessing WSI

Web Service Interface is accessed through different URLs depending on the protocol used:

- [HTTP GET]: `http://<server>:<port>/WSIQueryService/ProcessCommandsViaGet?inputString=string`

Note: The `inputString` parameter must contain XML text (see Structure of WSI below).

Warning: All software supporting the GET method used to have limits for query size. That means that if you have very large query, your call may fail. Please refer to the corresponding manual of the software you use to know the exact limit value.

- [HTTP POST]: `http://<server>:<port>/WSIQueryService/ProcessCommandsViaPost`

Note: The command must have a single parameter called `inputString` that must contain XML text (see Structure of WSI below).

- To access WSI via SOAP, use the following link: `http://<server>:<port>/WSIService/ProcessCommands`.

If you want to use WSDL (`http://<server>:<port>/WSIService/ProcessCommands?WSDL`), you need to do the following steps if you have installed Audio Data Services as a Windows service:

Step 1: Stop 3Shape Audio Data Services in Windows Services.

Step 2: Open the configuration file (AudioDataServices.exe.config) of Audio Data Services, which is located in the installation folder of Audio Data Services.

Step 3: Change the value of EnableWSDL to true: `<add key="EnableWSDL" value="true"/>`.

▶ Step 4: Make sure the `<server>` has the name or IP address of the machine where Audio Data Services is installed instead of `http://localhost`.

▶ Step 5: Save the changes.

▶ Step 6: Restart 3Shape Audio Data Services in the Windows Services.

▶ Step 7: If you have installed Audio Data Services as an application under Internet Information Services (IIS), you also need to enable WSDL in the Web.config file which is located in the root folder C:/Program Files/3Shape/AudioDataServices.

- To access WSI from the browser (mainly for debugging purposes), use the following link: `http://<server>:<port>/WSIQueryService/ProcessCommandsPage`.

Note: This way uses HTTP POST method.

The server and port parameters are the host and the port of Audio Data Services.

Structure of WSI

The communication interface consists of XML request-and-response messaging structure. Both XML messages have the root element `<OrderInterface>`, which encapsulates either the command list as the input or a result list as the output.

The root element `<OrderInterface>` has the attributes version and type. The `version` attribute indicates the version of the order interface. The `type` attribute can be either `"Input"` or `"Output"`. The type `"Input"` implies a command and the type `"Output"` implies a result.

Below is an example of an input XML string. This string does not contain any commands, only the basic input XML structure:

```
<?xml version="1.0" ?>
<OrderInterface version="1.0" type="Input" xmlns="http://3shape.com/ThreeShape.HI.Manager.OrderInterface/Commands.xsd">
   <CommandList></CommandList>
</OrderInterface>
```

The format of the input XML string is defined in the Commands.xsd schema (see About WSI above), and it must be referenced in the root element of the input string.

The input XML string is validated against the Commands.xsd schema. So if validation of the input XML string fails, an error is returned with a message element, but with no `<ResultList>` element in the output XML string.

Below is the example of an output XML string. This string would be the result of the above input string. The Results.xsd schema contains the format of the output XML string. Reading the Results.xsd schema (see About WSI above) will provide you with detailed specification of the output XML strings:

```
<?xml version="1.0" encoding="utf-16"?>
<OrderInterface version="1.0" type="Output" xmlns="http://3shape.com/ThreeShape.HI.Manager.OrderInterface/Results.xsd">
  <ResultList></ResultList>
</OrderInterface>
```

Input Commands

The input XML string has the command list which can handle any number of commands. A command must contain both a command identifier and an `id` attribute:

```
<CommandReadOrder id="3435">.
```

The command identifier indicates which type of operation needs to be carried out. This could be `ReadOrder`, like in the example above, which will try to read out an order with a specified order ID (not the `id` attribute of the command). The `id` attribute is required for identifying one command from another when executing several simultaneous commands. By setting an `id` attribute (this should be unique within an XML string), the corresponding result element will then have the same `id` attribute (see Output Results below).

Below is the example of a simple command element which deletes the order with the order ID 23040:

```
<CommandDeleteOrder" id="3441">
    <OrderID>23040</OrderID>
</CommandDeleteOrder >
```

Detailed specification of commands and elements in each command can be found the in the Commands.xsd schema (see About WSI above).

Output Results

The output XML string contains a result list, with as many results as there were commands in the input XML string. The basic attributes of a result are `id` and `success`:

```
<Result success="true" id="3437">
```

The `id` attribute of the result is the same as in the corresponding command. If the operation of the command was successful, the `success` attribute returns `"true"`, if not - `"false"`. For example, if you try to read an order which does not exist, the `success` attribute would return `"false"`.

Below is an example of a complete result element (could be the result for the <CommandDeleteOrder>).

```
<Result success="true" id="3434">
</Result>
```

All result elements can also contain a message element and an error code, which is more descriptive than just true or false.

Detailed specification of results and elements in each result can be found in the Results.xsd schema (see About WSI above).

Command-Result Examples

Example 1: CommandDeleteOrder

Input:

```
<?xml version="1.0"?>
<OrderInterface xmlns="http://3shape.com/ThreeShape.HI.Manager.OrderInterface/Commands.xsd" type="Input" version="1.0">
   <CommandList>
      <CommandDeleteOrder id="1">
         <GroupID>707</GroupID>
         <OrderID>707</OrderID>
      </CommandDeleteOrder>
   </CommandList>
</OrderInterface>
```

Result:

```
<?xml version="1.0" encoding="UTF-16"?>
<OrderInterface xmlns="http://3shape.com/ThreeShape.HI.Manager.OrderInterface/Results.xsd" type="Output" version="1.0">
   <ResultList>
      <Result success="true" id="1"></Result>
   </ResultList>
</OrderInterface>
```

Conclusion:

Command with ID 1 has been successfully performed.

Example 2: CommandGetColorList

Input:

```
<?xml version="1.0"?>
<OrderInterface xmlns="http://3shape.com/ThreeShape.HI.Manager.OrderInterface/Commands.xsd" type="Input" version="1.0">
   <CommandList>
      <CommandGetColorList id="1"></CommandGetColorList>
   </CommandList>
</OrderInterface>
```

Result:

```
<?xml version="1.0" encoding="UTF-16"?>
<OrderInterface xmlns="http://3shape.com/ThreeShape.HI.Manager.OrderInterface/Results.xsd" type="Output" version="1.0">
   <ResultList>
      <Result success="true" id="1">
         <ColorList>
            <Color>
               <ID>23</ID>
               <Name>Red</Name>
            </Color>
            <Color>
               <ID>24</ID>
               <Name>Orange</Name>
            </Color>
         <ColorList>
      </Result>
   </ResultList>
</OrderInterface>
```

Conclusion:

List of two colors (red and orange) has been returned for the command with ID 1.

Command List

Command	Description
CommandAddOrderAttachment	Adds attachment files to the specified order.
CommandCheckLogin	Checks if the given user/password is defined on the server.
CommandCloseGroup	Closes the group.

CommandCopyScans	Copies scan and modelling data.
CommandCreateDispenser	Creates a new dispenser.
CommandCreateGroup	Creates a new group.
CommandCreateOrder	Creates a new order.
CommandCreateOrUpdateOrder	Creates an order. If the order already exists, then updates it with data from the command input.
CommandDeleteAllOrderAttachments	Deletes all attachments from the order.
CommandDeleteOrder	Deletes the order.
CommandDeleteOrderAttachment	Deletes a single attachment or specific collection of attachments from the order.
CommandEarLocked	Checks if a certain ear is locked.
CommandErrorMark **New**	Assigns or cancels the Error Mark property as well as Error Message of the shell by specifying Group ID, External Order ID, External Shell ID and External Error Code.
CommandFinalizeGroup	Finalizes an existing group.
CommandGetAllOrderAttachments	Gets all attachments of the order.
CommandGetColorList	Gets the list of colors.
CommandGetCreatedOrderList	Gets the list of orders created in a specific period of time.
CommandGetCustomFieldList	Retrieves the list of custom fields. Each result item contains ID, Name and Type. The fields in the list will also include entries. The Restriction (for entries) can also be set when calling this command (see the CommandWithRestriction type for details on Restriction).
CommandGetDispenserList	Gets the list of dispensers.
CommandGetErrors **New**	Retrieves the list of errors by specifying a Group ID and External Order ID. The `<IncludeHandled>` tag can be used to specify whether or not to include already handled errors. The tag accepts boolean values: "true" or "false". The default value is "false".
CommandGetFaceplateList	Gets the list of faceplates.
CommandGetGroupList	Gets the list of groups.
CommandGetModelList	Gets the list of available models.
CommandGetOffsetTemplateList	Gets the list of offset templates.
CommandGetOptionalComponentList	Gets the list of optional components (e.g. amplifier, telecoil, etc.).
CommandGetOrderList	Gets the list of orders from a specific group.
CommandGetOrdersShortInfo	Gets the list of order information details for all orders in a specific group. Each entry in the list corresponds to a single order, so for binaural orders there will be two entries.
CommandGetScan	Retrieves the scan of particular type from the order.
CommandGetShapeList	Gets the list of shapes.
CommandGetShellSizeList	Gets the list of shell sizes.
CommandGetSites	Gets the list of sites registered in the system.
CommandGetSoundboreList	Gets the list of soundbores.
CommandGetStatusList	Gets the list of available status codes.
CommandGetTransducerList	Gets the list of transducers.
CommandGetVentList	Gets the list of available vent types.
CommandGetVersionNumber	Gets the version number of the software.
CommandGetWaxList	Gets the list of available wax guards.
CommandLockEar	Requests to lock or unlock the given ear (depends on the lock parameter value).
CommandMoveOrderToGroup	Moves the order to the specified group. Order files are also moved in the production folder.
CommandOpenGroup	Opens the group that is closed.
CommandQueryCreateOrder	Tests if the order can be created.
CommandReadOrder	Reads the order.
CommandRemakeOrder	Creates a remake of the existing order. Any parameters not supplied will be inherited from the original order. The optional RemakeCodeID field added.
CommandSaveScan	Saves the order with scan data.
CommandScanExists	Checks if a scan of specific type exists in the order.
CommandSearch	Searches for orders that match specific conditions.
CommandSiteExists	Returns true if the specified site is registered in the system.
CommandTransferOrder	Sends the order to the remote site.
CommandReturnOrder	Retrieves the order back to its original site.
CommandUpdateOrder	Updates the existing order. PrintJobName, PrintJobType and PrintName information is provided in the callback data after the order status is changed to PRINTED.

See also:
External IDs, WSI Callbacks and Error Codes

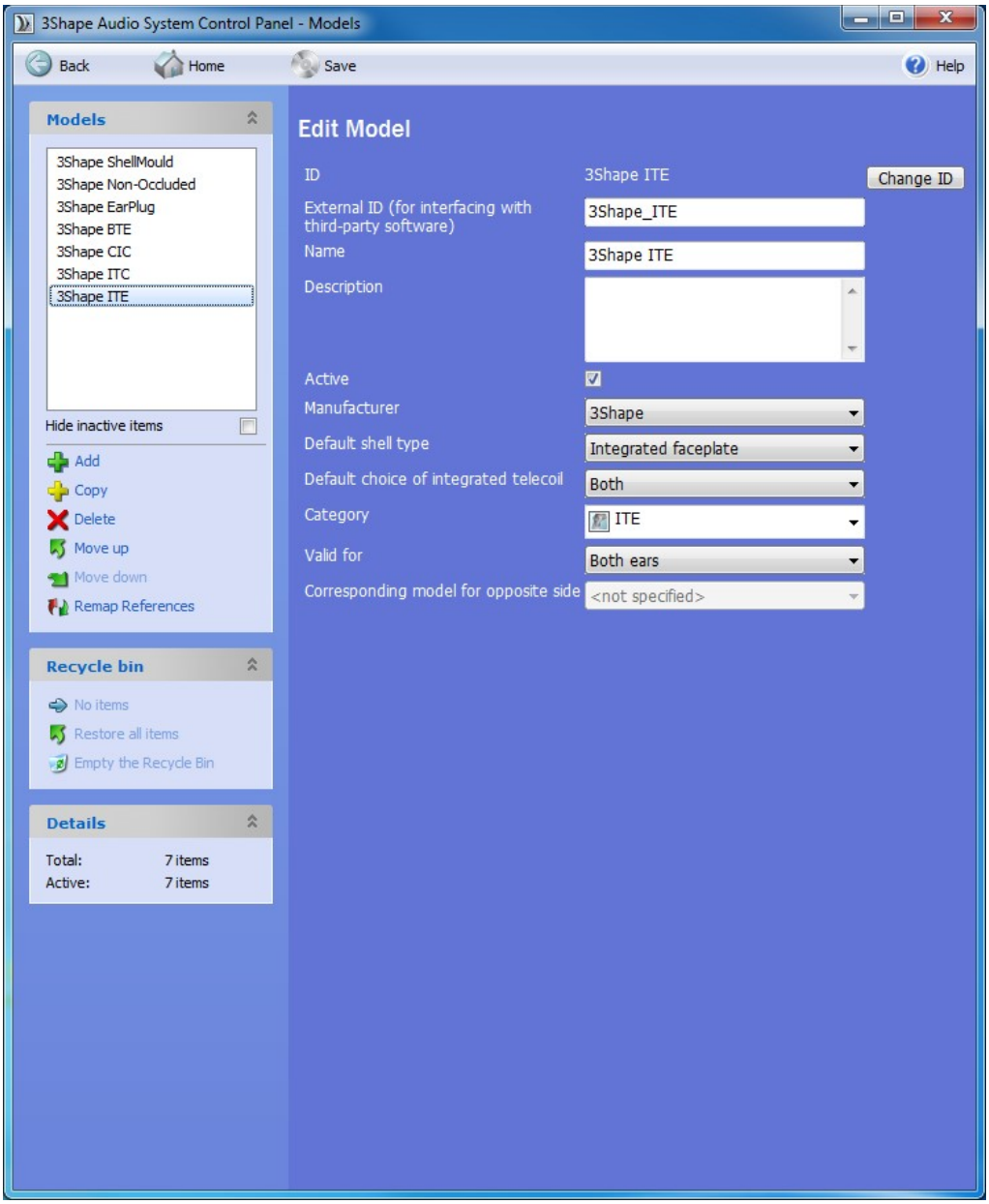
8.2 External IDs, WSI Callbacks and Error Codes

External IDs

All external access to materials is done through external IDs. All items in the ShellManager system that you wish to access through WSI should have external IDs created. A list of the reference keys is shown below, which must be set up for all selectable options. If an option has no defined reference value, it cannot be selected and the call will fail.

Reference key	Control Panel	Attribute name
DispenserID	ShellManager/Dispensers	Reference
CategoryID	Materials/Categories	External ID
ModelID	Materials/Models	External ID
SizeID	Materials/Sizes	External ID
FaceplateID	Materials/Faceplate systems	External ID
VentID	Materials/Vents	External ID
WaxID	Materials/Wax guards	External ID
TransducerID	Materials/Transducers	External ID
OptionalComponentID	Materials/Optional components	External ID
ShapeID	Materials/Shapes	External ID
SoundboreID	Materials/Sound bores	External ID

This is managed in Control Panel by going to Materials > Models:



Models setup using the External ID attribute

WSI Callbacks

In order for a third-party system (i.e. ERP, order handling system) to be informed on status changes, a callback function is available.

When an order change takes place, an event notification is sent as an HTTP call to the preconfigured URLs. These sites should be created in a specific section of Audio Data Services configuration file. The name of the configuration file depends on the way Audio Data Services has been installed (see the chapter Audio Data Services Setup). If it is installed as a Windows service, then the file name is HostApplication.exe.config. In case when Audio Data Services is installed as an application under Internet Information Services (IIS), the file name will be Web.config. The configuration file can be found in the folder C:/Program Files/3Shape/AudioDataServices:

```
<callbackUrls>
   <add name="TestSite1" url="www.mysite1.com/path1" />
   <add name="TestSite2" url="www.mysite2.com/some path" />
</callbackUrls>
```

The following table shows lists of callbacks resulted from WSI commands:

Command	List of resulting callbacks
CommandCreateOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderCreate UserID=ERP
CommandCreateOrUpdateOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange

	UserID=ERP
CommandDeleteOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderDelete UserID=ERP
CommandRemakeOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderCreate UserID=ERP ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=RemakeCreate UserID=ERP
CommandTransferOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=TransferToRemoteSite UserID=ERP
CommandUpdateOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=StatusChange EarID=< EarID > EarSide=LEFT UserID=ERP OldStatus=< OldStatus > NewStatus=< NewStatus> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=ERP ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=StatusChange EarID=< EarID > EarSide=RIGHT UserID=ERP OldStatus=< OldStatus > NewStatus=< NewStatus> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=ERP ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=ERP
CommandReturnOrder	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=< UserID >/ERP ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=TransferFromRemoteSite UserID=ERP
CommandErrorMark	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderErrorMark EarID=< EarID > EarSide=LEFT ErrorDescription=< ErrorDescription > ErrorCode=< ErrorCode ID> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderErrorMark EarID=< EarID >

	EarSide=RIGHT ErrorDescription=< ErrorDescription > ErrorCode=< ErrorCode ID>

The following table shows lists of callbacks resulted from manual operations:

Action	List of resulting callbacks
Create order in RegClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderCreate UserID=< UserID >
Modify order details in RegClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=<UserID>
Create Remake in RegClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderCreate UserID=<UserID> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=RemakeCreate UserID=<UserID>
Delete order in RegClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderDelete UserID=< UserID >
Approve order using the ***Approve*** button in ModelClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=StatusChange EarID=< EarID > EarSide=LEFT UserID=< UserID > OldStatus=< OldStatus > NewStatus=< NewStatus> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=< UserID > ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=StatusChange EarID=< EarID > EarSide=RIGHT UserID=< UserID > OldStatus=< OldStatus > NewStatus=< NewStatus> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=< UserID >
Approve order after modelling in the approve/disapprove dialog	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=< UserID >
Error mark order in ModelClient in the approve/disapprove dialog	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderErrorMark EarID=< EarID > EarSide=LEFT ErrorDescription=< ErrorDescription > ErrorCode=< ErrorCode ID> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderErrorMark

	EarID= <EarID> EarSide=RIGHT ErrorDescription=< ErrorDescription > ErrorCode=< ErrorCode ID> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=< UserID >
Error mark order in RegClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderErrorMark EarID=< EarID > EarSide=LEFT ErrorDescription=< ErrorDescription > ErrorCode=< ErrorCode ID> ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderErrorMark EarID=< EarID > EarSide=RIGHT ErrorDescription=< ErrorDescription > ErrorCode=< ErrorCode ID>
SWIFT order from RegClient	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=TransferToRemoteSite UserID=<UserID>
Return order from remote site	SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=OrderChange UserID=< UserID > ---------- SiteID=< SiteID > OrderID=< OrderID > TimeStamp=<YYYY-MM-DD HH:MM:SS> EventType=TransferFromRemoteSite UserID=< UserID >

Note: All resulting callbacks are shown for binaural orders. In case of a monaural order or if sides have different properties (for example, in the Approve/Disapprove dialog), callback will describe actions, if any, for each side.

Each callback uses the GET method. For example, a callback URL: `www.mysite1.com/path1` will be translated into a callback URL string: `www.mysite1.com/path1?param1=value1&param2=value2`.

The destination site should then parse the URL string and interpret the relevant event and its appropriate parameters:

Parameter	Description	Type	Example
OrderID	Internal order ID	Alphanumeric (JournalExtID)	113
EarID	External order ID	Alphanumeric (ShellExtID)	133
EarSide	Side of ear	Alphanumeric (LEFT or RIGHT)	RIGHT
ErrorDescription	Error mark message	Alphanumeric	Vent is placed incorrectly
ErrorCode	External ID error code	Alphanumeric	Minor
TimeStamp	Event time and date	Alphanumeric (YYYY-MM-DD HH:MM:SS)	2015-02-27 12:52:02
EventType	Type of event (see table above)	Alphanumeric (see EventType table below)	StatusChange
UserID	User that caused the event	Alphanumeric (LoginID)	Admin
OldStatus	Status that was before change	Alphanumeric (REGISTERED, SCANNED, MODELLED, CHECKED, MILLED, MOUNTED, PRINTED or SHIPPED)	REGISTERED
NewStatus	Status that is after change	Alphanumeric (REGISTERED, SCANNED, MODELLED, CHECKED, MILLED, MOUNTED, PRINTED or SHIPPED)	SCANNED
PrintJobName	Name of print job	Alphanumeric	Job1
PrintJobType	Type of print job	Alphanumeric	Beige
PrinterName	Id of printer	Alphanumeric	Printer1
SiteID	Id of site that initiated	Alphanumeric	MySite1

	an event		

The following table shows possible event types (each value in the first column is an actual value for **EventType** parameter in the URL string):

EventType	Description
StatusChange	Order status is changed. PrintJobName, PrintJobType and PrinterName is provided if new status is PRINTED or MILLED.
OrderCreate	New order has been created.
OrderDelete	Existing order has been deleted.
OrderChange	Order details have been updated.
OrderErrorMark	Occurs when the user error marks the order.
TransferToRemoteSite	The order has been sent (SWIFT) to a remote site.
TransferFromRemoteSite	The order has been received (SWIFT) to a remote site.
RemakeCreate	Remake order has been created.
GeometryAccept	Obsolete
GeometryReject	Obsolete

Error Codes

The API returns predefined codes if an error occurs:

Error ID	Type	Description
1	ID exists	Order already exists with the specified ID.
2	ID doesn't exist	Order doesn't exist.
3	ID duplication	Duplicate order with the same ID.
4	Not changeable	Order does not support changing (i.e. unable to change status).
5	ID missing	Referenced model error (i.e. faceplate model ID is not supplied).
6	Insert failed	Duplicate model error (i.e. optional component already exists in order).

See also:
WSI, Structure, Input and Output, Command List