Deltek VisionXtend™

Web Services and APIs for Deltek Vision

Deltek Vision 6.1

Deltek®
13880 Dulles Corner Lane
Herndon VA 20171
TEL: 703.734.8606
FAX: 703.734.1146

While every attempt has been made to ensure that the information in this document is accurate and complete, some typographical or technical errors may exist. Deltek, Inc. cannot accept responsibility for any kind of loss resulting from the use of this publication.

This page shows the original publication date. The information contained in this publication is subject to change without notice. Any improvements or changes to either the product or the book will be documented in subsequent updates.

This edition published June, 2009

Contents

Introduction

Vision incorporates service-oriented architecture (SOA) by supporting various Web services.

This documentation specifically applies to Vision Web services that provide Web-accessible application programming interfaces (APIs) that can be used by external applications to send data to and receive data from the Vision info centers.

This document provides detailed technical information about the available Web services and methods, and describes how these Web services can be used by other applications. This document also provides examples with code samples to demonstrate proper usage of Web services, and includes descriptions of input and output messages handled by these Web services.

Overview

The APIs exposed through Web services are mainly intended to carry out three types of database transactions related to info centers; namely **Insert**, **Update**, and **Delete**. There are Web services specific to each info center that can be separately called to carry out any of the three types of transactions. A generic API is also available to run one, or a combination of more than one, type of transaction for any given info center by making a single call.

Web Services Description Language (WSDL) documents for Web services are included and they provide critical information for developers to understand the interface and functional behaviors of APIs represented by Web services. WSDL Simple Object Access Protocol (SOAP) Binding is used to describe Web services.

The proxy class files for Web services are included and they are available for Vision Basic .NET (VB.NET) and C# languages. In addition, a compiled dynamic link library (DLL) is also available to access Web services.

All of the APIs require at least two parameters in Extensible Markup Language (XML) format: One consisting of Vision login information including Vision user-name, password, and the database; and the other consisting of the actual data that needs to be saved to Vision database. XML schemas for all Info Centers are included and they describe expected format of the input XML data that is passed to Web services.

Detailed descriptions of return values and messages, including exception handling and error capturing methods, are included below.

The following is a brief description of available Web-accessible APIs. Details of calling syntax, along with parameters required and return values, are included in the next section of this document.

SendDataToDeltekVision

A generic API that can be used to send data to any of Vision's supported Info Centers.

Each supported Info Center has two APIs available, one to add new records and the other to update existing records. For example, for Employee Info Center, the APIs are AddEmployee() and UpdateEmployee().

The basic functional steps of all of the APIs are as follows:

1. **XML data validation**: Input XML data is validated against published schema.
2. **Vision login validation**: Checks if Vision login credentials are valid.

3. **Vision record access validation**: Validates record-level access rights (add and modify) for the given Vision user.
4. **Data validation**: Input data is validated to ensure referential integrity and other database column level dependencies.
5. **Save data**.

The general format of the return message is shown below. The message format is in XML. The 'ReturnCode' is a number and 'ReturnDesc' is a short description of the result returned by the service/API. The 'Detail' node consists of additional relevant information based on the return code.

```
<DLTKVisionMessage>
        <ReturnCode></ReturnCode>
        <ReturnDesc></ReturnDesc>
        <Detail></Detail>
</DLTKVisionMessage>
```

List of Web Services/APIs

All of the Web services use at least two or all of the following calling parameters:

- **InfoCenter**. Represents the generic Info Center name. For example, 'Employees', 'Projects', 'Clients', and so on.
- **ConnInfoXML**. Vision login information represented in XML.
- **DataXML**. The data that needs to be saved to Vision database. This is in XML and it complies with the XML schema provided for the Info Center.

Return values for all services are sent in XML format as strings. Detailed information regarding parameters, messages-returned and their format can be found later in this document under the heading 'Application Development Notes'.

The following table contains a list of APIs available through Web services.

SendDataToDeltekVision(InfoCenter, ConnInfoXML, DataXML)	
Description	A generic API that can add or update information in any Info Center. The value of 'InfoCenter' parameter determines the Info Center area. DataXML needs to include values for internal key fields. DataXML can include data for multiple tables related to the Info Center. For example, adding an employee along with multiple degrees and skills for that employee.
AddProject(ConnInfoXML, DataXML)	
Description	Adds new projects. DataXML needs include primary-key values as specified in the XML schema.
UpdateProject(ConnInfoXML, DataXML)	
Description	Updates existing project records. The key values (internal or user-entered) for tables need to be included in the XML input.
AddClient(ConnInfoXML, DataXML)	
Description	Adds new client records. DataXML does not need to include values for the internal key column 'ClientID'. Internal keys are generated automatically and they are passed back to the calling application once records are added.
UpdateClient(ConnInfoXML, DataXML)	
Description	Updates existing client records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
SendDataToDeltekVision(InfoCenter, ConnInfoXML, DataXML)	
Description	Adds new contact records. DataXML does not need to include values for the internal key column 'ContactID'. Internal keys are generated automatically and they are passed back to the calling application once records are added.

UpdateContact(ConnInfoXML, DataXML)	
Description	Updates existing contact records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
AddEmployee(ConnInfoXML, DataXML)	
Description	Adds new employee records. DataXML needs to include primary key values as specified in the XML schema.
UpdateEmployee(ConnInfoXML, DataXML)	
Description	Updates existing employee records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
AddOpportunity(ConnInfoXML, DataXML)	
Description	Adds new opportunity records. DataXML does not need to include values for the internal key column 'OpportunityID'. Internal keys are generated automatically and they are passed back to the calling application once records are added.
UpdateOpportunity(ConnInfoXML, DataXML)	
Description	Updates existing opportunity records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
AddVendor(ConnInfoXML, DataXML)	
Description	Adds new vendor records. DataXML needs to include primary-key values as specified in the XML schema.
UpdateVendor(ConnInfoXML, DataXML)	
Description	Updates existing vendor records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
AddLead(ConnInfoXML, DataXML)	
Description	Adds new lead records. DataXML does not need to include values for the internal key column 'LeadID'. Internal keys are generated automatically and they are passed back to the calling application once records are added.
UpdateLead(ConnInfoXML, DataXML)	
Description	Updates existing lead records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.

AddCampaign(ConnInfoXML, DataXML)	
Description	Adds new marketing-campaign records. DataXML does not need to include values for the internal key column 'CampaignID'. Internal keys are generated automatically and they are passed back to the calling application once records are added.
UpdateCampaign(ConnInfoXML, DataXML)	
Description	Updates existing marketing-campaign records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
AddTextLibrary(ConnInfoXML, DataXML)	
Description	Adds new text library records. DataXML needs to include primary-key values as specified in the XML schema.
UpdateTextLibrary(ConnInfoXML, DataXML)	
Description	Updates existing text library records. Multiple tables within a single Info Center can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.
AddActivity(ConnInfoXML, DataXML)	
Description	Adds new activity records. DataXML does not need to include values for the internal key column 'ActivityID'. Internal keys are generated automatically and they are passed back to the calling application once records are added.
UpdateActivity(ConnInfoXML, DataXML)	
Description	Updates existing activity records. Multiple activity-related tables within can be updated with a single call. The key values (internal or user-entered) for tables need to be included in the XML input.

XML Schema for Vision Web Services/APIs

The data that needs to be added to or updated in Vision database is sent in XML format. The format of the XML data has to comply with the schema provided.

Each applicable Info Center in Vision has an XML schema defined and these schema files can be found on the Vision Web/app server in <InstallDir>\Vision\Web\Xsd directory (<InstallDir> is the directory where Deltek Vision is installed). The names of the schema files start with the generic Info-Center-name followed by '_Schema.xsd'. For example, the name of the XML schema file used for Employee Info Center would be 'Employee_Schema.Xsd'.

Handling Primary Key Columns That Use Internally Generated IDs

Some of the Info Centers have an internally generated unique key (GUID – globally unique identifier) as the primary key, and primary key values do not need to be provided when calling specific Web services to add new records. For example, AddClient, AddContact, etc.

In order to have APIs generate unique values (GUIDs) for the internal primary key column on the fly, simply include a generic value '@Generate' for both the key attribute and key column. This generic value serves as the placeholder for the key value that is generated automatically.

The following Info Centers use an internal primary key, and the key columns in the database are listed below.

Info Center	Internal ID – Primary Key Column	Table
Client	ClientID	CL
Contact	ContactID	Contacts
Opportunity	OpportunityID	Opportunity
Lead	LeadID	Leads
Marketing Campaign	CampaignID	MktCampaign
Activity	ActivityID	Activity

The following is an example of XML input that specifies the use of auto-generated internal key values.

```
<RECS>
        <REC>
                <Leads name="Leads" alias="Leads" keys="LeadID">
                        <ROW tranType="INSERT”>
                                <LeadID>@Generate</LeadID>
                                <Prefix>Mr.</Prefix>
                                <FirstName>Trevor</FirstName>
                                <MiddleName>Kelly</MiddleName>
                                <LastName>Jackman</LastName>
```

```
                                        .
                                        .
                        </ROW>
                <LeadCustomTabFields name="LeadCustomTabFields"
alias="LeadCustomTabFields" keys="LeadID">
                        <ROW tranType="INSERT">
                                <LeadID>@Generate</LeadID>
                                <custTest1Char>MyTest</custTest1Char>
                                        .
                                        .
                        </ROW>
                </LeadCustomTabFields>
                <LeadFileLinks name="LeadFileLinks" alias=" LeadFileLinks "
keys="LinkID,LeadID">
                        <ROW tranType="INSERT">
                                <LinkID>@Generate</LinkID>
                                <LeadID>@Generate</LeadID>
                                <Description>MyTestFile</Description>
                                        .
                                        .
                        </ROW>
                </ LeadFileLinks >
        </REC>
</RECS>
```

The XML schema consists of constraints and rules to ensure that the incoming XML data is valid for Vision Web services to process. The schema needs to be carefully followed to make sure that the calling application is sending XML data in the right format.

The schema is designed to validate a number of things with respect to XML data that is sent as a parameter when calling Web services. The following are some of the validations that are done:

- Data type and length/size
- Uniqueness of the key (within the XML ata)
- Mandatory columns/attributes
- Enumerated values

The basic format of the XML data in a simplistic representation looks like the following:

```
<RECS>
        <REC>
                <@TableName name="" alias="" keys="">
                        <ROW transType="">
                                <@key></@key>
                                <@Column></@Column>
                        </ROW>
                </@TableName>
                <@TableName name="" alias="" keys="">
                        <ROW transType="">
                                <@key></@key>
                                <@Column></@Column>
                        </ROW>
                </@TableName>
        </REC>
        <REC>
                <@TableName name="" alias="" keys="">
                        <ROW transType="">
                                <@key></@key>
                                <@Column></@Column>
                        </ROW>
                </@TableName>
        </REC>
</RECS>
```

- **@TableName** is the actual table name(s) depending on the Info Center for which the schema is defined.
- The **'@key'** and **'@Column'** references shown above are actual names of key columns and other columns respectively and they are different for different Info Centers. For example in the case of Client Info Center, @key will be replaced with 'clientID' and @Column will be replaced with actual column-names from the main client table (CL).

For a complete description of all the elements and the attributes, refer to schema documents containing the schema definitions.

Depending on the table, there can be one or more <ROW> elements. For the base or primary table, there can only be a single row. However, when certain tables share a one-to-many relationship with the primary table, there can be multiple <ROW> elements. The schema is

designed such that it validates these restrictions and a validation message is returned when incoming XML data violates any of the constraints in the schema.

Internal keys (primary-key columns in Info Center base tables whose names end with 'ID') are not required if an Info-Center-specific Web service/API is used to Add records. Description of different API functions can be found later in this document. When internal key values are not provided, they are generated automatically by the Web service and new key values are returned after new records are added to the database.

Schema for Custom Tabs in Vision Info Centers

As custom tabs for Info Centers are user-defined and they vary from database to database, there is no predefined schema to represent custom tabs. However, there is a Web service/API to dynamically refresh Info Center schemas to contain schema definition for custom tabs based on the current status of the database. Below is a description of the API to refresh schema with custom tabs information.

GenerateCustomTabSchema(ConnInfoXML, InfoCenter)	
Description	Updates existing schema file for the given Info Center with schema definition for custom tabs based on the current state of the Vision database. Schema files for Info Centers are located in <VisionInstallDir>\XSD directory and depending on the 'InfoCenter' parameter value for the call, respective schema file is updated. This Web service can be called any number of times and it always refreshes and replaces existing schema definition for custom tabs in the file with the latest information from Vision database.

Obtaining Latest Schema Files for Vision Info Centers

The latest schema files for Vision Info Centers can be obtained by using a Web service that returns schema file content (String).

Using API

The following API can be used to obtain schema file content for an Info Center.

GetSchema(ConnInfoXML, InfoCenter)	
Description	Based on the value for 'InfoCenter' parameter for the call, content of the respective schema file is returned back.

Application Development Notes

Input parameters Used to call Web Services

All parameters are passed-in as Strings.

InfoCenter

Indicates the name of the Info Center in Vision to/from which the data is being sent-to/retrieved from.

While Info Center names are customizable within Vision application, the following generic names are used while calling Web services. Regardless of what the customized logical names are within Vision, the following Info Center names corresponding to those logical names within Vision need to be used.

Info-Center Name*	Description	Primary Table	Key Column(s)
Projects	Project Info Center	PR	WBS1,WBS2,WBS3
Clients	Client Info Center	CL	ClientID
Contacts	Contact Info Center	Contacts	ContactID
Employees	Employee Info Center	EM	Employee
Opportunities	Opportunity Info Center	Opportunity	OpportunityID
Leads	Leads Info Center	Leads	LeadID
MktCampaigns	Mkt. Campaign Info Center	MktCampaign	CampaignID
Vendors	Vendor Info Center	VE	Vendor
TextLibraries	TextLibrary Info Center	TextLibrary	Name
Activities	Activity Info Center	Activity	ActvityID

*****Used for the 'InfoCenter' parameter while calling Web services.

The key-column names that end with 'ID' use internally generated key values.

ConnInfoXML

Vision login information that is used to authenticate calling application's access to Vision database. The XML format of this parameter is below.

```
<VisionConnInfo>
        <databaseDescription>VisionDemo30</databaseDescription>
        <userName>Admin</userName>
        <userPassword>test</userPassword>
```

```
        <integratedSecurity>N</integratedSecurity>
</VisionConnInfo>
```

- **Database description** — It is the description of the Vision database that has been entered into 'weblink' utility for Vision. If multiple Vision databases are in use, this particular parameter identifies the Vision database that needs to be used for Web-service transactions.
- **User name** — It is the name of the Vision user from Vision application security. In addition to using this Vision login to authenticate calling application's access to Vision database, the access rights of the Vision security role that the user belongs to, is used to determine whether or not new records can be added, and as to which records can be modified.
- **User password** — It is the password stored in Vision database.
- **Integrated security** — This is to indicate if integrated Windows login from the calling machine needs to be used to validate Vision login. Allowable values are 'Y' and 'N'.

DataXML

Input data in XML format that is in compliance with the supplied schema.

Below is a sample DataXML string that can be passed to 'SendDataToDeltekVision' API to add a new lead record. This XML string includes internal-primary-key value (for leadID). This example consists of a single row/record but data XML can contain any number of rows/records.

```
<?xml version="1.0"?>
<RECS xmlns="http://deltek.vision.com/XMLSchema"
xmlns:xdv="http://deltek.vision.com/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <REC>
                <Leads name="leads" alias="leads" keys="leadID">
                        <ROW tranType="INSERT”>
                                <leadID>WSERV123</leadID>
                                <prefix>Mr.</prefix>
                                <firstName>Trevor</firstName>
                                <middleName>Kelly</middleName>
                                <lastName>Jackman</lastName>
                                <suffix>AIA</suffix>
                                <title>Chief Strategist</title>
                                <description>Test Lead Description</description>
                                <company>My Company, Inc.</company>
                                <email>jtrevor@myCompany.com</email>
                                <website>www.myComp.com</website>
                                <employee>000201</employee>
```

```
                        <source>Client Reference</source>
                        <status>sysNew</status>
                        <statusReason/>
                        <industry>01</industry>
                        <rating>01</rating>
                        <recordStatus>A</recordStatus>
                        <address1>1234 Test Drive</address1>
                        <address2>Suite 123</address2>
                        <address3/>
                        <city>Pleasantville</city>
                        <state>MD</state>
                        <zip>22343</zip>
                        <country>USA</country>
                        <businessPhone>703-989-8878</businessPhone>
                        <businessFax>703-989-8879</businessFax>
                        <mobile>703-989-8899</mobile>
                        <home/>
                        <pager>703-989-8877</pager>
                    </ROW>
                </Leads>
                <LeadCustomTabFields name="LeadCustomTabFields"
alias="LeadCustomTabFields" keys="LeadID">
                <ROW tranType="INSERT">
                    <LeadID>WSERV123</LeadID>
                    <custTest1Char>MyTest</custTest1Char>
                    <custTest2Num>23</custTest2Num>
                    <custTest3Char>Test</custTest3Char>
                    <custTest4Char></custTest4Char>
                    <custTest5Emp>000201</custTest5Emp>
                    <custTest6Lookup>Test1</custTest6Lookup>
                </ROW>
        </LeadCustomTabFields>
        </REC>
</RECS>
```

Return Messages

As specified earlier in this document, return values or messages are sent back in a standard XML format. The calling application can use this information to verify if the API/Service call was successful or if it had any problems. The structure of the return message makes it easy for the application to parse and process it, and determine the next action based on what was returned from Vision.

The format is as shown below.

```
<DLTKVisionMessage>
        <ReturnCode></ReturnCode>
        <ReturnDesc></ReturnDesc>
        <Detail></Detail>
</DLTKVisionMessage>
```

- **Return-Code** is a short-code and it is predefined to indicate success or the type of error.
- **Return-Desc** is a brief description of the message or the event that happened during the call.
- The **Detail node** consists of either further detailed description of the error or return values that correspond to transaction.
- **Return-Code** is always “1” when the service has been successfully run with no errors.

Vision Record Access Validation

In Vision, access to Info Centers in terms of adding new records and updating existing records is controlled at the ‘role’ level. Record level security is assigned to a role and it controls the list of records that a user belonging to that role can edit/update.

All Web-accessible API calls validate record access rights in Vision with respect to the login/user name specified in connection-XML (ConnXML) parameter.

The following is the format of the returned error message when record access validation fails. The <AddRec> section indicates whether or not there is a restriction to add new records and <UpdateRec> section provides information on failed record level security validation. The <ReturnValues> section under <UpadateRec> lists key values of all of the records that are not available to update.

```
<DLTKVisionMessage>
        <ReturnCode>ErrRecAccess</ReturnCode>
        <ReturnDesc>Record Access Validation Error</ReturnDesc>
        <Detail>
                <AddRec>
                        <Message>Permission to add new records is denied</Message>
                </AddRec>
                <UpdateRec>
```

```
                    <Message>Permission to update records is denied</Message>
                    <ReturnValues>
                        <keyVal>ADMIN1074277037011</keyVal>
                        <keyVal>ADMIN1086620567592</keyVal>
                    </ReturnValues>
                </UpdateRec>
            </Detail>
</DLTKVisionMessage>
```

Validation of XML Data

Incoming XML data is validated to ensure that all of the business rules that are either application-enforced or user-defined are completely complied with. For each of the records in each of the tables, the values for applicable columns are checked with respect to each of the business rules. If any of the column values are not in compliance with the business rules, a return message in a specified format is passed back to the calling application. The validation is completed for the entire input XML data before returning the validation result. The validation result would therefore consist of information about all values for various columns that do not satisfy the business rules.

The following is the general format of the validation result that is returned. There can be multiple <Message> nodes in the validation result depending on how many fields and how many rows had validation problems.

```
<DLTKVisionMessage>
        <ReturnCode>ErrDataVal</ReturnCode>
        <ReturnDesc>Data Validation Error</ReturnDesc>
        <Detail>
            <ValidationResult>
                <Message>
                    <Table></Table>
                    <Column></Column>
                    <ColValue></ColValue>
                    <RowNum></RowNum>
                    <PrimaryRecKeyVal></PrimaryRecKeyVal>
                    <MsgCode></MsgCode>
                    <MsgDesc></MsgDesc>
                </Message>
            </ValidationResult>
        </Detail>
</DLTKVisionMessage>
```

Setting Fields as 'Required'

Fields can be set as required based on the role type (CRM, Accounting, or Administrator) by making changes to ValidationInfo.xml file found in <VisionInstallDir>\Web\Xml directory. The settings in this file are used only for Vision APIs and the file also has information used to ensure implicit and explicit referential integrity constraints. For example, ensuring that a field that is bound to a code table accepts only those values that are already present in the code table.

The following is the format of the ValidationInfo.xml file.

```
<XML>
        <InfoCenter name="@InfoCenter">
                <Table name="@TableName">
                        <Column name="@ColumnName" required="@RoleType">
                                <CodeTable>@ReferenceTable</CodeTable>
                                <CodeCol>@ReferenceColumn</CodeCol>
                                <DescCol>@DescriptionColumn</DescCol>
                        </Column>
                        <Column name="@ColumnName" required="@RoleType">
                        </Column>
                </Table>
        </InfoCenter>
</XML>
```

- **@InfoCenter**. The Info Center name.
- **@TableName**. The name of the table related to the Info Center, which is also referenced in the schema.
- **@ColumnName**. The name of the column in the table indicated by @TableName.
- **@RoleType**. The role type from Vision security. The possible values are 'CRM", 'ACCT', and 'Admin'. When 'required' attribute is set as 'ALL', the field will be required for both CRM and ACCT type roles.
- **@ReferenceTable**. The name of the master or reference table against which the data for the column in question needs to be verified.
- **@ReferenceColumn**. The name of the column in @ReferenceTable.
- **@DescriptionColumn**. The name of the column containing the description for code/key in **@ReferenceColumn**. This is not currently being used and it is optional.

As shown above, the nodes <CodeTable>, <CodeCol>, and <DescCol> are optional if a column needs to be set as required but does not have any referential integrity to validate.

The following is an example of what an excerpt of the file ValidationInfo.xml may look like.

```
<XML>
        <InfoCenter name="Projects">
                <Table name="PR">
                        <Column name="Org" required="ACCT">
                                <CodeTable>Organization</CodeTable>
                                <CodeCol>Org</CodeCol>
                                <DescCol>Name</DescCol>
                        </Column>
                        <Column name="ClientID" required="CRM">
                                <CodeTable>CL</CodeTable>
                                <CodeCol>ClientID</CodeCol>
                                <DescCol>Name</DescCol>
                        </Column>
                        <Column name="ProjectType">
                                <CodeTable>CFGProjectType</CodeTable>
                                <CodeCol>Code</CodeCol>
                                <DescCol>Description</DescCol>
                        </Column>
                        <Column name="Description" required="CRM">
                        </Column>
                </Table>
        </InfoCenter>
</XML>
```

The ValidationInfo.xml file can be edited to set ‘required’ attribute for fields that are either already in the file or for new fields that have been added. Changes need to be carefully made and it is strongly recommended that the entries for fields that are a part of the standard ValidationInfo.xml file must not be removed or altered.

Custom Tab Fields Validation

In addition to validating application-defined validation rules, any user-defined validation rules on custom-tab fields are also validated. Following are the user-defined validation constraints that are validated.

- The 'Required' constraint defined based on Vision role-type.
- For numeric values, minimum and maximum bounds are checked to ensure that the value is in the correct range.
- Lookup type fields with the option 'LimitToList' set.
- For fields that have a default-value specified, the default value is automatically into the database if no value is provided in the input XML data.

For each column value that violates validation rules, a <Message> node is added under <ValidationResult>. Each message node consists of table name, column name, row number (as appears in the XML data), the primary key value for the main Info Center record, message code (MsgCode), and message description (MsgDesc) information.

The message code is a predefined code that represents the kind of validation error that has occurred. The message description is a brief text message that explains the error.

Below is a list of message codes and descriptions for validation messages that are included in the <Detail> section.

Validation error code (<MsgCode>)	Validation error description (<MsgDesc>)
VAL1001	Value not in the master/code table
VAL1002	Value is required
VAL1003	Min value constraint not met
VAL1004	Max value constraint not met

The design of the XML schema allows input XML data to omit non-primary tables and fields when adding or updating records. For validation purposes, only those fields that are included in the data-XML are validated, and corresponding validation messages are returned. Deltek recommends that if all required field constraints need to be validated, all of the fields in the table be included in XML data, whether or not they have data.

APIs to Retrieve Data from Vision

There are two basic APIs to get record level information from Info Centers in Vision and there are also Info Center specific APIs to retrieve data from respective Info Centers. In terms of functionality both basic APIs and Info Center specific APIs do exactly the same thing. The difference is with respect to calling parameters and generic APIs require Info-Center-Name as one of the parameters.

Generic APIs

- GetRecordsByKey
- GetRecordsByQuery

GetRecordsByKey(ConnInfoXML, InfoCenter, Keys, RecordDetail)	
Description	The set of records retrieved is based on the key value/values specified in 'keys' parameter. Records are returned in XML format complying with the schema for the particular Info Center. The 'RecordDetail' parameter determines if only the record from the main table is retrieved or if information from all child/association tables are also returned.
Parameters	**ConnInfoXML** — Connection information to validate Vision login and the database access. **InfoCenter** — Info Center name as specified in the beginning of this document. **Keys** — Primary key values separated by a comma. It can also be a single key value to retrieve a single record. **RecordDetail —** If left empty, record(s) from all of the association and child tables are retrieved. To retrieve only the basic information (record from the main Info Center table only), a value of 'Primary' is used with this parameter.
GetRecordsByQuery(ConnInfoXML, InfoCenter, Query, RecordDetail)	
Description	The set of records retrieved is based on the key value/values specified in 'keys' parameter. Records are returned in XML format complying with the schema for the particular Info Center. The 'RecordDetail' parameter determines if only the record from the main table is retrieved or if information from all child/association tables are also returned.
Parameters	**ConnInfoXML** — Connection information to validate Vision login and the database access. **InfoCenter** — Info Center name as specified in the beginning of this document.

GetRecordsByQuery(ConnInfoXML, InfoCenter, Query, RecordDetail)	
Parameters (cont.)	**Query** — A SQL statement selecting records from the main Info Center table. The query can have nested and correlated queries to base selection on any of the related tables. The required format of the query would be: SELECT <Info Center main table>.* FROM <Info Center main table> WHERE <Where Clause> FROM clause can include other tables in addition to the main table depending on how complex the Where clause is. Example query for Employee Info Center: SELECT EM.* FROM EM, Organization WHERE EM.Org = Organization.Org AND **Organization.Name** = 'My Company' **RecordDetail** — If left empty, record(s) from all of the association and child tables are retrieved. To retrieve only the basic information (record from the main Info Center table only), a value of 'Primary' is used with this parameter. **Note** In case of Projects Info Center, the record detail parameter can have one of the following four values: ▪ **Empty ("")** — Data from all child/associated tables are retrieved but for only the top level project. ▪ **Primary** — Data from only the main Info Center table (PR) is retrieved for only the top level project. ▪ **AllPrimary** — Data from only the main Info Center table (PR) is retrieved for all WBS levels of the project. ▪ **All** — Data from all child/associated tables is retrieved for all WBS levels of the project.

Info Center Specific APIs to Retrieve Data

All records are returned in XML format complying with the schema for the particular Info Center. The 'RecordDetail' parameter determines if only the record from the main table is retrieved or if information from all child/association tables are also returned.

For more details regarding parameters used for these APIs, refer to the section above where generic APIs to retrieve data from Vision are described.

GetProjectsByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (PR.WBS1, PR.WBS2, PR.WBS3) value/values specified in 'keys' parameter.
GetProjectsByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.

GetClientsByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (CL.ClientID) value/values specified in ‘keys’ parameter.
GetClientsByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetContactsByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (Contacts.ContactID) value/values specified in ‘keys’ parameter.
GetContactsByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetEmployeesByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (EM.Employee) value/values specified in ‘keys’ parameter.
GetEmployeesByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetOpportunitiesByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (Opportunity.OpportunityID) value/values specified in ‘keys’ parameter.
GetOpportunitiesByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetVendorsByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (VE.Vendor) value/values specified in ‘keys’ parameter.
GetVendorsByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetLeadsByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (Leads.LeadID) value/values specified in ‘keys’ parameter.

GetLeadsByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetMktCampaignsByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (MktCampaign.CampaignID) value/values specified in 'keys' parameter.
GetMktCampaignsByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetTextLibrariesByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (TextLibrary.Name) value/values specified in 'keys' parameter.
GetTextLibrariesByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.
GetActivitiesByKey(ConnInfoXML, Keys, RecordDetail)	
Description	A set of records is retrieved based on the key (Activity.ActivityID) value/values specified in 'keys' parameter.
GetActivitiesByQuery(ConnInfoXML, Query, RecordDetail)	
Description	A set of records is retrieved based on the query.

APIs to Delete Info Center Records in Vision

For each of the Info Centers there is an API available to delete records. All delete APIs accept two parameters, similar to APIs used to add and update records. The two parameters used with delete APIs are listed below. The detailed descriptions of these parameters can be found in the earlier part of this document.

- **ConnInfoXML**. Vision login information that is used to authenticate calling application's access to Vision database.
- **DataXML**. Input data in XML format, in compliance with the supplied schema for the Info Center.

Unlike APIs used for adding and updating Info Center records, the DataXML for delete APIs are required to include only data for key fields for the main table of the Info Center. It is important that the 'tranType' (transaction type) attribute in the XML data is set to 'DELETE'. Also, data for only the main table of the Info Center is required,

If DataXML string includes the data for non-key fields and non-primary tables of the Info Center, it will be stripped off automatically before processing the data for deletion of records.

Return Message

If deletion is successful, a return value of '1' (as explained in the earlier part of this document) is returned. In the event of problems, appropriate error messages in the standard return-message XML format are returned.

List of Delete APIs

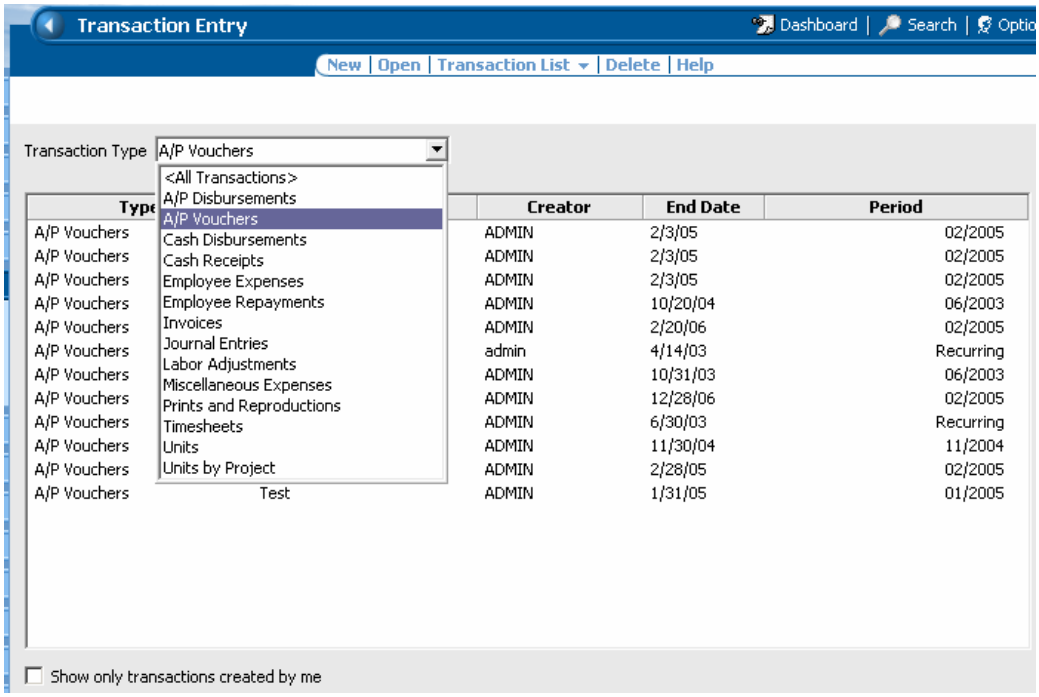
DeleteClient(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **CL** Key Field: **ClientID**
DeleteContact(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **Contacts** Key Field: **ContactID**

DeleteEmployee(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **EM** Key Field: **Employee**
DeleteProject(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **PR** Key Fields: **WBS1, WBS2, WBS3** **Notes** ▪ As specified in the schema, the 'keys' attribute needs to include values for all three key fields, separated by a semicolon. ▪ Individual keys are separated by a comma. ▪ Because Projects Info Center uses more than one key field and a default value of space is used for wbs2 and wbs3 fields when the corresponding level for the project does not exist, the list of keys should always end with a comma and spaces need to be included when applicable. For example, 0100000.00;02; ,0200000.00; ; ,
DeleteOpportunity(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **Opportunity** Key Field: **OpportunityID**
DeleteLead(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **Leads** Key Field: **LeadID**

DeleteCampaign(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **MktCampaign** Key Field: **CampaignID**
DeleteVendor(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **VE** Key Field: **Vendor**
DeleteTextLibrary(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **TextLibrary** Key Field: **Name**
DeleteActivity(ConnInfoXML, DataXML)	
Description	Records whose primary key information is included in DataXML are deleted from the Info Center. DataXML is required to include XML data for the table and the key fields listed below: Primary Table: **Activity** Key Field: **ActivityID**

Web APIs (Web Services) for Accounting Transactions

A complete set of Web APIs is available to add/edit/delete and post all types of accounting transactions in Vision. The supported transactions for APIs are as listed on the screen below. For each of the transaction types, there are APIs to add transaction entries, edit existing entries, delete the entire or part of unposted transaction entries, and finally to post transactions.



XML Schema for Vision Transaction Web APIs

The transaction data that needs to be added/updated/deleted or posted in Vision database is sent in XML format. The format of the XML data has to comply with the schema provided.

Each applicable transaction type in Vision has an XML schema defined and these schema files can be found on the Vision Web/app server in <InstallDir>\Vision\Web\Xsd directory (<InstallDir> is the directory where Deltek Vision is installed). The names of the schema files start with the name of the transaction type followed by '_Transaction_Schema.xsd'. For example, the name of the XML schema file used for A/P Vouchers type transactions would be 'APVouchers_Transaction_Schema.Xsd'.

Schema Structure

The schema for each transaction type maps to three entities in the database, based on the way transaction information is stored in the Vision database. These three entities share the common names Control, Master, and Detail, and they are prefixed with letters that indicate the type of transaction. For example, in case of A/P Voucher type of transactions, the names of the entities would be apControl, apMaster, and apDetail.

The Control entity tracks the primary details of the transaction file. The Master entity keeps track of different components of the transaction file such as invoices and checks. Finally, the Detail entity stores all of the details or line-items of the transaction.

In terms of the relationship between these entities, each transaction file is represented by a unique entry (transaction file) and the Control entity can have one or more entries/rows in the Master entity. Each record (e.g., invoice, check) in the Master entity can have one or more rows in the Detail entity.

Please refer to the Vision data dictionary for more information on fields/columns included in entities that are used for various types of transactions in Vision.

Sample Schema

The following is an example of a schema structure. This particular example shows the schema structure for A/P Voucher transaction in Vision.

Each transaction (file) information is included within a <REC> element. The XML data that is sent to Vision can have data for one or more transactions (files), each included in a separate <REC> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2006 sp2 U (http://www.altova.com)-->
<RECS xmlns="http://deltek.vision.com/XMLSchema"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://deltek.vision.com/XMLSchema
C:\DeltekVision\Vision41\Web\xsd\APVouchers_Transaction_Schema.xsd">
        <REC>
                <apControl keys="Batch" name="apControl" alias="apControl">
                        <ROW tranType="">
                                <Batch/>
                                <Recurring/>
                                <Posted/>
                                <Creator/>
                                <Period/>
                                <EndDate/>
                                <Total/>
                                <DefaultLiab/>
                                <DefaultBank/>
                                <DefaultDate/>
                                <DefaultTaxCode/>
                                <PostPeriod/>
                                <PostSeq/>
                                <Company/>
                                <DefaultCurrencyCode/>
                        </ROW>
                </apControl>
                <apMaster keys="Batch,MasterPKey" name="apMaster" alias="apMaster">
                        <ROW tranType="">
                                <Batch/>
```

```
                        <MasterPKey/>
                        <Vendor/>
                        <InvoiceDate/>
                        <Invoice/>
                        <TransDate/>
                        <LiabCode/>
                        <BankCode/>
                        <PayTerms/>
                        <PayDate/>
                        <Address/>
                        <Posted/>
                        <Seq/>
                        <Voucher/>
                        <CurrencyCode/>
                        <CurrencyExchangeOverrideMethod/>
                        <CurrencyExchangeOverrideDate/>
                        <CurrencyExchangeOverrideRate/>
                        <BarCode/>
                        <PaymentExchangeOverrideMethod/>
                        <PaymentExchangeOverrideDate/>
                        <PaymentExchangeOverrideRate/>
                </ROW>
        </apMaster>
        <apDetail keys="Batch,MasterPKey,PKey" name="apDetail" alias="apDetail">
                <ROW tranType="">
                        <Batch/>
                        <MasterPKey/>
                        <PKey/>
                        <Seq/>
                        <Description/>
                        <WBS1/>
                        <WBS2/>
                        <WBS3/>
                        <Account/>
                        <Amount/>
                        <SuppressBill/>
                        <TaxCode/>
                        <NetAmount/>
                        <TaxAmount/>
                        <CurrencyExchangeOverrideRate/>
                        <PONumber/>
                        <PaymentExchangeRate/>
                        <PaymentAmount/>
                        <PaymentExchangeInfo/>
                </ROW>
        </apDetail>
</REC>
```

```
</RECS>
```

Obtaining Schema Files for Transaction Areas

The latest schema files for Vision Web APIs can be obtained by using a Web service that returns schema file content (String), and the other by using Vision.

Using API

The following API can be used to obtain schema file content for an Info Center.

GetSchema(ConnInfoXML, TransactionArea)	
Description	Based on the value for 'TransactionArea' parameter for the call, content of the schema file respective to the transaction area is returned back.

The following parameter values can be used to retrieve schema files for various transaction areas:

Parameter Value	Transaction Area
APVouchers_Transaction	A/P Vouchers
APDisbursements_Transaction	A/P Disbursements
CashDisb_Transaction	Cash Disbursements
CashReceipts_Transaction	Cash Receipts
EmpExpense_Transaction	Employee Expenses
EmpRepayment_Transaction	Employee Repayments
Invoice_Transaction	Invoices
JournalEntry_Transaction	Journal Entry
LaborAdjust_Transaction	Labor Adjustments
Misc_Transaction	Miscellaneous Expenses
PrintsAndRepro_Transaction	Prints and Reproductions
Timesheet_Transaction	Timesheets
Unit_Transaction	Units
UnitByProject_Transaction	Units by project

List of Vision Transaction Web APIs

Each API for transactions take two parameters: One is the XML string with connection information (ConnXML) and the other is the XML string that contains the transaction data.

A/P Disbursements
AddAPDisbursementsTransaction(ConnXML, DataXML)
UpdateAPDisbursementsTransaction(ConnXML, DataXML)
DeleteAPDisbursementsTransaction(ConnXML, DataXML)
A/P Vouchers
AddAPVouchersTransaction(ConnXML, DataXML)
UpdateAPVouchersTransaction(ConnXML, DataXML)
DeleteAPVouchersTransaction(ConnXML, DataXML)
Cash Disbursements
AddCashDisbTransaction(ConnXML, DataXML)
UpdateCashDisbTransaction(ConnXML, DataXML)
DeleteCashDisbTransaction(ConnXML, DataXML)
Cash Receipts
AddCashReceiptsTransaction(ConnXML, DataXML)
UpdateCashReceiptsTransaction(ConnXML, DataXML)
DeleteCashReceiptsTransaction(ConnXML, DataXML)
Employee Expenses
AddEmpExpenseTransaction(ConnXML, DataXML)
UpdateEmpExpenseTransaction(ConnXML, DataXML)
DeleteEmpExpenseTransaction(ConnXML, DataXML)
Employee Repayments
AddEmpRepaymentTransaction(ConnXML, DataXML)

UpdateEmpRepaymentTransaction(ConnXML, DataXML)
DeleteEmpRepaymentTransaction(ConnXML, DataXML)
Invoices
AddInvoiceTransaction(ConnXML, DataXML)
UpdateInvoiceTransaction(ConnXML, DataXML)
DeleteInvoiceTransaction(ConnXML, DataXML)
Journal Entries
AddJournalEntryTransaction(ConnXML, DataXML)
UpdateJournalEntryTransaction(ConnXML, DataXML)
DeleteJournalEntryTransaction(ConnXML, DataXML)
Labor Adjustments
AddLaborAdjustTransaction(ConnXML, DataXML)
UpdateLaborAdjustTransaction(ConnXML, DataXML)
DeleteLaborAdjustTransaction(ConnXML, DataXML)
Miscellaneous Expenses
AddMiscTransaction(ConnXML, DataXML)
UpdateMiscTransaction(ConnXML, DataXML)
DeleteMiscTransaction(ConnXML, DataXML)
Prints and Reproductions
AddPrintsReproTransaction(ConnXML, DataXML)
UpdatePrintsReproTransaction(ConnXML, DataXML)
DeletePrintsReproTransaction(ConnXML, DataXML)
Timesheets
AddTimesheetTransaction(ConnXML, DataXML)
UpdateTimesheetTransaction(ConnXML, DataXML)
DeleteTimesheetTransaction(ConnXML, DataXML)

Units
AddUnitTransaction(ConnXML, DataXML)
UpdateUnitTransaction(ConnXML, DataXML)
DeleteUnitTransaction(ConnXML, DataXML)
Units by Project
AddUnitByProjectTransaction(ConnXML, DataXML)
UpdateUnitByProjectTransaction(ConnXML, DataXML)
DeleteUnitByProjectTransaction(ConnXML, DataXML)

Web APIs to Post Transactions

There is a single Web API to post transaction data. The API to post transactions takes the following parameters:

- **ConnXML**. Connection information in XML format.
- **TransType**. The type of accounting transaction data that is being posted (possible values are listed below).
- **BatchList**. The list of transaction files (identified by the batch number), and if there are multiple values, they need to be separated by commas.
- **Period**. The accounting period (as an integer) for which the transaction needs to be posted.

Values for TransType Parameter

Value	Transaction Type
AP	A/P Disbursements and A/P Vouchers
CD	Cash Disbursements
CR	Cash Receipts
ER	Employee Repayments
EX	Employee Expenses
IN	Invoices
JE	Journal Entries
LA	Labor Adjustments
MI	Miscellaneous Expenses

Value	Transaction Type
PR	Print and Reproductions
TS	Timesheets
UN	Units
UP	Units by Project

PostTransaction(ConnXML, TransType, BatchList, Period)

Transaction Security Validation

In addition to validation of login and access to transaction application-area, the following security rights are validated and used when processing transaction Web API calls. These security options are found on the Accounting tab in Configuration/Security/Roles area.

- **Record level security** — When the **Apply to All Transaction Center** option is selected, record-level security as specified for Info Centers is validated while processing transaction data. If any of the transactions sent through the API references Info Center records (for example, projects) that the Vision login/role does not have access to in the Info Center, those transaction entries will not be saved to the database.
- **Access to transaction types** — If the **Full access to all transaction types** option is not selected, access to transaction types as listed in the transaction-type grid is validated. Transaction data that is sent through APIs is processed and saved to the database only if the Vision login (used in ConnXML parameter for Web APIs) has access to respective transaction-types.
- **Transactions in closed periods** — When the **Allow Processing in Closed Periods** option is selected, the posting Web API allows postings in closed periods.
- **Transactions in prior periods** — When the **Allow Processing in Prior Periods** option is selected, the posting Web API allows postings in prior periods. The current period is the most recent period that has been opened and periods before the current period are considered as prior periods.

Transaction Data Validation

Vision accounting applications employ several data validation processes to validate data that is entered by the user and these validations are done either during the data entry or at the time of Save. All of the business logic that constitute data validation that is done during and after data entry in Vision, is also applicable when processing Web API calls to add/update/post transactions. These data validations include checks with respect to referential dependencies, record-level security, format of the data, data integrity, and access rights. Given the enormity of the details of specific data validation rules and logic, they are not individually listed in this section. The sections of application-specific online help may be used to understand validation rules in each of the accounting transaction areas.

Multicompany Support for Transactions

In Vision databases where multi-company module has been activated, the Web API calls to manipulate accounting transaction information are processed in the context of the company that is used in a given transaction. The company information included in XML for the Control entity (with reference to schema for a transaction-type) is used as the company for which the transaction is being processed and any business logic or data validation that is dependent on the company is processed in the context of the company information specified. For Vision databases where multi-company module is not in use, the company information in Control entity (with reference to the schema) can be set to single-space.

Transaction API Return Messages

The return messages in XML for transaction Web APIs follow the standard error message format applicable to all Web APIs for Vision in general.

Return Messages

The following table contains sample return messages with respect to various events and errors.

Event/Error and Error Code	Return message in XML
Successful API call **Return Code: 1**	<DLTKVisionMessage> <ReturnCode>1</ReturnCode> <ReturnDesc>Service has been successfully run</ReturnDesc> </Detail> </DLTKVisionMessage>
Schema validation error **Return Code:** **ErrSchemaVal**	<DLTKVisionMessage> <ReturnCode>ErrSchemaVal</ReturnCode> <ReturnDesc>Schema Validation Error</ReturnDesc> <Detail>Validation error on line 1 col 490 The 'http://deltek.vision.com/XMLSchema:suffix' element has an invalid value according to its data type. An error occurred at , (1, 490). ... Continuing Validation ... </Detail> </DLTKVisionMessage>
Vision login validation error **Return Code:** **ErrLoginVal**	<DLTKVisionMessage> <ReturnCode>ErrLoginVal</ReturnCode> <ReturnDesc>Login Validation Error</ReturnDesc> <Detail>Vision login failed, no data is saved to the database</Detail> </DLTKVisionMessage>

Event/Error and Error Code	Return message in XML
Vision record access rights error **Return Code:** **ErrRecAccess**	<DLTKVisionMessage> <ReturnCode>ErrRecAccess</ReturnCode> <ReturnDesc>Record Access Validation Error</ReturnDesc> <Detail> <AddRec> <Message>Permission to add new records is denied</Message> </AddRec> <UpdateRec> <Message>Permission to update records is denied</Message> <ReturnValues> <keyVal>Admin162326539399</keyVal> </ReturnValues> </UpdateRec> </Detail> </DLTKVisionMessage>

Event/Error and Error Code	Return message in XML
Vision data validation error **Return Code:** **ErrDataVal**	<DLTKVisionMessage> <ReturnCode>ErrDataVal</ReturnCode> <ReturnDesc>Data Validation Error</ReturnDesc> <Detail> <ValidationResult> <Message> <Table>leads</Table> <Column>status</Column> <ColValue>sysNew1</ColValue> <RowNum>1</RowNum> <PrimaryRecKeyVal> Admin162326539399</PrimaryRecKeyVal> <MsgCode>VAL1001</MsgCode> <MsgDesc>Value not in the master/code table!</MsgDesc> </Message> <Message> <Table>leads</Table> <Column>employee</Column> <ColValue>0002011</ColValue> <RowNum>2</RowNum> <PrimaryRecKeyVal> Admin162326539399</PrimaryRecKeyVal> <MsgCode>VAL1001</MsgCode> <MsgDesc>Value not in the master/code table!</MsgDesc> </Message> </ValidationResult> </Detail> </DLTKVisionMessage>

Event/Error and Error Code	Return message in XML
Save error (record exists) **Return Code:** **ErrSave**	<DLTKVisionMessage> <ReturnCode>ErrSave</ReturnCode> <ReturnDesc>Save Error</ReturnDesc> <Detail>Record already exists and cannot be added</Detail> </DLTKVisionMessage>
Add records (using 'Add' APIs) **Return Code: 1** (when successful)	<DLTKVisionMessage> <ReturnCode>1</ReturnCode> <ReturnDesc>Records Added</ReturnDesc> <Detail> <ReturnValues> <AddRec> <Row Number="1"> <KeyVal>Admin162326539399</KeyVal> </Row> <Row Number="2"> <KeyVal>Admin162326539799</KeyVal> </Row> </AddRec> </ReturnValues> </Detail> </DLTKVisionMessage>
Schema generation error **Return Code:** **ErrGenSchema**	<DLTKVisionMessage> <ReturnCode>ErrGenSchema</ReturnCode> <ReturnDesc>Custom Tabs Schema Generation Error</ReturnDesc> <Detail></Detail> </DLTKVisionMessage>

Event/Error and Error Code	Return message in XML
Get schema from server error **Return Code:** **ErrGetSchema**	<DLTKVisionMessage> <ReturnCode>ErrGetSchema</ReturnCode> <ReturnDesc>Get Schema File - Error</ReturnDesc> <Detail></Detail> </DLTKVisionMessage>
Error when retrieving records **Return Code:** **ErrGetRecs**	<DLTKVisionMessage> <ReturnCode>ErrGetRecs</ReturnCode> <ReturnDesc>Get Records - Data Retrieval Error</ReturnDesc> <Detail></Detail> </DLTKVisionMessage>
Error when validating data sent for deleting records **Return Code:** **ErrDeleteVal**	<DLTKVisionMessage> <ReturnCode>ErrDeleteVal</ReturnCode> <ReturnDesc>Delete Records - Input XML Error</ReturnDesc> <Detail></Detail> </DLTKVisionMessage>
Error when deleting records **Return Code:** **ErrDeleteRecs**	<DLTKVisionMessage> <ReturnCode>ErrDeleteRecs</ReturnCode> <ReturnDesc>Delete Records - Save Error</ReturnDesc> <Detail></Detail> </DLTKVisionMessage>

Integrating Web Services/API into an Application

Depending on the tool used for application development, one of the following three ways can be used to access Vision Web services from an external application.

- **Using WSDL file generated for VB**. The file DeltekVisionOpenAPIWebService.vb can be added to a VB.Net project and Web-services listed above in the document can be called using an instance of 'DeltekVisionOpenAPIWebService'.

  A sample excerpt of the code is shown below.

```
Dim returnMessage As String
Dim ConnInfoXML As String
Dim DataXML As String
Dim InfoCenter As String

'<Set values for variables here!>

Dim VisionWS As New DeltekVisionOpenAPIWebService

returnMessage = VisionWS.SendDataToDeltekVision(InfoCenter, ConnInfoXML, DataXML)

'<Parse and process returnMessage here!>
```

- **Using WSDL file generated for C# language**. The file DeltekVisionOpenAPIWebService.cs is available and it can be used with projects developed in C#.
- **Using a compiled DLL with the application**. WSDL file for VB or C# can be used to compile a DLL and it can be used with the calling application.

Security and Calls to Web Services

The Web accessible APIs (Web services) for Vision are accessed through the main VisionWS.asmx page deployed in <InstallDir>\Deltek\Vision\Web directory. The URL to this service page is embedded in the WSDL file. The calls to Web services use SOAP as the application-level protocol and use http as the transport protocol. The URL to the main service page can be found in the 'Constructor' in WSDL file.

For example, it would look like the following:

```
Public Sub New()
    MyBase.New
    Me.Url = "http://VisionServer/Vision/VisionWS.asmx"
End Sub
```

Since parameters for Web services are sent as plain text, there is an option to make secured calls to Web services using https instead of http. As long as the Web server supports secured https calls, Vision Web services can be accessed securely. In order to do it, change the URL above to use https instead of http.