

# LinkPlus Webservice API Reference

## Basic Methods

---

### GetIncidentByID

**public string GetIncidentByID**(string linkedsystemID, string linkedprojectID,  
int incidentID)

This method is used to get specific "Incident" based on some parameters.

#### Input Parameters:

**linkedsystemID:**This parameter indicates the qualified incident's system ID.

**linkedprojectID:**This parameter indicates the qualified incident's project ID.

**incidentID:**This parameter indicates the qualified incident's ID which you want to get.

#### Return Value: XML Data String

The returned string is the incident which you want to get in the xml format.

#### Sample XML string:

```
<Incident>
  <IncidentID>
    24
  </IncidentID>
  <AssignedByAccount>scott-w</AssignedByAccount>
  <AssignedByPerson>Scott Williams</AssignedByPerson>
  <ClosedByAccount/>
  <CloseStatus/>
  <CreatedByAccount>pamela-m</CreatedByAccount>
  <CreatedByPerson>Pamela Miller</CreatedByPerson>
  <CurrentOwner>Paul Wagner</CurrentOwner>
  <CurrentOwnerAccount>paul-w</CurrentOwnerAccount>
  <CustomPage1InXML/>
  <CustomPage2InXML/>
  <CustomPage3InXML/>
  <DateAssigned>2005-08-18T18:07:06</DateAssigned>
  <DateClosed>1970-01-01T08:00:00</DateClosed>
  <DateCreated>2005-08-02T18:06:10</DateCreated>
  <DateModified>2005-08-18T18:07:06</DateModified>
  <DescriptionPgField1Text>Bug (Discrepancy Report)</DescriptionPgField1Text>
  <DescriptionPgField2Text>1 - Build V1.0A</DescriptionPgField2Text>
  <DescriptionPgField3Text>2 - High</DescriptionPgField3Text>
  <DescriptionPgField4Text>Windows NT</DescriptionPgField4Text>
  <DescriptionPgField5Text>Other</DescriptionPgField5Text>
  <DescriptionPgField6Text>0</DescriptionPgField6Text>
  <DescriptionPgField7Text/>
  <DescriptionPgField8Text/>
  <IfClosed>0</IfClosed>
  <IsActive>1</IsActive>
  <NoOfHistories>4</NoOfHistories>
  <PlannedEndDate>1970-01-01T08:00:00</PlannedEndDate>
  <PlannedStartDate>1970-01-01T08:00:00</PlannedStartDate>
  <ProblemDescription>Test web service.</ProblemDescription>
  <ProgressStatus>QA Testing</ProgressStatus>
  <StatusPgField1Text>Beta</StatusPgField1Text>
  <StatusPgField2Text>cutom 1</StatusPgField2Text>
  <StatusPgField3Text/>
  <SubStatus>Testing</SubStatus>
  <Title>test by tang </Title>
  <WorkDescription>Test web service.</WorkDescription>
</Incident>
```

**Code Sample:**

```
String xmlIncident=""; //this statement initialize the xml //string to empty.
xmlIncident= wsIncident.GetIncidentByID(StrSystemID, StrProjectID,incidentID); //this statement get
the //incident.
```

**FindIncident**

**public string FindIncident**(string linkedsystemID,string linkedprojectID,string bstrOwner, int nDateType,

System.DateTime fromDate, System.DateTime toDate, string bstrStatus, int nFieldType, string bstrFieldValue, string bstrKeyWord, int nOpenStatus, int nSortOrder, int nSortType, string bstrCustomSearchCond, int pageSize, int currentPage)

This method is used to find incidents which fulfill the inputing conditions.

**Inuput Parameters:**

**linkedsystemID:** This parameter indicates the qualified incient’s system ID.

**linkedprojectID:** This parameter indicates the qualified incident’s project ID.

**bstrOwner:** This parameter indicates the qualified incident’s owner name.

The user name format should be “FirstName LastName” or “LastName, FirstName”.

**nDateType:** This parameter indicates the DateType which is used to search the incident.

Integer Value	Date Field
0	
1	Date Created
2	Date Closed
3	Date Assigned
4	Date Modified
5	Planned Start Date
6	Planned Finish Date

**fromDate:** This parameter specifies the starting date to be used for searching incidents.

**toDate:** This parameter specifies the ending date to be used for searching incidents.

DateTime type is XML standard date time type. Please refer W3C

<http://www.w3.org/TR/xmlschema-2/#dateTime>, The correct format is YYYY-MM-DD. If

you want to include time, the format should be YYYY-MM-DDTHH-MM-SS. There is a ‘T’ between date and time. For example, 2007-01-02 is right, 2007-01-02T01:02:03 is also

right, but 2007-1-2 or 2007-1-2T1:2:3 is wrong.

For Null value, please set nDateType to 0, then fromDate and toDate won’t be included.

**bstrStatus:** This parameter specifies the qualified incident’s processing status.

**nFieldType:** This parameter specifies which field to be used for searching incidents.

Integer Value	Field Number	Default label in Sample Project
1	1	Type
2	2	Priority
3	3	Component
4	4	Version
5	5	Platform

**bstrFieldValue:** This parameter specifies the field’s value,the field is used to search incidents.

**bstrKeyWord:** This parameter specifies the keyword value.If incident’s title or description contain the keyword ,the incident is qulified.

**nSortOrder:** This parameter specify the returning incidents’ order.

Integer Value	Sort Order
0	Ascending
1	Descending

**nOpenStatus:** This parameter specifies the qualified incident’s overall state.

Integer Value	Issue State
0	Open
1	Closed
>1	Open and Closed

**nSortType:** This parameter specifies the field ID that will be used to sort the returned incidents.

Integer Value	Field
0	IncidentID
1	Date Created
2	Date Assigned
3	Date Modified
4	Date Closed
5	“Type” (default label in sample project)
6	“Priority” (default label in sample project)
7	“Component” (default label in sample project)
8	“Version” (default label in sample project)
9	“Platform” (default label in sample project)

**bstrCustomSearchCond:** This parameter specifies custom search conditions.

This parameter should be used like this: “ columnName1=value1 AND columnName2=value2 OR columnName3=value3”. The same as WHERE CLAUSE without the “WHERE” key word.

Search condition is just a SQL sentence. Here is our source code:

```
if(!strCustomSearchCond.IsEmpty())
{
    strFilter += " AND " + strCustomSearchCond;
}
```

For datetime type in SQL statement, different database has different way. For example, SQL Server, you can just use ‘2007-1-2 1:2:3’ or ‘2007-1-2’. Please see database document for help.

**pageSize:** This parameter specifies the incident count each page can contain.

**currentPage:** The parameter specifies which page’s incidents should be returned.

**Return Value:** XML Data String

The returned string is the incidents which fulfill the input condition in the xml format.

**Sample returned XML string:**

```
<IncidentSet LinkedSystemID="2" ,LinkeProjectID="1",ProjectID=""4 >
<Incidents>
<Incident>
..
</Incident>
..
</Incidents>
</ IncidentSet>
```

**Using Code Sample:**

```
strxml = wsIncident.FindIncident(StrSystemID,StrProjectID, ownerName,
cmbSeachDateType.SelectedIndex,
    startTime, endTime, strStatus, 0, "", keyword, 2, 0, cmbSortBy.SelectedIndex, "", 10,
    pageIndex);
```

## FindIncidentID

**public string FindIncidentID**(string linkedsystemID,string linkedprojectID,string bstrOwner, int nDateType,

System.DateTime fromDate, System.DateTime toDate, string bstrStatus,  
int nFieldType, string bstrFieldValue, string bstrKeyWord, int nOpenStatus,  
int nSortOrder, int nSortType, string bstrCustomSearchCond)

This method is used to quick find incidents which fulfill the inputing conditions, the returned XML string only include the incident's ID

**Input Parameters:**

All is as same as method **FindIncident**, except that this method do not need **pageSize** and **currentPage**

**Return Value:** XML Data String

The returned XML string only include the XML node of the incident's ID which fulfill the input condition in the xml format.

**Sample returned XML string:**

```
<IncidentSet LinkedSystemID="2" ,LinkeProjectID="1",ProjectID=""4 >  
  
  <Incidents>  
    <Incident><IncidentID>1</IncidentID></Incident>  
    <Incident><IncidentID>2</IncidentID></Incident>  
    ..  
  </Incidents>  
</ IncidentSet>
```

**DeleteIncident**

**public int DeleteIncident**(string linkedsystemID,string linkedprojectID,int incidentID)

This method is used to delete incident.

**Input Parameters:**

**linkedsystemID:**This parameter indicates the qualified incident's system ID.

**linkedprojectID:**This parameter indicates the qualified incident's project ID.

**incidentID:**This parameter indicates the qualified incident's ID which you want to delete.

**Using Code Sample:**

```
this.wsIncident.DeleteIncident(StrSystemID, StrProjectID, Convert.ToInt32(id));
```

**AddNewIncident**

**public int AddNewIncident**(string linkedsystemID,string linkedprojectID,string xmlIncident)

This method is used to add new incident.

**Input Parameters:**

**linkedsystemID:**This parameter indicates the system ID.

**linkedprojectID:**This parameter indicates the project ID.

**xmlIncident:** This parameter contain the new incident's information to be added,The information is xml format.

**Return Value:**

int: The returned int is the newly added incident's ID.

**Input Sample XML String:** xmlIncident

```
<Incident>  
  <AssignedByAccount>scott-w</AssignedByAccount>  
  <AssignedByPerson>Scott Williams</AssignedByPerson>  
  <ClosedByAccount/>  
  <CloseStatus/>  
  <CreatedByAccount>pamela-m</CreatedByAccount>  
  <CreatedByPerson>Pamela Miller</CreatedByPerson>  
  <CurrentOwner>Paul Wagner</CurrentOwner>  
  <CurrentOwnerAccount>paul-w</CurrentOwnerAccount>  
  <CustomPage1InXML/>  
  <CustomPage2InXML/>  
  <CustomPage3InXML/>
```

```

<DateAssigned>2005-08-18T18:07:06</DateAssigned>
<DateClosed>1970-01-01T08:00:00</DateClosed>
<DateCreated>2005-08-02T18:06:10</DateCreated>
<DateModified>2005-08-18T18:07:06</DateModified>
<DescriptionPgField1Text>Bug (Discrepancy Report)</DescriptionPgField1Text>
<DescriptionPgField2Text>1 - Build V1.0A</DescriptionPgField2Text>
<DescriptionPgField3Text>2 - High</DescriptionPgField3Text>
<DescriptionPgField4Text>Windows NT</DescriptionPgField4Text>
<DescriptionPgField5Text>Other</DescriptionPgField5Text>
<DescriptionPgField6Text>0</DescriptionPgField6Text>
<DescriptionPgField7Text/>
<DescriptionPgField8Text/>
<IfClosed>0</IfClosed>
<IsActive>1</IsActive>
<NoOfHistories>4</NoOfHistories>
<PlannedEndDate>1970-01-01T08:00:00</PlannedEndDate>
<PlannedStartDate>1970-01-01T08:00:00</PlannedStartDate>
<ProblemDescription>Test web service.</ProblemDescription>
<ProgressStatus>QA Testing</ProgressStatus>
<StatusPgField1Text>Beta</StatusPgField1Text>
<StatusPgField2Text>cutom 1</StatusPgField2Text>
<StatusPgField3Text/>
<SubStatus>Testing</SubStatus>
<Title>test by tang </Title>
<WorkDescription>Test web service.</WorkDescription>
</Incident>

```

#### Using Code Sample:

```
xmlIncident=""// Init the xmlcident to a proper xml string which contain the new incident's information.
```

```
int intResult = wsIncident.AddNewIncident(StrSystemID, StrProjectID, xmlIncident);//the return value is the newly added incident's ID.
```

### EditIncident

```
public int EditIncident(string linkedsystemID,string linkedprojectID,string xmlIncident)
```

This method is used to edit an incident.

#### Input Parameters:

**linkedsystemID:**This parameter indicates the system ID.

**linkedprojectID:**This parameter indicates the project ID.

**xmlIncident:** This parameter must contain this incident's ID and other field which you want to change,this parameter's xml format is same as the function "AddNewIncident"

#### Return Value:

int: The returned value will be this incident's ID if it has been changed successfully, else will be 0.

#### Input Sample XML String: xmlIncident

```

<Incident>
  <IncidentID>1</IncidentID>
  <Title>New Title</Title>
  <WorkDescription>New Description</WorkDescription>
</Incident>

```

#### Using Code Sample:

```
xmlIncident=""// Init the xmlcident to a proper xml string which contain incident's ID and other new information.
```

```
int intResult = wsIncident.EditIncident(StrSystemID, StrProjectID, xmlIncident);//the return value will be this incident's ID if it has been changed successfully
```

---

## Issue Type Functions

### ReturnIssueType

**ReturnIssueType** (ProjectID, IssueID) – Will return a string value of the issue's current type (Issue 56 issue type: Enhancement Request [ID: 4])

**Return Value:** XML Data string.

```
XML Schema:  
<IssueTypeSet>  
<Response>  
<ID>int</ID>  
<ErrorMsg>string</ErrorMsg>  
</Response>  
<IssueTypes>  
<IssueType>  
<IssueTypeID>int</IssueTypeID>  
<IssueTypeName>string</IssueTypeName>  
</IssueType>  
</IssueTypes>  
</IssueTypeSet>
```

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

### ListIssueTypes

**ListIssueTypes**(ProjectID) – Will generate a list of all issue types and their IDs for the project.

**Return Value:** XML Data string.

XML Schema: same as ReturnIssueType function..

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

**SetIssueType** (ProjectID, IssueID, IssueTypeID, StateID) – Will set the issue to given issue type and place it into the given stateID.

**Return Value:** int type

1: successful

0: Failure.

-100: No data found.

---

## Subproject and Release Management Functions

### ExportSubprojectTree

**ExportSubprojectTree** (ProjectID) – Exports a list of the subproject tree in XML with their IDs

**Return Value:** XML Data string.

XML Schema: see as projecttree.xml

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

### **ExportProductVersionTree**

**ExportProductVersionTree** - Exports a list of the product version tree with IDs for products, versions, and builds, in XML.

**Return Value:** XML Data string.

XML Schema: see as ProductTree.xml

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

### **ExportApplicableProductVersionTree**

**ExportApplicableProductVersionTree**(ProjectID) – Exports a list of the product version tree with IDs for products, versions, and builds, in XML that are applicable to the given ProjectID

**Return Value:** XML Data string.

XML Schema: same as ExportProductVersionTree

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

### **SetIssueSubproject**

**SetIssueSubproject** (ProjectID, IssueID, SubprojectID) – Will throw an error if the subproject is invalid for the issue type (ie: someone tries to put a development issue into a build folder or a QA child issue into a development folder.)

**Return Value:** int type.

1: successful

0: Failure.

-100: No data found.

-95: can not put development issue to builder folder.

-94: can not put QA Issue to development folder.

---

### **CreateSubproject**

**CreateSubproject** (ProjectID, ParentID, SubprojectName) – Will create a subproject under the given parentID using the provided name.

**Return Value:** int type

1: successful

0: Failure.

-100: No data found.

---

### **CreateRMSubproject**

**CreateRMSubproject** (ProjectID, ParentID, SubprojectName) – Will create a RM subproject under the given parentID using the provided name. Duplicate names are not allowed.

**Return Value:** int type

1: successful

0: Failure.

-100: No data found.

-76: Duplicate name detected

-72: Can not creat RM subproject

-103: Can not support subproject

---

### **CreateUniqueSubproject.**

**CreateUniqueSubproject** (ProjectID, ParentID, SubprojectName) – Will create a subproject under the given parentID using the provided name. Duplicate names are not allowed.

**Return Value:** int type

- 1: successful
  - 0: Failure.
  - 100: No data found.
  - 76: Duplicate name detected
  - 103: Can not support subproject
- 

### **SetSubprojectProduct**

**SetSubprojectProduct**(ProjectID, SubprojectID,ProductID) – Sets the linked product to a subproject. Will throw an error if the subproject already has a product through inheritance.

**Return Value:**int type.

- 1: successful
  - 0: Failure.
  - 100: No data found.
  - 93: Product not found.
  - 92: Can't link Product to the subproject.
  - 91: Can't link a release management folder or build folder to a product.
- 

### **SetSubprojectVersion**

**SetSubprojectVersion**(ProjectID, SubprojectID,ProductID,VersionID) – Sets the linked version to a subproject. Will throw an error if the subproject already has a product through inheritance.

**Return Value:** int type.

- 1: successful
  - 0: Failure.
  - 100: No data found.
  - 93: No the product found.
  - 90: Don't link Version to the subproject.
  - 89: Don't link a release management folder or build folder to a version
  - 88: Product doesn't include the Version
  - 84: No the Version found.
- 

### **SetSubprojectBuild**

**SetSubprojectBuild** (ProjectID, SubprojectID, ProductID, VersionID, BuildID) – sets the linked build to a release subproject - will throw an error if used on a non-build subproject or if the subproject already has a build.

**Return Value:** int type.

- 1: successful
  - 0: Failure.
  - 100: No data found.
  - 93: No the product found.
  - 88: Product doesn't include the Version.
  - 87: Don't link build to the subproject.
  - 86: Don't link a release management folder to a build.
  - 85: Version doesn't include the build
  - 84: No the Version found.
  - 83: No the Build found.
-

# Functions for Release Management

## GetProductID

**GetProductID** (ProductName <string>) – Returns the ID of the Product based on Name

**Return Value:** XML Data string.

XML Schema:  
<ProductIDSet>  
<Response>  
<ID>int</ID>  
<ErrorMsg>string</ErrorMsg>  
</Response>  
<ProductID>  
<ID>int</ID>  
</ProductID>  
</ProductIDSet >  
ID: 0: successful  
-100: No data found.

**ErrorMsg:** the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

## GetVersionID

**GetVersionID** (ParentProductID <int>, VersionName <string>) – Returns the ID of the version based on Parent product and name

**Return Value:** XML Data string.

XML Schema: same as GetProductID function..

ID: 0: successful

-100: No data found.

**ErrorMsg:** the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

## GetBuildID

**GetBuildID** (ParentVersionID<int>,BuildName <string>)- Returns the ID of the build based on Parent version and name

**Return Value:** XML Data string.

XML Schema: same as GetProductID function..

ID: 0: successful

-100: No data found.

**ErrorMsg:** the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

## AddVersion

**AddVersion** (ParentProductID <int>, VersionName <string>) – Adds a version to the release tree under the product.

**Return Value:** int type

>0: successful, and return new version id

0: Failure.

-78: can't add version.

-93: no product found and meaning the parent product is not applicable.

-76: the name already exists.

---

## AddBuild

**AddBuild** (ParentVersionID<int>, BuildName <string>) – Adds a build to release tree under the version.

**Return Value:** int type

>0: successful, and return new build id

0: Failure.

-77: can't add build.

-84: no version found and meaning the parent version is not applicable.

-76: the name already exists.

---

### **SetApplicableProduct**

**SetApplicableProduct** (ProductID<int>,Applicable Projects <array>) – Sets the product applicable to given projects.

**Return Value:** int type

1: successful

0: Failure.

-93: no product found.

-74: no family found and meaning the parent family is not applicable.

---

### **SetApplicableVersion**

**SetApplicableVersion** (VersionID <int>, Applicable Projects <array>) – Sets the versions applicable to the given projects. Returns an error if the parent product is not applicable.

**Return Value:** int type

1: successful

0: Failure.

-84: no version found

-93: no product found and meaning the parent product is not applicable.

-74: no family found and meaning the parent family is not applicable.

---

### **SetApplicableBuild**

**SetApplicableBuild** (BuildID <int> , , Applicable Projects <array>) – Sets the builds applicable to the given projects. Returns an error if the parent version is not applicable.

**Return Value:** int type

1: successful

0: Failure.

-83: no build found.

-84: no version found and meaning the parent version is not applicable.

-93: no product found and meaning the parent product is not applicable.

-74: no family found and meaning the parent family is not applicable.

---

### **AddNewQAIncident()**

**Used to create a new QA issue from an existing development issue**

**Input Parameters:**

linkedSystemID – Follow other interface

linkedProjectID – Follow other interface

ParentDevIssueID – The ID for the parent development issue. Must be a development issue

ReleaseLinkTypeID – The ID of the release link type. Must be a release link type

QAIssueTypeID - The issue type of target QA issue. Must be a QA issue type

TargetBuildSubprojectID - The target build subproject for the target QA issue. Must be a build subproject.

NewWorkflowStateID - The workflow state ID for the new QA issue. If 0, the workflow state of new QA issue will be same as the original development issue

NewOwnerID - The owner's user ID for the new QA issue. If 0, the owner will be same as the original development issue

**Return Value: int type**

>0: successful, The New QA issue ID

-100: The LinkedSystemID and LinkedprojectID is not valid

-71: The ParentDevIssueID is not existed or is not a development issue.

-75: The ReleaseLinkTypeID is not existed or is not a multi-release link

-70: The QAIssueTypeID is not existed or is not a QA issue type.

-69: The SubprojectID is not existed or is not a build subproject.

-68: Invalid Workflow State ID

-67: Invalid User ID

---

## ListLinkedQAIssues

List all the QA issues linked to the specified Dev issues in a particular build subproject

### Input Parameters:

linkedSystemID – Follow other interface  
linkedProjectID – Follow other interface  
DevIssueID – the development issue ID  
BuildSubprojectID – The target build subproject ID

### Return Value: XML Data string.

```
XML Schema:  
<QAIssueSet>  
  <Response>  
    <ID>int</ID>  
    <ErrorMsg>string</ErrorMsg>  
  </Response>  
  <QAIssues>  
    <QAIssue>  
      <IncidentID>int</IncidentID >  
      <Title>string</Title >  
      <CurrentOwner >string</ CurrentOwner>  
      <ProgressStatus >int</ ProgressStatus>  
    </QAIssue>  
  </QAIssues>  
</QAIssueSet>
```

ID                      ErrorMsg:  
>=0: (total number)  
-100:                    “Can’t connect to the database.”  
-71:                     “The DevIssueID is not existed or is not a development issue.”  
-69:                     “The subproject must be a build subproject.”

---

## Issue Linking Functions

### GetLinkTypeID

**GetLinkTypeID** (ProjectID <int>, LinkTypeName <string> [LinkedProjectID <int>]) – Returns a link type ID based on the link type name – optionally can a linked project ID for InterProject links.

**Parameter:** if [LinkedProjectID <int>] set to 0, it means no linkedprojectID parameter needed.

**Return Value:** XML Data string.

XML Schema: same as GetProductID function..

ID: 0: successful

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

### LinkIssue

**LinkIssue** (ProjectID <int>, IssueID <int>, LinkedProjectID <int>, LinkedIssueID <int>, LinkTypeID <int>) – Links the two issues with the given link type. Returns errors if its an invalid link type.

**Return Value:** int type

1: successful

0: Failure.

-100: No issue found.

-75: No link type found

-73: The issue link already exists.

---

## File Attachment Functions

### AddNewNote

#### Adds a new note with file attachment to an existing issue

##### Input Parameters:

linkedSystemID	– Follow other interface
linkedProjectID	– Follow other interface
nIncidentID	– The issue ID of the issue you want to add note
NoteTitle	– Title for the new note.
NoteDescription	– Detail description of the new note.
FileName:	– The full path of the file you want to attach to.
UserName	– The user whom you specify to be the creator of the note.

##### Return Value: int type

>0:	successful, The New Note ID
-100:	The LinkedSystemID and LinkedprojectID is not valid
0:	failure

---

### AddNewNoteWithAttachment()

#### Adds a new note with an attached HTML path to an existing issue

##### Input Parameters:

linkedSystemID	– Follow other interface
linkedProjectID	– Follow other interface
nIncidentID	– The issue ID of the issue you want to add note
NoteTitle	– Title for the new note.
NoteDescription	– Detail description of the new note.
AttachmentOption	– 0: No attachment, the parameter 'AttachmentPath' and 'AttachContent' will be ignored 1: Attach a file, the parameter 'AttachmentPath' can not be empty and will be explained as the save as file name, the 'AttachContent' can not be empty. 2: Attach a HTML path, the parameter 'AttachmentPath' can not be empty and will be explained as a HTML path, the 'AttachContent' will be ignored
AttachmentPath	–The save as file name or the HTML path.
UserName	– The user whom you specify to be the creator of the note.
AttachContent	– This parameter's type is byte [], represent the file content's byte array.

##### Return Value: int type

>0:	successful, The New Note ID
-100:	The LinkedSystemID and LinkedprojectID is not valid
0:	failure

##### Note:

The AttachContent should be less than 10 MB, otherwise it may an HTTP runtime error may occue.

You can Configures ASP.NET HTTP run-time settings in web.config file, LinkPlus WebService have set is as:

```
<httpRuntime executionTimeout="300" maxRequestLength="102400" />
```

---

### AddNewNoteWithAttachmentEx()

#### Adds a new note with file attachment or HTML path to an existing issue with a note type

**Input Parameters:**

linkedSystemID – Follow other interface  
 linkedProjectID – Follow other interface  
 nIncidentID – The issue ID of the issue you want to add note  
 NoteTypeID – The NoteType ID of the new note  
 NoteTitle – Title for the new note.  
 NoteDescription – Detail description of the new note.  
 AttachmentOption – 0: No attachment, the parameter ‘AttachmentPath’ and ‘AttachContent’ will be ignored  
                           1: Attach a file, the parameter ‘AttachmentPath’ can not be empty and will be explained as the save as file name, the ‘AttachContent’ can not be empty.  
                           2: Attach a HTML path, the parameter ‘AttachmentPath’ can not be empty and will be explained as a HTML path, the ‘AttachContent’ will be ignored  
 AttachmentPath –The save as file name or the HTML path.  
 UserName – The user whom you specify to be the creator of the note.  
 AttachContent – This parameter’s type is byte [], represent the file content’s byte array.

**Return Value: int type**

>0: successful, The New Note ID  
 -100: The LinkedSystemID and LinkedprojectID is not valid  
 0: failure

Note:

The AttachContent should be less than 10 MB, otherwise it may an HTTP runtime error may occue.

You can Configures ASP.NET HTTP run-time settings in web.config file, LinkPlus WebService have set is as:

```
<httpRuntime executionTimeout="300" maxRequestLength="102400" />
```

---

**ListNoteTypes()**

**Used to list all the Note Types name and ID**

**Input Parameters:**

linkedSystemID – Follows other interfaces  
 linkedProjectID – Follows other interface

**Return Value: XML Data string.**

XML Schema:

```

<NoteTypeSet>
  <Response>
    <ID>int</ID>
    <ErrorMsg>string</ErrorMsg>
  </Response>
  <NoteTypes>
    <NoteType>
      <NoteTypeID>int</NoteTypeID>
      <NoteTypeName>string</NoteTypeName >
    </ NoteType >
  </NoteTypes >
</NoteTypeSet >

```

ID

ErrorMsg:

0:

“No Note Type found”

>0 (total number)  
-100: "Can't connect to the database."

---

## Administrative Functions

### ReturnUser()

Returns the information of a user

**Input Parameters:**

linkedSystemID – Follow other interface  
linkedProjectID – Follow other interface  
UserID – the userID

**Return Value: XML Data string.**

```
XML Schema:  
< UserSet >  
  < Response >  
    < ID >int </ID>  
    < ErrorMsg >string </ErrorMsg>  
  </ Response >  
  < Users >  
    < User >  
      < UserID >int </ UserID >  
      < UserLoginName >string </ UserLoginName >  
      < UserFullName >string </ UserFullName >  
      < UserStatus>int </ UserStatus>  
    </ User >  
  </ Users >  
</ UserSet >
```

ID                      ErrorMsg:  
0:                        "No user found"  
>0 (total number)  
-100:                    "Can't connect to the database."

---

### ListUsers

Returns all the project members within current project

**Input Parameters:**

linkedSystemID – Follow other interface  
linkedProjectID – Follow other interface  
UserStatus – 0:Inactive users, 1:Active Users, 2 or others except 0 and 1: All users

**Return Value: XML Data string.**

XML Schema: same as ReturnUser function..

ID                      Error Msg:  
0:                        "No user found"  
>0 (total number)  
-100:                    "Can't connect to the database."

---

### SetIssueState

Sets the workflow status for a series of issues

**Input Parameters:**

linkedSystemID – Follow other interface  
linkedProjectID – Follow other interface  
IssueIDList – A string of issue ID list , sample: 1, 43, 25, 46  
WorkflowStateID – The workflow state ID

**Return Value: XML Data string.**

```
XML Schema:  
<Response>  
  <ID>int</ID>  
  <ErrorMsg>string</ErrorMsg>  
</Response>
```

ID	ErrorMsg:
0:	“Updated issue ID: xx,xx.”
-100:	“Can’t connect to the database.”
-71:	“Updated issue ID: xx,xx. Invalid issue ID: xx,xx”
-68:	“The WorkflowStateID is invalid.”

---

## SetIssueOwner

### Sets the Owner for a series of issues

**Input Parameters:**

linkedSystemID – Follow other interface  
linkedProjectID – Follow other interface  
IssueIDList – A string of issue ID list: 1,43,25,46  
UserID – The user ID,  
          or the GroupfolderID if UserType = 1  
          or the teamgroupID if UserType = 2  
UserType – The type of user  
          0 or others except 1 and 2 – general user  
          1 – group folder,  
          2 – team group

Notes: similar with DevTrackClient, we can assign the issue to these users:

{ Unassigned }:	UserType = 0, User ID = -1
{ Submitter }:	UserType = 0, User ID = -2
{ Auto-Assignment }:	UserType = 0, User ID = -3
{ Previous Owner }:	UserType = 0, User ID = -5

Once group folder is enabled for this project, the owner can not be set to a team group. Only when group folder is disabled, current owner can be set to team group. This is follow current DevTrack Client behavior.

Users can select { Auto-Assignment } only when auto routing is enabled in the Admin

**Return Value: XML Data string.**

```
XML Schema:  
<Response>  
  <ID>int</ID>  
  <ErrorMsg>string</ErrorMsg>  
</Response>
```

ID	ErrorMsg:
0:	“Updated issue ID: xx,xx.”
-100:	“Can’t connect to the database.”
-71:	“Updated issue ID: xx,xx. Invalid issue ID: xx,xx”
-67:	“The UserID is invalid.”
-66:	“This project don’t support Group Folder or the FolderID is invalid”
-65:	“This project don’t support Team Group or the TeamGroupID is invalid”
-64:	“Auto assignment is not enabled in this project”

---

## ListGroupFolders()

Returns a string value of the group folder name and its ID

**Input Parameters:**

linkedSystemID – Follow other interface  
 linkedProjectID – Follow other interface

**Return Value: XML Data string.**

XML Schema:

```
<GroupFolderSet>
<Response>
  <ID>int</ID>
  <ErrorMsg>string</ErrorMsg>
</Response>
<GroupFolders>
  <GroupFolder>
    <GroupFolderID>int</GroupFolderID >
    <GroupFolderName >string</GroupFolderName >
    <OwnerGroupID >int</OwnerGroupID>
  </GroupFolder>
</GroupFolders >
</GroupFolderSet>
```

ID                                      ErrorMsg:  
 0:                                      “No Group Folder found”  
 >0 (total number)  
 -100:                                  “Can’t connect to the database.”

---

**ListTeamGroups()****Returns a string value of the team group name and its ID****Input Parameters:**

linkedSystemID – Follow other interface  
 linkedProjectID – Follow other interface

**Return Value: XML Data string.**

XML Schema:

```
<TeamGroupSet>
<Response>
  <ID>int</ID>
  <ErrorMsg>string</ErrorMsg>
</Response>
<TeamGroups>
  <TeamGroup>
    <TeamGroupID>int</TeamGroupID>
    <TeamGroupName >string</TeamGroupName>
  </TeamGroup>
</TeamGroups>
</TeamGroupSet>
```

ID                                      ErrorMsg:  
 0:                                      “No Team group found”  
 >0 (total number)  
 -100:                                  “Can’t connect to the database.”

---

**ListIssuesInSubproject****Lists all issues within a subproject recursively****Input Parameters:**

linkedSystemID – Follow other interface  
 linkedProjectID – Follow other interface  
 SubprojectID – The target subproject ID  
 IssueType – issue type 0 – all issues, 1- development issues, 2 - QA issues

**Return Value: XML Data string.**

XML Schema:

```
<IssueSet>
  <Response>
    <ID>int</ID>
    <ErrorMsg>string</ErrorMsg>
  </Response>
  <Issues>
    <Issue>
      <IncidentID>int</IncidentID >
      <Title>string</Title >
      <CurrentOwner>string</CurrentOwner>
      <ProgressStatus>int</ProgressStatus>
      <SubProjectID>int</SubProjectID>
      <IssueType>int</IssueType >
    </ Issue>
  </ Issues>
</IssueSet>
```

ID	ErrorMsg:
0:	“no issue found”
>0 ( <b>total number</b> )	
-100:	“Can’t connect to the database.”
-69:	“Invalid subproject ID”

---

### SetIssueField()

**Sets the value of a field for one of more issues.**

**Input Parameters:**

- linkedSystemID – Follow other interface
- linkedProjectID – Follow other interface
- IssueIDList – A string of issue ID list 1,43,25,46
- FieldName – The name of the field, should be consistent as the XML tag name in the function GetIncidentByID
- Value – The value to be set

**Return Value: XML Data string.**

```
XML Schema:
<Response>
  <ID>int</ID>
  <ErrorMsg>string</ErrorMsg>
</Response>
```

ID	ErrorMsg:
0	“Updated issue ID: xx,xx.”
-100:	“Can’t connect to the database.”
-63:	“Invalid field name”
-71:	“Updated issue ID: xx,xx. Invalid issue ID: xx,xx”

Notes:

1. This fuction is similar with EditIncident, These field name should be consistent with the XML tag name.
  2. These fields are reserved and can not be modified: IncidentID, AttachmentFileName, CreatedByPerson.
- 

### GetIncidentData()

**Returns the specified fields information of a set of issues**

**Input Parameters:**

- linkedSystemID – Follow other interface
- linkedProjectID – Follow other interface

IssueIDList – A string of issue ID list: 1,43,25,46

FieldNameList – list of fields to be returned

Such as: IncidentID, AssignedByAccount, AssignedByPerson

These field name should be consistent as the XML tag name in the function  
GetIncidentByID()

**Return Value: XML Data string.**

XML Schema:

```
<IncidentSet>
  <Response>
    <ID>int</ID>
    <ErrorMsg>string</ErrorMsg>
  </Response>
  <Incidents>
    <Incident>
      <IncidentID>int</IncidentID>
      <AssignedByAccount>string</AssignedByAccount>
      <AssignedByPerson>string</AssignedByPerson>
      ...
    </ Incident>
  </ Incidents>
</ IncidentSet>
```

ID	ErrorMsg:
0:	“no issue found”
>0(total number)	
-100:	“Can’t connect to the database.”
-71:	“Valid issue ID: xx,xx. Invalid issue ID: xx,xx.”
-63:	“Invalid field name: xx,xx.”

Notes:

1. Whether the FieldnameList contains “IncidentID”, the return string will always contain the key node of <IncidentID>.
  2. If the FieldnameList contains an invalid field name, it will return -64 and the IncidentSet will be empty.
  3. If the IssueIdList contains invalid issueID, will return -71 and the Incidentset only contains the valid issues.
-

## Workflow Functions

### ReturnState (ProjectID<int>, StateID<int>)

Will return a string value of the state name and its ID

**Return Value:** XML Data string.

```
XML Schema:
<ProgressStateSet>
<Response>
  <ID>int</ID>
  <ErrorMsg>string</ErrorMsg>
</Response>
<ProgressStates>
  <ProgressState>
    <ProgressStateID>int</ ProgressStateID >
    <ProgressStateName>string</ ProgressStateName >
  </ ProgressState >
</ ProgressStates >
</ ProgressStateSet >
```

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

### ListStates(ProjectID<int>)

Will generate a list of all progress states and their IDs for the project.

**Return Value:** XML Data string.

XML Schema: same as ReturnState function..

ID: 0: successful

1: Can't connect to Database

-100: No data found.

ErrorMsg: the message info (such as successful or not, DB error message etc) returned from the function being invoked

---

## User Authentication Functions

**AuthenticateUser ()** – Find out whether the inputted user is a valid and active

**DevTrack user**

**Input Parameters:**

linkedsystemID: This parameter indicates the system ID.

linkedprojectId: This parameter indicates the project ID.

userName: Inputted user's login name.

passWord: Inputted user's password.

**Return Value:** int

0 - Authentication failure

1 - Authentication successful

---

# Appendix

## Common Error IDs in return XML

- 101 'The project does not support release management',
- 100 'No data found',
- 95 'Cannot put development issue into build folder',
- 94 'Cannot put QA issue in development folder',
- 93 'No product found',
- 92 'Unable to link product to subproject',
- 91 'Unable to release management folder or build folder to a product',
- 90 'Unable to link version to subproject',
- 89 'Unable to link a release management folder or build folder to a version',
- 88 'Product does not include the version',
- 87 'Cannot link build to the supproject',
- 86 'Cannot link a release management folder to a build',
- 85 'Version does not include a build',
- 84 'Version not found',
- 83 'Build not found',
- 81 'Unable to set QA test issue type',
- 78 'Unable to add version',
- 77 'Unable to add build',
- 76 'The name already exists',
- 75 'No link type found',
- 74 'No family found',
- 73 'The issue link already exists',
- 72 'Cannot create a release management folder in a version folder',
- 71 'Invalid issue ID'
- 70 'Invalid QA issue Type ID'
- 69 'Invalid subproject ID'
- 68 'Invalid workflow state ID'
- 67 'Invalid user ID'
- 66: 'This project don't support Group Folder or the FolderID is invalid'
- 65: 'Invalid team group ID'
- 64: 'Auto assignment is not enabled for this project'
- 63: 'Invalid field name'
- 62: 'Invalid value'