**Keylight API Basics**

|  |  |
| --- | --- |
| **SecurityServices Call** | **Description** |
| Login | Accepts an account username and password, verifies them within Keylight and provides an encrypted cookie that can be used to authenticate additional API transactions. |
| Ping | Refreshes a valid Keylight Platform session. |
| Logout | Terminates a Keylight Platform session. |
| GetUser | Returns all fields for a given user. |
| GetUsers | Returns users and supporting fields. Filters may be applied to return only the users meeting selected criteria. |
| GetUserCount | Returns a count of users. Filters can be applied to return only the users meeting selected criteria. |
| CreateUser | Create a user account. |
| UpdateUser | Update a user account. |
| DeleteUser | Delete a user account. |
| GetGroup | Returns all fields for a given group. |
| GetGroups | Returns the ID and Name for groups. A filter may be applied to return only the groups meeting selected criteria. |
| CreateGroup | Creates a group. |
| UpdateGroup | Updates a group. |
| DeleteGroup | Delete a group. |

|  |  |
| --- | --- |
| **ComponentServices Call** | **Description** |
| GetComponent | Retrieves a component specified by its ID. A component is a user-defined data object such as a custom content table. The component ID may be found by using GetComponentList. |
| GetComponentList | Returns a complete list of all Keylight components available to the user based on account permissions. No input elements are used. The list will be ordered in ascending alphabetical order of the component name. |
| GetComponentByAlias | Retrieves a component specified by its Alias. A component is a user-defined data object such as a custom content table. The component Alias may be found by using GetComponentList (ShortName). |
| GetField | Retrieves details for a field specified by its ID. The field ID may be found by using GetFieldList. |
| GetFieldList | Retrieves detail field listing for a component specified by its ID. The component ID may be found by using GetComponentList. Documents or Assessments field types will not be visible in this list. |
| GetAvailableLookupRecords | Retrieves records that are available for population for a lookup field. |
| GetLookupReportColumnFields | Gets the field information of each field in a field path that corresponds to a lookup report column. The lookupFieldId corresponds to a lookup field with a report definition on it and the fieldPathId corresponds to the field path to retrieve fields from, which is obtained from GetDetailRecord.  GetLookupReportColumnFields compliments GetRecordDetail by adding additional details about the lookup report columns returned from GetRecordDetail. |
| GetRecord | Returns the complete set of fields for a given record within a component. |
| GetRecords | Returns the title/default field for a set of records within a chosen component. Filters may be applied to return only the records meeting selected criteria. |
| GetRecordCount | Returns the number of records in a given component. Filters may be applied to return the count of records meeting a given criteria. This function may be used to help determine the amount of records before retrieving the records themselves. |
| GetDetailRecord | Retrieves record information based on the provided component ID and record ID, with lookup field report details. Lookup field records will detail information for fields on their report definition, if on is defined. |
| GetDetailRecords | GetDetailRecords provides the ability to run a search with filters and paging (GetRecords) while returning a high level of detail for each record (GetRecord). GetDetailRecords also allows multiple sorts to modify the order of the results. For performance and security concerns, the maximum number of records returned (pageSize) is 1000. |
| GetRecordAttachment | Gets a single attachment associated with the provided component ID, record ID, documents field ID, and document ID. The file contents are returned as a Base64 string. |
| GetRecordAttachments | Gets information for all attachments associated with the provided component ID, record ID, and Documents field id. No file data is returned, only file name, field ID, and document ID information. |
| GetWorkflow | Retrieves workflow details and all workflow stages specified by ID. The ID for a workflow may be found by using GetWorkflows. |
| GetWorkflows | Retrieves all workflows for a component specified by its Alias. A component is a user-defined data object such as a custom content table. The component Alias may be found by using GetComponentList (ShortName). |
| TransitionRecord | Transition a record in a workflow stage. |
| VoteRecord | Cast a vote for a record in a workflow stage. |
| CreateRecord | Creates a new record within the specified component of the Keylight application. Note that the Required option for a field in Keylight is only enforced through the user interface, not through the API. Therefore, CreateRecord does not enforce the Required option for fields. |
| UpdateRecord | Updates fields within a specified record. Note that the Required option for a field in the Keylight Platform is only enforced through the user interface, not through the API. Therefore, UpdateRecord does not enforce the Required option for fields. The response will include the complete set of fields for the given record. |
| UpdateRecordAttachments | Adds new attachments and/or updates existing attachments to the provided Documents field(s) on a specific record, where the FileData is represented as a Base64 string. The maximum data size of the request is controlled by the maxAllowedContentLength and maxReceivedMessageSize values in the API web.config. |
| ImportFile | Queue a job to import a file for a defined import template. |
| Issue Assessments | Assessments can be initiated via the API into fields on DCF tables and on Master Detail records. Assessments require specific data to be issued appropriately via a Request XML file. |
| DeleteRecord | Deletes a selected record from within a chosen component. DeleteRecord will update the record, making it so it will no longer be viewable within Keylight.  Records are soft-deleted to maintain any historical references to the record and can be restored with a database script. |
| DeleteRecordAttachments | Deletes the specified attachments from the provided document fields on a specific record. |

**Rules to Keylight API**

The Keylight API implements RESTful web services using:

Base URL http://[instance name]:[port]/ComponentService/GetComponent?id={ID}

HTTP verbs  GET

* POST
* DELETE

Responses XML

JSON

**Access Configuration**

All API requests are made using the base URL.

All Logins will be based on the following structure:

* http[s]://<instance\_name>:<port>/SecurityService/Login
* All data gathering/manipulation requests will be made through
* http[s]://<instance\_name>:<port>/ComponentService/<operation>

The instance name describes the URL of the Keylight installation. The port is determined by the individual configuration of the site installation.

See http[s]://<machine\_name>:<port>/SecurityService/help for the published list of call templates.

**XML/JSON Payload**

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Permissions | Permissions for the API are determined by the logon credentials used to access it. API access uses the permission settings for an account configured within the Security Roles section of the Keylight Platform. Users must purchase a subscription to the API. For more information, contact your account manager.  The credentials are accessed using the Login function in the SecurityService API. The credentials return a cookie that is then passed as authorization to all of the other portions of the API and is detailed in the SecurityService API. |
| cURL | While the REST Framework supports almost any development language, examples in this documentation are provided in cURL and can be executed in a standalone command prompt. cURL is a tool to transfer data from or to a server, using HTTP(s) to connect to the Keylight web service API and return an XML response that can be stored and parsed. Additional information about cURL can be found at curl.haxx.se. For more information on key switches, see [cURL](#_bookmark50) [Switches](#_bookmark50) in the appendix. |
| Data Structure | A key component of the Keylight architecture is the Dynamic Content Framework (DCF) which allows customers to build customized, dynamic tables to store a variety of data elements. The DCF also allows for complex relational interaction between custom and permanent Keylight data elements. As such, the content is completely customized to each business and does not have a fixed structure. Some terms will be useful to understand when using the API to access custom content.  Component A single table in the Keylight Platform. For example, a Risks table.  Field A definition for a single piece of information in a component, for example, the address of a building. Each field is restricted to a user- defined data type. For more information on the field types, see Field Types in the appendix.  Record A complete grouping of multiple fields for a single identifier. For example, a record may consist of a name, address, and phone number for an employee. |

2: Security Services API

This chapter includes a description of the functions for logging in and logging out API calls and methods for getting, creating, updating, and deleting users and groups.

[Security Services API](#_bookmark5) [12](#_bookmark5)

[Login](#_bookmark6) [13](#_bookmark6)

[Ping](#_bookmark7) [15](#_bookmark7)

[Logout](#_bookmark8) [16](#_bookmark8) [GetUser](#_bookmark9) [17](#_bookmark9) [GetUsers](#_bookmark10) [20](#_bookmark10) [GetUserCount](#_bookmark11) [25](#_bookmark11) [CreateUser](#_bookmark12) [28](#_bookmark12) [UpdateUser](#_bookmark13) [35](#_bookmark13) [DeleteUser](#_bookmark14) [40](#_bookmark14) [GetGroup](#_bookmark15) [41](#_bookmark15) [GetGroups](#_bookmark16) [43](#_bookmark16) [CreateGroup](#_bookmark17) [46](#_bookmark17) [UpdateGroup](#_bookmark18) [49](#_bookmark18) [DeleteGroup](#_bookmark19) [52](#_bookmark19)

**Security Services API**

SecurityService Login function generates an encrypted cookie in the return header. This cookie must be captured and used as a parameter in all ComponentService calls to verify identity and set permissions.

Permissions for the data that the API can access are based on permissions of the Logon account that is used. For example, if the login account has read-only permissions to an asset table, the API returns data for user viewing but does not allow update or delete functions to be performed.

**Login**

Accepts an account username and password, verifies them within Keylight and provides an encrypted cookie that can be used to authenticate additional API transactions.

URL: http://[instance-name]:[port]/SecurityService/Login Method: POST

Input: Username (Text) Username for the Keylight application account Password (Text) Password for the Keylight application account

Output: A cookie to establish sessions within the component services API.

Permissions: The account that is used to log into the application must have access to the Keylight API. Users must also have appropriate permissions for any data they wish to access or manipulate.

**Examples**

In this example, cURL uses the –c option to store the encrypted cookie that is returned in a file. The cookie can be sent to authenticate all data manipulation commands in the ComponentService API with a single cURL command line switch (-b). The response sample includes both the header of the response (to view the cookie) and the XML response.

**XML REQUEST (cURL)**

curl -c cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d "<Login><username>username</username><password>password</password></Login>" http://keylight.lockpath.com:4443/SecurityService/Login

**XML RESPONSE**

<boolean xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">true</boolean>

**JSON REQUEST (cURL)**

curl -c cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d "{ \"username\": \"user1\", \"password\": \"12345\" }" https://keylight.lockpath.com:4443/SecurityService/Login

**JSON RESPONSE**

true

**Auto-provision an account using LDAP**

#### If Auto-Provision is enabled for the LDAP Profile and the authenticated user does not have a Keylight application account, one will be created with API access enabled and then authenticated to the Keylight application.

Input: ldapSettingsId (Int) LDAP Profile ID within Keylight application

**Examples**

**XML REQUEST (cURL)**

curl -c cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d "<Login><username>username</username><password>password</password><ldapSettingsId>1</ldapSet tingsId></Login>" http://keylight.lockpath.com:4443/SecurityService/Login

**XML RESPONSE**

<boolean xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">true</boolean>

**JSON REQUEST (cURL)**

curl -c cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d "{ \"username\": \"user1\", \"password\": \"12345\", \"ldapSettingsId\": \"1\"}" https://keylight.lockpath.com:4443/SecurityService/Login

**JSON RESPONSE**

true

**Ping**

#### Refreshes a valid Keylight Platform session.

URL: http://[instance-name]:[port]/SecurityService/Ping Method: GET

Input: No input allowed

Permissions: The account that is used to log into the application must have access to the Keylight API.

**Examples**

In this example, cURL uses the -b option to provide authentication. Ping uses the default GET method, so the method does not need to be specified in the request.

**XML REQUEST (cURL)**

curl -b cookie.txt http://keylight.lockpath.com:4443/SecurityService/Ping

**XML RESPONSE**

<boolean xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">true</boolean>

**JSON REQUEST (cURL)**

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/SecurityService/Ping

**JSON RESPONSE**

true

## Logout

#### Terminates a Keylight Platform session.

URL: http://[instance-name]:[port]/SecurityService/Logout Method: GET

Input: No input allowed

Permissions: The account that is used to log into the application must have access to the Keylight API.

**Examples**

In this example, cURL uses the -b option to retrieve a stored session, authenticate it, and then terminate the session.

**XML/JSON REQUEST (cURL)**

curl -b cookie.txt http://keylight.lockpath.com:4443/SecurityService/Logout

**XML/JSON RESPONSE**

true

## GetUser

#### Returns all fields for a given user.

URL: http://[instance name]:[port]/SecurityService/GetUser?id={USERID} Method: GET

Input: ID (Integer): The ID of the desired user

Permissions: The authentication account must have Read Administrative Access permissions to Administer - Users.

The Language object of the GetUser method reveals the language in use in the Keylight Platform. The Language object works in combination with the Preferred Locale feature. When one of the languages with a corresponding locale code is active in the Keylight Platform, the Preferred Locale field value in My Profile preferences is set for the user. In the Keylight Platform, the default language is English, and since English has a related locale code, the default language value is "1033."

If an API request returns a language that is not available, or if a language is not active in the instance, the error message "Invalid Language ID" returns. You can hover the cursor over the language name in the Keylight Setup > Multilingual > Languages area of the Keylight Platform to reveal the language ID. For a list of languages and corresponding language IDs available in the Keylight Platform, see Language IDs in the appendix.

The following list includes the language IDs and language names, and the corresponding locale IDs and locale names.

|  |  |  |  |
| --- | --- | --- | --- |
| **Language ID** | **Language Name** | **Locale ID** | **Locale Name** |
| 9 | English | 1033 | English (United States) |
| 9 | English | 2057 | English (United Kingdom) |
| 12 | French | 1036 | French (France) |
| 16 | Italian | 1040 | Italian (Italy) |
| 10 | Spanish | 2058 | Spanish (Mexico) |
| 10 | Spanish | 3082 | Spanish (Spain) |

**Examples**

GetUser returns all fields for a given user. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" "http://keylight.lockpath.com:4443/SecurityService/GetUser?id=123"

**XML RESPONSE**

<?xml version="1.0" encoding="UTF-8"?>

<UserItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>123</Id>

<FullName>Betty Barnes</FullName>

<Username>bettybarnes</Username>

<IsActive>true</IsActive>

<IsLocked>false</IsLocked>

<IsDeleted>false</IsDeleted>

<AccountType>1</AccountType>

<FirstName>Betty</FirstName>

<MiddleName />

<LastName>Barnes</LastName>

<Title />

<Language>1033</Language>

<EmailAddress>[betty.barnes@email.com](mailto:betty.barnes@email.com)</EmailAddress>

<HomePhone />

<WorkPhone />

<MobilePhone />

<Fax />

<IsSAML>false</IsSAML>

<IsLDAP>false</IsLDAP>

<SecurityConfiguration>

<Id>1</Id>

<DisplayName>Standard Configuration</DisplayName>

</SecurityConfiguration>

<APIAccess>false</APIAccess>

<Groups />

<SecurityRoles>

<SecurityRole>

<Id>1</Id>

<Name>End User</Name>

</SecurityRole>

</SecurityRoles>

<FunctionalRoles />

</UserItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" "http://keylight.lockpath.com:4443/SecurityService/GetUser?id=123"

### JSON RESPONSE

{

"Id": 123,

"FullName": "Barnes, Betty", "Username": "bettybarnes", "IsActive": true, "IsLocked": false, "IsDeleted": false, "AccountType": 1, "FirstName": "Betty", "MiddleName": "", "LastName": "Barnes", "Title": "",

"Language": 1033,

"EmailAddress": "[betty.barnes@email.com](mailto:betty.barnes@email.com)", "HomePhone": "",

"WorkPhone": "",

"MobilePhone": "",

"Fax": "",

"IsSAML": false, "IsLDAP": false, "SecurityConfiguration": {

"Id": 1,

"DisplayName": "Standard Configuration"

},

"APIAccess": false, "Groups": [], "SecurityRoles": [

{

"Id": 1,

"Name": "End User"

}

],

"FunctionalRoles": []

}

## GetUsers

#### Returns a list of users and supporting fields. The list does not include Deleted users and can include non-Keylight user accounts.

URL: http://[instance name]:[port]/SecurityService/GetUsers Method: POST

Input: pageIndex (integer): The index of the page of result to return. Must be > 0 pageSize (integer): The size of the page results to return. Must be >= 1 FieldFilter (optional) <Filters>: The filter parameters the users must meet to be included

Use filters to return only the users meeting the selected criteria. Remove all filters to return a list of all users including deleted non-Keylight user accounts.

Filters:

|  |  |  |
| --- | --- | --- |
| **Field** | **Filter Types** | **Usable Values** |
| Active | * 5 - EqualTo * 6 - NotEqualTo | * True * False |
| Deleted | * 5 - EqualTo * 6 - NotEqualTo | * True * False |
| AccountType | * 5 - EqualTo * 6 - NotEqualTo * 10002 - ContainsAny | * 1, Full, FullUser * 2, Vendor, VendorContact, VendorContactUser * 4, Awareness, AwarenessUser |

Permissions: The authentication account must have Read Administrative Access permissions to Administer - Users.

**Filter Examples**

<filters>

<FieldFilter>

<FieldPath>

<Field>

<ShortName>Deleted</ShortName>

</Field>

</FieldPath>

<FilterType>6</FilterType>

<Value>True</Value>

</FieldFilter>

<FieldFilter>

<Field>

<ShortName>Active</ShortName>

</Field>

<FilterType>5</FilterType>

<Value>False</Value>

</FieldFilter>

<FieldFilter>

<Field>

<ShortName>AccountType</ShortName>

</Field>

<FilterType>10002</FilterType>

<Value>1|4</Value>

</FieldFilter>

</filters>

**Examples**

#### GetUsers returns users and supporting fields. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetUsersInput.xml

"http://keylight.lockpath.com:4443/SecurityService/GetUsers"

### XML REQUEST (GetUsersInput.xml)

<GetUsers>

<pageIndex>0</pageIndex>

<pageSize>4</pageSize>

<filters>

<FieldFilter>

<Field>

<ShortName>AccountType</ShortName>

</Field>

<FilterType>10002</FilterType>

<Value>1|2</Value>

</FieldFilter>

</filters>

</GetUsers>

### XML RESPONSE

<?xml version="1.0" encoding="UTF-8"?>

<UserList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<User>

<Id>19</Id>

<FullName>Contact1, Vendor</FullName>

<Username>vc1</Username>

<Active>true</Active>

<Deleted>false</Deleted>

<AccountType>2</AccountType>

<Vendor>

<Id>1</Id>

<DisplayName>Vendor1</DisplayName>

</Vendor>

</User>

<User>

<Id>10</Id>

<FullName>User, Test</FullName>

<Username>test</Username>

<Active>true</Active>

<Deleted>false</Deleted>

<AccountType>1</AccountType>

</User>

</UserList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetUsersInput.json "http://keylight.lockpath.com:4443/SecurityService/GetUsers"

### JSON REQUEST (GetUsersInput.json)

{

"pageIndex": "0",

"pageSize": "4", "filters":

[

{

"Field":

{

"ShortName": "AccountType"

},

"FilterType": "10002", "Value": "1|2"

}

]

}

### JSON RESPONSE

[

{

"Id": 19,

"FullName": "Contact1, Vendor", "Username": "vc1",

"Active": true, "Deleted": false, "AccountType": 2, "Vendor": {

"Id": 1,

"DisplayName": "Vendor1"

}

},

{

"Id": 10,

"FullName": "User, Test",

"Username": "test", "Active": true, "Deleted": false, "AccountType": 1,

}

]

## GetUserCount

#### Returns a count of Keylight users. The count does not include Deleted users and can include non-Keylight user accounts, such as Vendor Contacts.

URL: http://[instance name]:[port]/SecurityService/GetUserCount Method: POST

Input: FieldFilter <Filters>: The filter parameters the users must meet to be included

Use filters to return only the users meeting the selected criteria. Remove all filters to return a count of all users including deleted non-Keylight user accounts.

Filters:

|  |  |  |
| --- | --- | --- |
| **Field** | **Filter Types** | **Usable Values** |
| Active | * 5 - EqualTo * 6 - NotEqualTo | * True * False |
| Deleted | * 5 - EqualTo * 6 - NotEqualTo | * True * False |
| AccountType | * 5 - EqualTo * 6 - NotEqualTo * 10002 - ContainsAny | * 1, Full, FullUser * 2, Vendor, VendorContact, VendorContactUser * 4, Awareness, AwarenessUser |

Permissions: The authentication account must have Read Administrative Access permissions to Administer - Users.

**Filter Examples**

<filters>

<FieldFilter>

<FieldPath>

<Field>

<ShortName>Deleted</ShortName>

</Field>

</FieldPath>

<FilterType>6</FilterType>

<Value>True</Value>

</FieldFilter>

<FieldFilter>

<Field>

<ShortName>Active</ShortName>

</Field>

<FilterType>5</FilterType>

<Value>False</Value>

</FieldFilter>

<FieldFilter>

<Field>

<ShortName>AccountType</ShortName>

</Field>

<FilterType>10002</FilterType>

<Value>1|4</Value>

</FieldFilter>

</filters>

**Examples**

#### GetUserCount returns a count of users. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetUserCount.xml

"http://keylight.lockpath.com:4443/SecurityService/GetUserCount"

### XML REQUEST (GetUsersCount.xml)

<GetUserCount>

<filters>

<FieldFilter>

<Field>

<ShortName>AccountType</ShortName>

</Field>

<FilterType>5</FilterType>

<Value>1</Value>

</FieldFilter>

</filters>

</GetUserCount>

### XML RESPONSE

<int xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">#</int>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetUserCount.json "http://keylight.lockpath.com:4443/SecurityService/GetUserCount"

### JSON REQUEST (GetUsersCount.json)

[

{

"Field":

{

"ShortName": "AccountType"

},

"FilterType": "5",

"Value": "1"

}

]

### JSON RESPONSE

#

## CreateUser

#### Create a user account.

URL: http://[instance name]:[port]/SecurityService/CreateUser Method: POST

Input: Various user fields

Permissions: The authentication account must have Read and Create Administrative Access permissions to Administer - Users. For vendor contacts, the authentication account must also have the following permissions:

* + Read, Create, Update General Access to Vendor Profiles
  + View and Edit Vendor Profiles workflow stage
  + Vendor Profiles record permission

The Language object of the CreateUser method determines the language in the Keylight Platform. The Language object works in combination with the Preferred Locale feature. When one of the languages with a corresponding locale code is active in the Keylight Platform, the Preferred Locale field value in My Profile preferences is set for the user. In the Keylight Platform, the default language is English, and since English has a related locale code, the default language value is "1033."

If an API request uses an available language, but the language object does not have a corresponding locale code, the existing language persists, and the default Preferred Locale ID remains set to "1033." For example, if the existing language is Portuguese, which does not have a related locale code, the language persists, and the Preferred Locale field value is English (United States).

If an API request calls for a language that is not available, or if a language is not active in the instance, the error message "Invalid Language ID" returns. You can hover the cursor over the language name in the Keylight Setup > Multilingual > Languages area of the Keylight Platform to reveal the language ID. For a list of languages and corresponding language IDs available in the Keylight Platform, see Language IDs in the appendix.

The following list includes the language IDs and language names, and the corresponding locale IDs and locale names.

|  |  |  |  |
| --- | --- | --- | --- |
| **Language ID** | **Language Name** | **Locale ID** | **Locale Name** |
| 9 | English | 1033 | English (United States) |
| 9 | English | 2057 | English (United Kingdom) |
| 12 | French | 1036 | French (France) |
| 16 | Italian | 1040 | Italian (Italy) |
| 10 | Spanish | 2058 | Spanish (Mexico) |
| 10 | Spanish | 3082 | Spanish (Spain) |

**Examples**

CreateUser creates a user account. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @CreateUser.xml

"http://keylight.lockpath.com:4443/SecurityService/CreateUser"

**XML REQUEST (CreateUser.xml)**

<CreateUser>

<user xmlns:i='[http://www.w3.org/2001/XMLSchema-instance'](http://www.w3.org/2001/XMLSchema-instance%27) xmlns:a='[http://www.w3.org/2001/XMLSchema'](http://www.w3.org/2001/XMLSchema%27)>

<Username>test</Username>

<Password>password</Password>

<Active>true</Active>

<Locked>false</Locked>

<AccountType>1</AccountType>

<FirstName>test</FirstName>

<MiddleName></MiddleName>

<LastName>user</LastName>

<Title></Title>

<Language>1033</Language>

<EmailAddress>[test@user.com](mailto:test@user.com)</EmailAddress>

<HomePhone></HomePhone>

<WorkPhone></WorkPhone>

<MobilePhone></MobilePhone>

<Fax></Fax>

<Manager>

<Id>12</Id>

</Manager>

<Department>

<Id>10</Id>

</Department>

<IsSAML>false</IsSAML>

<IsLDAP>false</IsLDAP>

<LDAPDirectory>

<Id>1</Id>

</LDAPDirectory>

<SecurityConfiguration>

<Id>1</Id>

</SecurityConfiguration>

<APIAccess>false</APIAccess>

<Groups>

<Group>

<Id>5</Id>

</Group>

</Groups>

<SecurityRoles>

<SecurityRole>

<Id>1</Id>

</SecurityRole>

</SecurityRoles>

<FunctionalRoles>

<Record>

<Id>9</Id>

</Record>

</FunctionalRoles>

</user>

</CreateUser>

### XML REQUEST (VendorContact.xml)

<CreateUser>

<user xmlns:i='[http://www.w3.org/2001/XMLSchema-instance'](http://www.w3.org/2001/XMLSchema-instance%27) xmlns:a='[http://www.w3.org/2001/XMLSchema'](http://www.w3.org/2001/XMLSchema%27)>

<Username>vendor</Username>

<AccountType>2</AccountType>

<Vendor>

<Id>1</Id>

</Vendor>

<FirstName>Vendor</FirstName>

<MiddleName></MiddleName>

<LastName>Contact</LastName>

<Title></Title>

<Language>1033</Language>

<EmailAddress>[vendor@contact.com](mailto:vendor@contact.com)</EmailAddress>

<WorkPhone></WorkPhone>

<MobilePhone></MobilePhone>

<Fax></Fax>

<VendorComments></VendorComments>

</user>

</CreateUser>

### XML RESPONSE

GetUser

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @CreateUser.json http://keylight.lockpath.com:4443/SecurityService/CreateUser

### JSON REQUEST (CreateUser.json)

{

"Username": "test", "Password": "password", "Active": true, "Locked": false, "AccountType": 1, "FirstName": "Test",

"MiddleName": "",

"LastName": "User",

"Title": "", "Language": 1033,

"EmailAddress": "[test@user.com](mailto:test@user.com)", "HomePhone": "",

"WorkPhone": "",

"MobilePhone": "",

"Fax": "",

"IsSAML": false, "IsLDAP": true, "LDAPDirectory": {

"Id": "1"

},

"Manager": {

"Id": "10"

},

"Department": { "Id": "10"

},

"SecurityConfiguration": { "Id": "1"

},

"APIAccess": false,

"Groups": [

{

"Id": "2"

}

],

"SecurityRoles": [

{

"Id": "1"

}

],

"FunctionalRoles": [

{

"Id": "9"

}

]

}

### JSON REQUEST (VendorContact.json)

{

"Username": "vendor", "AccountType": 2, "Vendor": {

"Id": "1"

},

"FirstName": "Vendor", "MiddleName": "", "LastName": "Contact", "Title": "", "Language": 1033,

"EmailAddress": "[vendor@contact.com](mailto:vendor@contact.com)", "HomePhone": "",

"WorkPhone": "",

"MobilePhone": "",

"Fax": "",

"VendorComments": ""

}

### JSON RESPONSE

GetUser

## UpdateUser

#### Update a user account.

URL: http://[instance name]:[port]/SecurityService/UpdateUser Method: POST

Input: Various user fields.

Permissions: The authentication account must have Read and Update Administrative Access permissions to Administer - Users. For vendor contacts, the authentication account must also have the following permissions:

* + Read and Update General Access to Vendor Profiles
  + View and Edit Vendor Profiles workflow stage
  + Vendor Profiles record permission

The Language object of the UpdateUser method determines the language in the Keylight Platform. The Language object works in combination with the Preferred Locale feature. When one of the languages with a corresponding locale code is active in the Keylight Platform, the Preferred Locale field value in My Profile preferences is set for the user. In the Keylight Platform, the default language is English, and since English has a related locale code, the default language value is "1033."

If an API request uses an available language, but the language object does not have a corresponding locale code, the existing language persists, and the default Preferred Locale ID remains set to "1033." For example, if the existing language is Portuguese, which does not have a related locale code, the language persists, and the Preferred Locale field value is English (United States).

If an API request calls for a language that is not available, or if a language is not active in the instance, the error message "Invalid Language ID" returns. You can hover the cursor over the language name in the Keylight Setup > Multilingual > Languages area of the Keylight Platform to reveal the language ID. For a list of languages and corresponding language IDs available in the Keylight Platform, see Language IDs in the appendix.

The following list includes the language IDs and language names, and the corresponding locale IDs and locale names.

|  |  |  |  |
| --- | --- | --- | --- |
| **Language ID** | **Language Name** | **Locale ID** | **Locale Name** |
| 9 | English | 1033 | English (United States) |
| 9 | English | 2057 | English (United Kingdom) |
| 12 | French | 1036 | French (France) |
| 16 | Italian | 1040 | Italian (Italy) |
| 10 | Spanish | 2058 | Spanish (Mexico) |
| 10 | Spanish | 3082 | Spanish (Spain) |

**Examples**

UpdateUser updates a user account. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @UpdateUser.xml

"http://keylight.lockpath.com:4443/SecurityService/UpdateUser"

**XML REQUEST (UpdateUser.xml)**

<UpdateUser>

<user xmlns:i='[http://www.w3.org/2001/XMLSchema-instance'](http://www.w3.org/2001/XMLSchema-instance%27) xmlns:a='[http://www.w3.org/2001/XMLSchema'](http://www.w3.org/2001/XMLSchema%27)>

<Id>10</Id>

<Username>test</Username>

<Password>password</Password>

<Active>true</Active>

<Locked>false</Locked>

<AccountType>1</AccountType>

<FirstName>test</FirstName>

<MiddleName></MiddleName>

<LastName>user</LastName>

<Title></Title>

<Language>1033</Language>

<EmailAddress>[test@user.com](mailto:test@user.com)</EmailAddress>

<HomePhone></HomePhone>

<WorkPhone></WorkPhone>

<MobilePhone></MobilePhone>

<Fax></Fax>

<IsSAML>false</IsSAML>

<IsLDAP>true</IsLDAP>

<LDAPDirectory>

<Id>1</Id>

</LDAPDirectory>

<Manager>

<Id>15</Id>

</Manager>

<Department>

<Id>601</Id>

</Department>

<SecurityConfiguration>

<Id>1</Id>

</SecurityConfiguration>

<APIAccess>true</APIAccess>

<Groups>

<Group>

<Id>2</Id>

</Group>

<Group>

<Id>3</Id>

</Group>

</Groups>

<SecurityRoles>

<SecurityRole>

<Id>1</Id>

</SecurityRole>

<SecurityRole>

<Id>2</Id>

</SecurityRole>

</SecurityRoles>

<FunctionalRoles>

<Record>

<Id>34</Id>

</Record>

<Record>

<Id>35</Id>

</Record>

</FunctionalRoles>

</user>

</UpdateUser>

### XML RESPONSE

GetUser

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @UpdateUser.json http://keylight.lockpath.com:4443/SecurityService/UpdateUser

### JSON REQUEST (UpdateUser.json)

{

"Id": "9504",

"Username": "test", "Password": "password", "Active": true, "Locked": false, "AccountType": 1, "FirstName": "Test",

"MiddleName": "",

"LastName": "User",

"Title": "", "Language": 1033,

"EmailAddress": "[test@user.com](mailto:test@user.com)", "HomePhone": "",

"WorkPhone": "",

"MobilePhone": "",

"Fax": "",

"IsSAML": false, "IsLDAP": true, "LDAPDirectory": {

"Id": "1"

},

"Manager": {

"Id": "15"

},

"Department": { "Id": "601"

},

"SecurityConfiguration": { "Id": "1"

},

"APIAccess": true, "Groups": [

{

"Id": "2"

},

{

"Id": "3"

}

],

"SecurityRoles": [

{

"Id": "1"

},

{

"Id": "2"

}

],

"FunctionalRoles": [

{

"Id": "34"

},

{

"Id": "35"

}

]

}

#### ]

**JSON RESPONSE**

GetUser

## DeleteUser

#### Delete a user account.

URL: http://[instance name]:[port]/SecurityService/DeleteUser Method: DELETE

Input: ID: The ID of the user

Permissions: The authentication account must have Read and Delete Administrative Access permissions to Administer - Users. For vendor contacts, the authentication account can alternatively have Read and Delete General Access to Vendor Profiles.

**Examples**

DeleteUser deletes a user account. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X DELETE -d "<DeleteUser><id>10</id></DeleteUser>"

"http://keylight.lockpath.com:4443/SecurityService/DeleteUser"

**XML RESPONSE**

<int xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">true</int>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X DELETE -d "10"

http://keylight.lockpath.com:4443/SecurityService/DeleteUser

### JSON RESPONSE

true

## GetGroup

#### Returns all fields for a given group.

URL: http://[instance name]:[port]/SecurityService/GetGroup Method: GET

Input: ID: The ID of the desired group

Permissions: The authentication account must have Read Administrative Access permissions to Administer - Groups.

**Examples**

GetGroup returns all fields for a given group. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" "http://keylight.lockpath.com:4443/SecurityService/GetGroup?id=2"

**XML RESPONSE**

<?xml version="1.0" encoding="UTF-8"?>

<GroupItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>2</Id>

<Name>test group</Name>

<Description />

<BusinessUnit>false</BusinessUnit>

<LDAPDirectory>

<Id>1</Id>

<DisplayName>whoa</DisplayName>

</LDAPDirectory>

<LDAPGroupName>0b7fb422-3609-4587-8c2e-94b10f67d1bf</LDAPGroupName>

<LDAPGroupDN>CN=whoa,DC=lockpath,DC=com</LDAPGroupDN>

<SecurityRoles />

<Users />

<ChildGroups />

<ParentGroups />

</GroupItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" http://keylight.lockpath.com:4443/SecurityService/GetGroup?id=2

### JSON RESPONSE

{

"Id": 2,

"Name": "0b7fb422-3609-4587-8c2e-94b10f67d1bf",

"Description": "", "BusinessUnit": false, "LDAPDirectory": {

"Id": 1,

"DisplayName": "whoa"

},

"LDAPGroupName": "0b7fb422-3609-4587-8c2e-94b10f67d1bf", "LDAPGroupDN": "CN=whoa,DC=dev,DC=lockpath,DC=com", "SecurityRoles": [],

"Users": [],

"ChildGroups": [], "ParentGroups": []

}

## GetGroups

#### Returns the ID and Name for groups. A filter may be applied to return only the groups meeting selected criteria. URL: http://[instance name]:[port]/SecurityService/GetGroups

Method: POST

Input: pageIndex (Integer): The index of the page of result to return. Must be > 0 pageSize (Integer): The size of the page results to return. Must be >= 1 FieldFilter (optional) <Filters>: The filter parameters the groups must meet to be included

Filter: <filters>

<FieldFilter>

<FieldPath>

<Field>

<ShortName>BusinessUnit</ShortName>

</Field>

</FieldPath>

<FilterType>5</FilterType>

<Value>False</Value>

</FieldFilter>

</filters>

#### Permissions: The authentication account must have Read Administrative Access permissions to Administer - Groups.

**Examples**

GetGroups returns the ID and name for groups. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetGroups.xml

http://keylight.lockpath.com:4443/SecurityService/GetGroups

**XML REQUEST (GetGroups.xml)**

<GetGroups>

<pageIndex>0</pageIndex>

<pageSize>100</pageSize>

<filters>

<FieldFilter>

<FieldPath>

<Field>

<ShortName>BusinessUnit</ShortName>

</Field>

</FieldPath>

<FilterType>5</FilterType>

<Value>False</Value>

</FieldFilter>

</filters>

</GetGroups>

### XML RESPONSE

<?xml version="1.0" encoding="UTF-8"?>

<GroupList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Group>

<Id>10</Id>

<Name>Anonymous Incident Analysts</Name>

</Group>

<Group>

<Id>7</Id>

<Name>Business Continuity Plan Approvers</Name>

</Group>

</GroupList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetGroups.json http://keylight.lockpath.com:4443/SecurityService/GetGroups

### JSON REQUEST (GetGroups.json)

{

"pageIndex": "0",

"pageSize": "100", "filters":

[

{

"Field":

{

"ShortName": "BusinessUnit"

},

"FilterType": "5",

"Value": "False"

}

]

}

### JSON RESPONSE

[

{

"Id": 10,

"Name": "Anonymous Incident Analysts"

},

{

"Id": 7,

"Name": "Business Continuity Plan Approvers"

}

]

## CreateGroup

#### Creates a group.

URL: http://[instance name]:[port]/SecurityService/CreateGroup Method: POST

Input: Various group fields

Permissions: The authentication account must have Read and Create Administrative Access permissions to Administer - Groups.

**Examples**

CreateGroup creates a group. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @CreateGroup.xml

http://keylight.lockpath.com:4443/SecurityService/CreateGroup

**XML REQUEST (CreateGroup.xml)**

<CreateGroup>

<group xmlns:i='[http://www.w3.org/2001/XMLSchema-instance'](http://www.w3.org/2001/XMLSchema-instance%27) xmlns:a='[http://www.w3.org/2001/XMLSchema'](http://www.w3.org/2001/XMLSchema%27)>

<Name>test group</Name>

<Description></Description>

<BusinessUnit>false</BusinessUnit>

<Users>

<User>

<Id>10</Id>

</User>

<User>

<Id>12</Id>

</User>

</Users>

<ChildGroups>

<Group>

<Id>5</Id>

</Group>

</ChildGroups>

<ParentGroups>

<Group>

<Id>10</Id>

</Group>

</ParentGroups>

</group>

</CreateGroup>

### XML RESPONSE

GetGroup

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @CreateGroup.json http://keylight.lockpath.com:4443/SecurityService/CreateGroup

### JSON REQUEST (CreateGroup.json)

{

"Name": "test group",

"Description": "", "BusinessUnit": false, "Users": [

{

"Id": "10"

},

{

"Id": "12"

}

],

"ChildGroups": [

{

"Id": "5"

}

],

"ParentGroups": [

{

"Id": "10"

}

]

}

### JSON RESPONSE

GetGroup

## UpdateGroup

#### Updates a group.

URL: http://[instance name]:[port]/SecurityService/UpdateGroup Method: POST

Input: Various group fields.

Permissions: The authentication account must have Read and Update Administrative Access permissions to Administer - Groups.

**Examples**

UpdateGroup updates a group. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @UpdateGroup.xml

http://keylight.lockpath.com:4443/SecurityService/UpdateGroup

**XML REQUEST (UpdateGroup.xml)**

<UpdateGroup>

<group xmlns:i='[http://www.w3.org/2001/XMLSchema-instance'](http://www.w3.org/2001/XMLSchema-instance%27) xmlns:a='[http://www.w3.org/2001/XMLSchema'](http://www.w3.org/2001/XMLSchema%27)>

<Id>6</Id>

<Name>API Updated Group</Name>

<Description>Here's a description.</Description>

<BusinessUnit>false</BusinessUnit>

<Users>

<User>

<Id>10</Id>

</User>

<User>

<Id>11</Id>

</User>

</Users>

<ChildGroups>

<Group>

<Id>5</Id>

</Group>

<Group>

<Id>7</Id>

</Group>

</ChildGroups>

<ParentGroups>

<Group>

<Id>2</Id>

</Group>

<Group>

<Id>3</Id>

</Group>

</ParentGroups>

</group>

</UpdateGroup>

### XML RESPONSE

GetGroup

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @UpdateGroup.json http://keylight.lockpath.com:4443/SecurityService/UpdateGroup

### JSON REQUEST (UpdateGroup.json)

{

"Id": "6",

"Name": "API Updated Group", "Description": "Here's a description.", "BusinessUnit": false,

"Users": [

{

"Id": "10"

},

{

"Id": "11"

}

],

"ChildGroups": [

{

"Id": "5"

},

{

"Id": "7"

}

],

"ParentGroups": [

{

"Id": "2"

},

{

"Id": "3"

}

]

}

#### ]

**JSON RESPONSE**

GetGroup

## DeleteGroup

#### Delete a group.

URL: URL: http://[instance name]:[port]/SecurityService/DeleteGroup Method: DELETE

Input: ID: The ID of the group

Permissions: The authentication account must have Read and Delete Administrative Access permissions to Administer - Groups.

**Examples**

DeleteGroup deletes a group. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X DELETE -d "<DeleteGroup><id>10</id></DeleteGroup>"

"http://keylight.lockpath.com:4443/SecurityService/DeleteGroup"

**XML RESPONSE**

<int xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">true</int>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X DELETE -d "10"

http://keylight.lockpath.com:4443/SecurityService/DeleteGroup

### JSON RESPONSE

true

# 3: Component Services API

#### This chapter includes a description of the various methods and calls for posting, getting, and deleting data from the Keylight Platform.

[GetComponent](#_bookmark21) [54](#_bookmark21) [GetComponentList](#_bookmark22) [55](#_bookmark22) [GetComponentByAlias](#_bookmark23) [57](#_bookmark23) [GetField](#_bookmark24) [59](#_bookmark24) [GetFieldList](#_bookmark25) [61](#_bookmark25) [GetAvailableLookupRecords](#_bookmark26) [65](#_bookmark26) [GetLookupReportColumnFields](#_bookmark27) [68](#_bookmark27) [GetRecord](#_bookmark28) [70](#_bookmark28) [GetRecords](#_bookmark29) [73](#_bookmark29) [GetRecordCount](#_bookmark30) [77](#_bookmark30) [GetDetailRecord](#_bookmark31) [81](#_bookmark31) [GetDetailRecords](#_bookmark32) [86](#_bookmark32) [GetRecordAttachment](#_bookmark33) [92](#_bookmark33) [GetRecordAttachments](#_bookmark34) [94](#_bookmark34) [GetWorkflow](#_bookmark35) [96](#_bookmark35) [GetWorkflows](#_bookmark36) [104](#_bookmark36) [TransitionRecord](#_bookmark37) [106](#_bookmark37) [VoteRecord](#_bookmark38) [108](#_bookmark38) [CreateRecord](#_bookmark39) [110](#_bookmark39) [UpdateRecord](#_bookmark40) [118](#_bookmark40) [UpdateRecordAttachments](#_bookmark41) [130](#_bookmark41) [ImportFile](#_bookmark42) [137](#_bookmark42) [Issue Assessments](#_bookmark43) [139](#_bookmark43) [DeleteRecord](#_bookmark44) [148](#_bookmark44) [DeleteRecordAttachments](#_bookmark45) [150](#_bookmark45)

**GetComponent**

Retrieves a component specified by its ID. A component is a user-defined data object such as a custom content table. The component ID may be found by using GetComponentList.

URL: http://[instance name]:[port]/ComponentService/GetComponent?id={ID} Method: GET

Input: ID (Integer): The ID of the desired component

Permissions: The authentication account must have: Read General Access permissions for the specific component enabled.

**Examples**

This sample obtains a list of the specified component within Keylight. GetComponent uses the default GET method, so the method does not need to be specified in the request. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetComponent?id=10005

**XML RESPONSE**

<?xml version="1.0"?>

<ComponentItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>10500</Id>

<Name>Incident Reports</Name>

<SystemName>LPIncidentReports</SystemName>

<ShortName>LPIncidentReports</ShortName>

</ComponentItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetComponent?id=10005

### JSON RESPONSE

{

"Id": 10050,

"Name": "Incident Reports", "SystemName": "LPIncidentReports", "ShortName": "LPIncidentReports"

}

**GetComponentList**

#### Returns a complete list of all Keylight components available to the user based on account permissions. No input elements are used. The list will be ordered in ascending alphabetical order of the component name.

URL: http://[instance name]:[port]/ComponentService/GetComponentList Method: GET

Input: No inputs allowed

Permissions: The authentication account must have: Read General Access permission to the enabled components.

**Examples**

This sample obtains a list of all components within the Keylight Platform. The -b option is used to provide authentication. GetComponentList uses the default GET method, so the method does not need to be specified in the request.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetComponentList

**XML RESPONSE**

<?xml version="1.0"?>

<ComponentList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Component>

<Id>10003</Id>

<Name>Device Types</Name>

<SystemName>DeviceTypes</SystemName>

<ShortName>DeviceTypes</ShortName>

</Component>

<Component>

<Id>10001</Id>

<Name>Devices</Name>

<SystemName>Devices</SystemName>

<ShortName>Devices</ShortName>

</Component>

…

</ComponentList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetComponentList

### JSON RESPONSE

[

{

"Id": "10003",

"Name": "Device Types", "SystemName": "DeviceTypes", "ShortName": "DeviceTypes"

},

{

"Id": "10001",

"Name": "Devices", "SystemName": "Devices", "ShortName": "Devices"

}

]

## GetComponentByAlias

#### Retrieves a component specified by its Alias. A component is a user-defined data object such as a custom content table. The component Alias may be found by using GetComponentList (ShortName).

URL: http://[instance name]:[port]/ComponentService/GetComponentByAlias?alias={Alias} Method: GET

Input: Alias (String): The Alias of the desired component

Permissions: The authentication account must have: Read General Access permission for the specific component enabled.

**Examples**

This sample obtains a list of the specified component within the Keylight Platform. GetComponentByAlias uses the default GET method, so the method does not need to be specified in the request. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetComponentByAlias?alias=IncidentReports

**XML RESPONSE**

<?xml version="1.0"?>

<ComponentItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>10021</Id>

<Name>Incident Reports</Name>

<SystemName>IncidentReports</SystemName>

<ShortName>IncidentReports</ShortName>

</ComponentItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetComponentByAlias?alias=IncidentReport s

### JSON RESPONSE

{

"Id": 10050,

"Name": "Incident Reports", "SystemName": "LPIncidentReports", "ShortName": "LPIncidentReports"

}

## GetField

#### Retrieves details for a field specified by its ID. The field ID may be found by using GetField. URL: http://[instance name]:[port]/ComponentService/GetField?id={FieldId} Method: GET

Input: ID (Integer): The field ID for the individual field within the component

Field Types: The response includes an integer value for field type. The fields are translated as described:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Types** | | | | | |
| **ID** | **Type** | **ID** | **Type** | **ID** | **Type** |
| 1 | Text | 5 | Lookup | 9 | Assessments |
| 2 | Numeric | 6 | Master/Detail | 10 | Yes/No |
| 3 | Date | 7 | Matrix |  | |
| 4 | IP Address | 8 | Documents |

Permissions: The authentication account must have: Read General Access permission to the enabled component and field.

**Examples**

The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetField?id=12

**XML RESPONSE**

<?xml version="1.0"?>

<FieldItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>12</Id>

<Name>Subnet Mask</Name>

<SystemName i:nil="true"/>

<ShortName>SubnetMask</ShortName>

<ReadOnly>false</ReadOnly>

<Required>false</Required>

<Nullable>false</Nullable>

<FieldType>4</FieldType>

<MaxLength i:nil="true"/>

<Precision i:nil="true"/>

<Scale i:nil="true"/>

<OneToMany>false</OneToMany>

</FieldItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetField?Id=12

### JSON RESPONSE

{

"Id": 12,

"Name": "Subnet Mask", "SystemName": "SubnetMask", "ShortName": "SubnetMask", "ReadOnly": false, "Required": false, "FieldType": 4, "OneToMany": false, "MatrixRows": []

}

## GetFieldList

#### Retrieves detail field listing for a component specified by its ID. The component ID may be found by using GetComponentList. Assessments field type will not be visible in this list.

URL: http://[instance name]:[port]/ComponentService/GetFieldList?componentId={COMPONENTID} Method: GET

Input: componentId (Integer): The ID of the desired component

Permissions: The authentication account must have: Read General Access permission to the selected component.

Fields to which the account does not have access are not returned.

**Examples**

The cURL -b option is used to provide authentication. GetFieldList uses the default GET method, so the method does not need to be specified in the request.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetFieldList?componentId=10001

**XML RESPONSE**

<FieldList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Field>

<Id>9</Id>

<Name>Acquisition Cost</Name>

<SystemName i:nil="true"/>

<ShortName>Cost</ShortName>

<ReadOnly>false</ReadOnly>

<Required>false</Required>

<Nullable>false</Nullable>

<FieldType>2</FieldType>

<MaxLength i:nil="true"/>

<Precision>15</Precision>

<Scale>2</Scale>

<OneToMany>false</OneToMany>

</Field>

<Field>

<Id>10</Id>

<Name>Acquisition Date</Name>

<SystemName i:nil="true"/>

<ShortName>AcquisitionDate</ShortName>

<ReadOnly>false</ReadOnly>

<Required>false</Required>

<Nullable>false</Nullable>

<FieldType>3</FieldType>

<MaxLength i:nil="true"/>

<Precision i:nil="true"/>

<Scale i:nil="true"/>

<OneToMany>false</OneToMany>

</Field>

<Field>

<Id>6</Id>

<Name>Asset Tag</Name>

<SystemName i:nil="true"/>

<ShortName>AssetTag</ShortName>

<ReadOnly>false</ReadOnly>

<Required>false</Required>

<Nullable>false</Nullable>

<FieldType>1</FieldType>

<MaxLength>100</MaxLength>

<Precision i:nil="true"/>

<Scale i:nil="true"/>

<OneToMany>false</OneToMany>

</Field>

<Field>

<Id>11</Id>

<Name>IP Address</Name>

<SystemName>IPAddress</SystemName>

<ShortName>IPAddress</ShortName>

<ReadOnly>false</ReadOnly>

<Required>false</Required>

<Nullable>false</Nullable>

<FieldType>4</FieldType>

<MaxLength i:nil="true"/>

<Precision i:nil="true"/>

<Scale i:nil="true"/>

<OneToMany>false</OneToMany>

</Field>

</FieldList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetFieldList?componentId=10001

### JSON RESPONSE

[

{

"Id": 9,

"Name": "Acquisition Cost", "SystemName": "Cost", "ShortName": "Cost", "ReadOnly": false, "Required": false, "FieldType": 2,

"Precision": 15,

"Scale": 2, "OneToMany": false, "MatrixRows": []

},

{

"Id": 10,

"Name": "Acquisition Date", "SystemName": "AcquisitionDate", "ShortName": "AcquisitionDate", "ReadOnly": false,

"Required": false, "FieldType": 3, "OneToMany": false, "MatrixRows": []

},

{

"Id": 6,

"Name": "Asset Tag", "SystemName": "AssetTag", "ShortName": "AssetTag", "ReadOnly": false, "Required": false, "FieldType": 1,

"MaxLength": 100, "OneToMany": false, "MatrixRows": []

},

{

"Id": 11,

"Name": "IP Address", "SystemName": "IPAddress", "ShortName": "IPAddress", "ReadOnly": false, "Required": false, "FieldType": 4, "OneToMany": false, "MatrixRows": []

}

]

## GetAvailableLookupRecords

#### Retrieves records that are available for population for a lookup field.

URL: http://[instance-name]:[port]/ComponentService/GetAvailableLookupRecords Method: POST

Input: fieldId (integer): The ID of the desired component

pageIndex (integer): The index of the page of result to return. Must be > 0 pageSize (integer): The size of the page results to return. Must be >= 1 recordId (integer): Optional ID of the record for which retrieving lookup records

Permissions: The authentication account must have Read/Create access to the component that contains the lookup field if no recordId is supplied, Read/Edit access to the component that contains the lookup field if a recordId is supplied, Read/Edit access to the lookup field, Read access to the recordId, and Read access to any lookup records.

**Examples**

The -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetAvailableLookupRecords.xml

http://keylight.lockpath.com:4443/ComponentService/GetAvailableLookupRecords

**XML REQUEST (GetAvailableLookupRecords.xml)**

<GetAvailableLookupRecords>

<fieldId>12345</fieldId>

<pageIndex>0</pageIndex>

<pageSize>1000</pageSize>

<recordId>123</recordId>

</GetAvailableLookupRecords>

### XML RESPONSE

<returns>

<ArrayOfDynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<DynamicRecordItem>

<Id>1</Id>

<DisplayName>The First Record</DisplayName>

<FieldValues/>

</DynamicRecordItem>

<DynamicRecordItem>

<Id>2</Id>

<DisplayName>The Second Record</DisplayName>

<FieldValues/>

</DynamicRecordItem>

<DynamicRecordItem>

<Id>3</Id>

<DisplayName>The Third Record</DisplayName>

<FieldValues/>

</DynamicRecordItem>

</ArrayOfDynamicRecordItem>

</returns>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetAvailableLookupRecords.json http://keylight.lockpath.com:4443/ComponentService/GetAvailableLookupRecords

### JSON REQUEST (GetAvailableLookupRecords.json)

{

"fieldId": "12345",

"pageIndex": "0",

"pageSize": "1000",

"recordId": "123"

}

### JSON RESPONSE

[

{

"Id": 1,

"DisplayName": "The First Record", "FieldValues": []

},

{

"Id": 2,

"DisplayName": "The Second Record", "FieldValues": []

},

{

"Id": 3,

"DisplayName": "The Third Record", "FieldValues": []

}

]

## GetLookupReportColumnFields

#### Gets the field information of each field in a field path that corresponds to a lookup report column. The lookupFieldId corresponds to a lookup field with a report definition on it and the fieldPathId corresponds to the field path to retrieve fields from, which is obtained from GetDetailRecord. GetLookupReportColumnFields compliments GetRecordDetail by adding additional details about the lookup report columns returned from GetRecordDetail.

URL: "http://[instance name]:[port]/ComponentService/GetLookupReportColumnFields?lookupFieldId=

{FIELDID}&fieldPathId={FIELDPATHID}"

Method: GET

Input: lookupFieldId (Integer): The ID of the desired lookup field fieldPathId (Integer): The ID for the desired lookup field path

Permissions: The authentication account must have Read General Access permissions to:

* Selected component
* Selected record
* Applicable fields in the component (table)

**Examples**

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8"

"http://keylight.lockpath.com:4443/ComponentService/GetLookupReportColumnFields?lookupFieldI d=1234&fieldPathId=5678"

**XML RESPONSE**

<LookupReportFieldList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Field>

<Id>123</Id>

<ComponentId>111</ComponentId>

<Name>Workflow Stage</Name>

<SystemName>WorkflowStage</SystemName>

</Field>

</LookupReportFieldList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetLookupReportColumnFields?lookupFieldI d=1234&fieldPathId=5678"

### JSON RESPONSE

[

{

"Id": 3,

"ComponentId": 10001, "Name": "DNS Name", "SystemName": "DNSName"

}

]

## GetRecord

#### Returns the complete set of fields for a given record within a component.

URL: http://[instance name]:[port]/ComponentService/GetRecord?componentId=

{COMPONENTID}&recordId={RECORDID}

Method: GET

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component Permissions: The authentication account must have Read General Access permissions to:

* + Selected component
  + Selected record
  + Applicable fields in the component (table)

**Examples**

The GetRecord retrieves all fields with permission to for a given record. The field keys can be matched with the data from the GetField method to get the name of the field. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" "http://keylight.lockpath.com:4443/ComponentService/GetRecord?componentId=10001&recordId=1"

**XML RESPONSE**

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>1</Id>

<DisplayName>192.168.1.84</DisplayName>

<FieldValues>

<KeyValuePair>

<key>10</key>

<value i:nil="true"/>

</KeyValuePair>

<KeyValuePair>

<key>1628</key>

<value i:type="DynamicRecordList"/>

</KeyValuePair>

<KeyValuePair>

<key>6</key>

<value i:type="a:string" xmlns:a="<http://www.w3.org/2001/XMLSchema>"/></KeyValuePair><KeyValuePair><k ey>34</key><value i:type="DynamicRecordList"/>

</KeyValuePair>

<KeyValuePair>

<key>2203</key>

<value i:type="DynamicRecordList"/>

</KeyValuePair>

<KeyValuePair>

<key>9</key>

<value i:nil="true"/>

</KeyValuePair>

<KeyValuePair>

<key>301</key>

<value i:type="a:dateTime" xmlns:a="<http://www.w3.org/2001/XMLSchema>">2012- 10-22T08:45:25.157</value>

</KeyValuePair>

<KeyValuePair>

<key>303</key>

<value i:type="DynamicRecordItem">

<Id>0</Id>

<DisplayName>System, Keylight</DisplayName>

<FieldValues/>

</value>

</KeyValuePair>

<KeyValuePair>

<key>15</key>

<value i:type="a:string" xmlns:a="<http://www.w3.org/2001/XMLSchema>"/>

</KeyValuePair>

…

<KeyValuePair>

<key>351</key>

<value i:type="DynamicRecordItem">

<Id>5</Id>

<DisplayName>Published</DisplayName>

<FieldValues/>

</value>

</KeyValuePair>

</FieldValues>

</DynamicRecordItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" "https://keylight.lockpath.com:4443/ComponentService/GetRecord?componentId=10001&recordId=1"

### JSON RESPONSE

{

"Id": 1,

"DisplayName": "192.168.1.84", "FieldValues": [

{

"Key": 3852,

"Value": 1

},

{

"Key": 3853,

"Value": 8

},

{

"Key": 3858,

"Value": {

" type": "DynamicRecordItem", "Id": 0,

"DisplayName": "System, Keylight", "FieldValues": []

}

},

{

"Key": 3860,

"Value": false

}

}

## GetRecords

#### Return the title/default field for a set of records within a chosen component. Filters may be applied to return only the records meeting selected criteria.

URL: http://[instance name]:[port]/ComponentService/GetRecords Method: POST

Input: componentID (Integer): The ID of the desired component

pageIndex (Integer): The index of the page of result to return. Must be > 0

pageSize (Integer): The size of the page results to return. Must be >= 1 SearchCriteriaItem (optional) <Filters>: The filter parameters the records must meet to be

counted

Filters: <filters>

<SearchCriteriaItem>

<Field Path>

<int>7</int>

</Field Path>

<FilterType>ID</FilterType>

<Value>value</Value>

</SearchCriteriaItem>

</filters>

#### SearchCriteriaItem: Describes a single filter. GetRecordCount supports adding an infinite amount of filter criteria.

Field Path: Describes the field ID for the column that the records will be filtered on. If the value is stored in the component directly, only one Field Path variable is needed. However, if the value is a lookup to another component, an additional Field Path variable will be required with the column value where the data resides. Field Path variables can be added as many as necessary to provide the correct path to the data.

FilterType: Describes the ID for the filter being implemented. For example, the FilterType for Is Null would be 15. If the FilterType would exclude the entry of a value like Is Empty (13) or Is Not Null (16), the <Value> tags should be removed from the request.

FilterTypes are listed in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Filter Types** | | | | | |
| **ID** | **Filter** | **ID** | **Filter** | **ID** | **Filter** |
| 1 | Contains | 8 | < | 15 | Is Null |
| 2 | Excludes | 9 | >= | 16 | Is Not Null |
| 3 | Starts With | 10 | <= | 10001 | Offset |
| 4 | Ends With | 11 | Between | 10002 | Contains Any |
| 5 | = | 12 | Not Between | 10003 | Contains Only |
| 6 | <> | 13 | Is Empty | 10004 | Contains None |
| 7 | > | 14 | Is Not Empty | 10005 | Contains At Least |

Value: Matches value (if applicable). For example, if the filter is Starts With "st," the Value would be

<Value>st</Value>.

Permissions: The authentication account must have Read General Access permission to:

* + Selected component
  + Selected record
  + Applicable fields in the component (table)

Describes the ID for the filter being implemented. For example, the FilterType for Is Null would be

1. If the FilterType would exclude the entry of a value like Is Empty (13) or Is Not Null (16), the

<Value> tags should be removed from the request. FilterTypes are listed in the table below.

**Examples**

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetRecordsInput.xml http://keylight.lockpath.com:4443/ComponentService/GetRecords

**XML REQUEST (GetRecordsInput.xml)**

<GetRecords>

<componentId>10001</componentId>

<pageIndex>0</pageIndex>

<pageSize>2</pageSize>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>11</int>

</FieldPath>

<FilterType>16</FilterType>

<Value></Value>

</SearchCriteriaItem>

</filters>

</GetRecords>

### XML RESPONSE

<ArrayOfDynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<DynamicRecordItem>

<Id>1</Id>

<DisplayName>192.168.1.84</DisplayName>

<FieldValues/>

</DynamicRecordItem>

<DynamicRecordItem>

<Id>2</Id>

<DisplayName>192.168.1.69</DisplayName>

<FieldValues/>

</DynamicRecordItem>

</ArrayOfDynamicRecordItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetRecordsInput.json http://keylight.lockpath.com:4443/ComponentService/GetRecordCount

### JSON REQUEST (GetRecordCount.json)

{

"componentId": "10001",

"pageIndex": "0",

"pageSize": "5", "filters": [

{

"FieldPath": [ 11

],

"FilterType": "1", "Value": "text value"

}

]

}

### JSON RESPONSE

[

{

"Id": 1,

"DisplayName": "192.168.1.84", "FieldValues": []

},

{

"Id": 2,

"DisplayName": "192.168.1.69", "FieldValues": []

},

]

## GetRecordCount

#### Return the number of records in a given component. Filters may be applied to return the count of records meeting a given criteria. This function may be used to help determine the amount of records before retrieving the records themselves.

URL: http://[instance name]:[port]/ComponentService/GetRecordCount Method: POST

Input: componentID (Integer): The ID of the desired component

SearchCriteriaItem (optional) <Filters>: The filter parameters the records must meet to be

counted

Filters: <filters>

<SearchCriteriaItem>

<Field Path>

<int>7</int>

</Field Path>

<FilterType>ID</FilterType>

<Value>value</Value>

</SearchCriteriaItem>

</filters>

#### SearchCriteriaItem: Describes a single filter. GetRecordCount supports adding an infinite amount of filter criteria.

Field Path: Describes the field ID for the column that the records will be filtered on. If the value is stored in the component directly, only one Field Path variable is needed. However, if the value is a lookup to another component, an additional Field Path variable will be required with the column value where the data resides. Field Path variables can be added as many as necessary to provide the correct path to the data.

FilterType: Describes the ID for the filter being implemented. For example, the FilterType for Is Null would be 15. If the FilterType would exclude the entry of a value like Is Empty (13) or Is Not Null (16), the <Value> tags should be removed from the request.

FilterTypes are listed in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Filter Types** | | | | | |
| **ID** | **Filter** | **ID** | **Filter** | **ID** | **Filter** |
| 1 | Contains | 8 | < | 15 | Is Null |
| 2 | Excludes | 9 | >= | 16 | Is Not Null |
| 3 | Starts With | 10 | <= | 10001 | Offset |
| 4 | Ends With | 11 | Between | 10002 | Contains Any |
| 5 | = | 12 | Not Between | 10003 | Contains Only |
| 6 | <> | 13 | Is Empty | 10004 | Contains None |
| 7 | > | 14 | Is Not Empty | 10005 | Contains At Least |

Value: Matches value (if applicable). For example, if the filter is Starts With "st," the Value would be

<Value>st</Value>.

Permissions: The authentication account must have Read General Access permission to:

* + Selected component
  + Applicable records
  + Applicable records in the component (table)

For more information on filter implementation, see Search Filters in the appendix.

**Examples**

The cURL -b option is used to provide authentication. In this example, the search is looking for values that do not have an empty address field. Because of this, the value parameter is left blank because IsNotEmpty is not comparing to any set value.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetRecordCount.xml

http://keylight.lockpath.com:4443/ComponentService/GetRecordCount

**XML REQUEST (GetRecordCount.xml)**

<GetRecordCount>

<componentId>10001</componentId>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>11</int>

</FieldPath>

<FilterType>16</FilterType>

</SearchCriteriaItem>

</filters>

</GetRecordCount>

### XML RESPONSE

<int xmlns="<http://schemas.microsoft.com/2003/10/Serialization/>">#</int>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetRecordCount.json http://keylight.lockpath.com:4443/ComponentService/GetRecordCount

### JSON REQUEST (GetRecordCount.json)

{

"componentId": "10001", "filters": [

{

"FieldPath": [ 11

],

"FilterType": "1", "Value": "text value"

}

]

}

### JSON RESPONSE

#

## GetDetailRecord

#### Retrieves record information based on the provided component ID and record ID, with lookup field report details. Lookup field records will detail information for fields on their report definition, if one is defined. Using the optional boolean parameter "embedRichTextImages" you can extract images contained in rich text fields.

URL: "http://[instance name]:[port]/ComponentService/GetDetailRecord?componentId=

{COMPONENTID}&recordId={RECORDID}&embedRichTextImages=true" Method: GET

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component Permissions: The authentication account must have Read General Access permissions to:

* + Selected component
  + Selected record
  + Applicable fields in the component (table)

**Examples**

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8"

"http://keylight.lockpath.com:4443/ComponentService/GetDetailRecord?componentId=12345&record Id=1&embedRichTextImages=true"

**XML RESPONSE**

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>1</Id>

<DisplayName>The First Record</DisplayName>

<FieldValues>

<KeyValuePair>

<key>1234</key>

<value i:type="a:string" xmlns:a="<http://www.w3.org/2001/XMLSchema>">Text Field Content</value>

</KeyValuePair>

<KeyValuePair>

<key>1235</key>

<value i:type="DynamicRecordList">

<Record>

<Id>1</Id>

<DisplayName>Lookup Record 1</DisplayName>

<FieldValues/>

<LookupReportColumns>

<Column>

<FieldPathId>2345</FieldPathId>

<ColumnName>Title</ColumnName>

<Value>Lookup Record 1</Value>

</Column>

<Column>

<FieldPathId>2346</FieldPathId>

<ColumnName>Workflow Stage: Workflow: Name</ColumnName>

<Value>Default Workflow</Value>

</Column>

</LookupReportColumns>

</Record>

</value>

</KeyValuePair>

</FieldValues>

</DynamicRecordItem>

<Record>

<Id>2</Id>

<DisplayName>Lookup Record 2</DisplayName>

<FieldValues/>

<LookupReportColumns>

<Column>

<FieldPathId>2347</FieldPathId>

<ColumnName>Title</ColumnName>

<Value>Lookup Record 2</Value>

</Column>

<Column>

<FieldPathId>2348</FieldPathId>

<ColumnName>Workflow Stage: Workflow: Name</ColumnName>

<Value>Default Workflow</Value>

</Column>

</LookupReportColumns>

</Record>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" https://keylight.lockpath.com:4443/ComponentService/GetDetailRecord?componentId=10001&record Id=1&embedRichTextImages=true"

### JSON RESPONSE

{

"Id": 1,

"DisplayName": "192.168.1.84", "FieldValues": [

{

"Key": 3852,

"Value": 1

},

{

"Key": 3853,

"Value": 8

},

{

"Key": 4982,

"Value": {

" type": "DynamicRecordItem", "Id": 219,

"DisplayName": "192.168.30.22", "FieldValues": [], "LookupReportColumns": [

{

"FieldPathId": 650, "ColumnName": "DNS Name", "Value": "machine1.test.com"

},

{

"FieldPathId": 651, "ColumnName": "MAC Address",

"Value": "11:70:5B:91:63:35"

},

{

"FieldPathId": 649, "ColumnName": "IP Address", "Value": "192.168.30.22"

}

]

}

},

{

"Key": 4983,

"Value": false

}

}

## GetDetailRecords

#### GetDetailRecords provides the ability to run a search with filters and paging (GetRecords) while returning a high level of detail for each record (GetRecord). GetDetailRecords also allows multiple sorts to modify the order of the results. For performance and security concerns, the maximum number of records returned (pageSize) is 1000.

URL: http://[instance name]:[port]/ComponentService/GetDetailRecords Method: POST

Input: componentID (Integer): The ID of the desired component

pageIndex (Integer): The index of the page of result to return. Must be >= 0 pageSize (Integer): The size of the page results to return. Must be >= 1 fieldIds (optional) (Integer): The ID of the field to be returned. If not provided, returns all

accessible fields. If provided, but empty, returns core system fields (CreatedAt, CreatedBy, etc.). If provided, returns core system fields plus accessible fields. Note that system fields will always be returned regardless.

Permissions: The authentication account must have Read General Access permissions to:

* + Selected component
  + Selected record
  + Applicable fields in the component (table)

**Examples**

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetDetailRecords.xml http://keylight.lockpath.com:4443/ComponentService/GetDetailRecords

#### XML REQUEST (GetDetailRecords.xml)

<GetDetailRecords>

<componentId>12345</componentId>

<pageIndex>0</pageIndex>

<pageSize>2</pageSize>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>1234</int>

</FieldPath>

<FilterType>7</FilterType>

<Value>10</Value>

</SearchCriteriaItem>

</filters>

<sortOrder>

</sortOrder>

<fieldIds>

</fieldIds>

</GetDetailRecords>

**XML RESPONSE**

<SortCriteriaItem>

<FieldPath>

<int>2345</int>

</FieldPath>

<Ascending>false</Ascending>

</SortCriteriaItem>

<SortCriteriaItem>

<FieldPath>

<int>6789</int>

</FieldPath>

<Ascending>true</Ascending>

</SortCriteriaItem>

<int>1234</int>

<int>5678</int>

<ArrayOfDynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>1</Id>

<DisplayName>The First Record</DisplayName>

<FieldValues>

<KeyValuePair>

<key>1234</key>

<value i:type="a:string" xmlns:a="<http://www.w3.org/2001/XMLSchema>">Text Field Content</value>

</KeyValuePair>

<KeyValuePair>

<key>1235</key>

<value i:type="DynamicRecordList">

<Record>

<Id>1</Id>

<DisplayName>Lookup Record 1</DisplayName>

<FieldValues/>

</Record>

<Record>

<Id>2</Id>

<DisplayName>Lookup Record 2</DisplayName>

<FieldValues/>

</Record>

</value>

</KeyValuePair>

</FieldValues>

</DynamicRecordItem>

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>2</Id>

<DisplayName>The Second Record</DisplayName>

<FieldValues>

<KeyValuePair>

<key>1234</key>

<value i:type="a:string" xmlns:a="<http://www.w3.org/2001/XMLSchema>">Text Field Content</value>

</KeyValuePair>

<KeyValuePair>

<key>1235</key>

<value i:type="DynamicRecordList">

<Record>

<Id>3</Id>

<DisplayName>Lookup Record 3</DisplayName>

<FieldValues/>

</Record>

<Record>

<Id>4</Id>

<DisplayName>Lookup Record 4</DisplayName>

<FieldValues/>

</Record>

</value>

</KeyValuePair>

</FieldValues>

</DynamicRecordItem>

</ArrayOfDynamicRecordItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @GetDetailRecords.json http://keylight.lockpath.com:4443/ComponentService/GetDetailRecords

### JSON REQUEST (GetDetailRecords.json)

{

"componentId": "10001",

"pageIndex": "0",

"pageSize": "1000", "filters": [

{

"FieldPath": [ 3881

],

"FilterType": "3",

"Value": "Blue"

}

],

"sortOrder": [

{

"FieldPath": [ 4991

],

"Ascending": "true"

}

],

"fieldIds": [2500,2502]

}

### JSON RESPONSE

[

{

"Id": 1,

"DisplayName": "Record 1", "FieldValues": [

{

"Key": 2500,

"Value": 4

},

{

"Key": 2502,

"Value": {

" type": "DynamicRecordItem", "Id": 10,

"DisplayName": "Admin, User", "FieldValues": []

}

}

]

},

{

"Id": 2,

"DisplayName": "Record 2", "FieldValues": [

{

"Key": 2500,

"Value": 7

},

{

"Key": 2502,

"Value": {

" type": "DynamicRecordItem",

"Id": 11,

"DisplayName": "End, User", "FieldValues": []

}

}

]

}

]

## GetRecordAttachment

#### Gets a single attachment associated with the provided component ID, record ID, documents field ID, and document ID. The file contents are returned as a Base64 string.

URL: "http://[instance name]:[port]/ComponentService/GetRecordAttachment?componentId=

{COMPONENTID}&recordId={RECORDID}&fieldId={FIELDID}&documentId={DOCUMENTID}" Method: GET

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component fieldId (Integer): The ID for the individual field within the component documentId (Integer): The ID for the individual document within the component

Permissions: The authentication account must have Read General Access permissions to:

* Selected component
* Selected record
* Applicable fields in the component (table)

**Examples**

The GetRecordAttachment retrieves a single attachment for a given field on a record. GetRecordAttachment uses the default GET method, so the method does not need to be specified in the request. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8"

"http://keylight.lockpath.com:4443/ComponentService/GetRecordAttachment?componentId=12345&re cordId=1&fieldId=1234&documentId=1"

**XML RESPONSE**

<RecordAttachmentItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<FileName>Attachment1.txt</FileName>

<FileData>Q2hyb25vIGNhbXBhaWduDQoNCi0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLQ0KDQpQbGF5ZXJ zOg0KLSBSZXB0aXRlIFNvc=</FileData>

</RecordAttachmentItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" "http://keylight.lockpath.com:4443/ComponentService/GetRecordAttachment?componentId=12345&re cordId=1&fieldId=1234&documentId=1"

### JSON RESPONSE

{

"FileName": "Attachment1.txt", "FileData":

"Q2hyb25vIGNhbXBhaWduDQoNCi0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLQ0KDQpQbGF5ZXJ\nzOg0KLS BSZXB0aXRlIFNvc="

}

## GetRecordAttachments

#### Gets information for all attachments associated with the provided component ID, record ID, and Documents field id. No file data is returned, only file name, field ID, and document ID information.

URL: "http://[instance name]:[port]/ComponentService/GetRecordAttachments?componentId=

{COMPONENTID}&recordId={RECORDID}&fieldId={FIELDID}"

Method: GET

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component fieldId (Integer): The ID for the individual field within the component

Permissions: The authentication account must have Read General Access permissions to:

* Selected component
* Selected record
* Applicable fields in the component (table)

**Examples**

The GetRecordAttachments retrieves a list of all of the attachments for a given field on a record. GetRecordAttachments uses the default GET method, so the method does not need to be specified in the request. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8"

"http://keylight.lockpath.com:4443/ComponentService/GetRecordAttachments?componentId=12345&r ecordId=1&fieldId=1234"

**XML RESPONSE**

<AttachmentInfoList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<AttachmentInfo>

<FileName>Attachment1.txt</FileName>

<FieldId>1234</FieldId>

<DocumentId>1</DocumentId>

</AttachmentInfo>

<AttachmentInfo>

<FileName>Attachment2.xml</FileName>

<FieldId>1234</FieldId>

<DocumentId>2</DocumentId>

</AttachmentInfo>

</AttachmentInfoList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" "http://keylight.lockpath.com:4443/ComponentService/GetRecordAttachments?componentId=12345&r ecordId=1&fieldId=1234"

### JSON RESPONSE

[

{

"FileName": "Attachment1.txt", "FieldId": "1234",

"DocumentId": "1"

},

{

"FileName": "Attachment2.xml", "FieldId": "1234",

"DocumentId": "2"

}

]

## GetWorkflow

#### URL: http://[instancename]:[port]/ComponentService/GetWorkflow?id=[ID] Method: GET

Input: ID: The ID of the desired workflow

Permissions: The authentication account must have: Read Administrative Access permission for the specific component enabled.

Retrieves workflow details and all workflow stages specified by ID. The ID for a workflow may be found by using GetWorkflows.

**Examples**

This sample obtains the details for a workflow and all workflow stages. GetWorkflow uses the default GET method, so the method does not need to be specified in the request. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetWorkflow?id=1

**XML RESPONSE**

<WorkflowItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>1</Id>

<Name>Default Workflow</Name>

<Description/>

<IsActive>true</IsActive>

<IsDefault>true</IsDefault>

<RoutingCriteria/>

<WorkflowOwnerGroups/>

<WorkflowOwnerUsers/>

<WorkflowStages>

<WorkflowStage>

<Id>1</Id>

<Name>Grammatical Review</Name>

<Description/>

<IsInitial>true</IsInitial>

<IsActive>true</IsActive>

<IsVoting>false</IsVoting>

<GroupAccess>

<Permission>

<Group>

<Id>8</Id>

<Name>Compliance Document Authors</Name>

<IsBusinessUnit>false</IsBusinessUnit>

</Group>

<CanViewAll>false</CanViewAll>

<CanEdit>true</CanEdit>

<CanTransition>true</CanTransition>

</Permission>

<Permission>

<Group>

<Id>9</Id>

<Name>Compliance Document Approvers</Name>

<IsBusinessUnit>false</IsBusinessUnit>

</Group>

<CanViewAll>true</CanViewAll>

<CanEdit>true</CanEdit>

<CanTransition>true</CanTransition>

</Permission>

</GroupAccess>

<UserAccess/>

<UseAssignments>true</UseAssignments>

<UseAssignmentValues>

<UseAssignment>

<AssignmentFieldPath>

<Field>

<Id>10</Id>

<Name>Created By</Name>

<SystemName>CreatedBy</SystemName>

<FieldType>5</FieldType>

</Field>

</AssignmentFieldPath>

<CanAssigneeEdit>true</CanAssigneeEdit>

<CanAssigneeTransition>true</CanAssigneeTransition>

</UseAssignment>

<UseAssignment>

<AssignmentFieldPath>

<Field>

<Id>11</Id>

<Name>Updated By</Name>

<SystemName>UpdatedBy</SystemName>

<FieldType>5</FieldType>

</Field>

</AssignmentFieldPath>

<CanAssigneeEdit>true</CanAssigneeEdit>

<CanAssigneeTransition>true</CanAssigneeTransition>

</UseAssignment>

</UseAssignmentValues>

<Transitions>

<Transition>

<Id>1</Id>

<Label>Promote</Label>

<ToStage>

<Id>2</Id>

<Name>Signoff</Name>

</ToStage>

</Transition>

</Transitions>

<CanAutoApprove>false</CanAutoApprove>

</WorkflowStage>

</WorkflowStages>

</WorkflowItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" http://keylight.lockpath.com:4443/ComponentService/GetWorkflow?id=1

### JSON RESPONSE

{

"Id": "1",

"Name": "Default Workflow", "IsActive": "true",

"IsDefault": "true", "RoutingCriteria": [], "WorkflowOwnerGroups": [], "WorkflowOwnerUsers": [], "WorkflowStages": {

"WorkflowStage": { "Id": "1",

"Name": "Grammatical Review", "Description": [],

"IsInitial": "true",

"IsActive": "true",

"IsVoting": "false", "GroupAccess": [

{

"Group": { "Id": "8",

"Name": "Compliance Document Authors", "IsBusinessUnit": "false"

},

"CanViewAll": "false", "CanEdit": "true", "CanTransition": "true"

},

{

"Group": { "Id": "9",

"Name": "Compliance Document Approvers", "IsBusinessUnit": "false"

},

"CanViewAll": "true", "CanEdit": "true", "CanTransition": "true"

}

],

"UserAccess": [], "UseAssignments": true, "UseAssignmentValues": [

{

"AssignmentFieldPath": [

{

"Id": 8182,

"Name": "Created By", "SystemName": "CreatedBy", "FieldType": 5

}

],

"CanAssigneeEdit": true, "CanAssigneeTransition": true

},

{

"AssignmentFieldPath": [

{

"Id": 8185,

"Name": "Updated By", "SystemName": "UpdatedBy", "FieldType": 5

}

],

"CanAssigneeEdit": true, "CanAssigneeTransition": true

}

],

"Transitions": [

{

"Id": 73,

"Label": "Approve", "ToStage": {

"Id": 302,

"Name": "Stage 2"

}

}

],

"CanAutoApprove": false

},

{

"Id": 302,

"Name": "Stage 2",

"Description": "", "IsInitial": false, "IsActive": true, "IsVoting": false, "GroupAccess": [

{

"Group": {

"Id": 0,

"Name": "Everyone"

},

"CanViewAll": true, "CanEdit": true, "CanTransition": true

}

],

"UserAccess": [], "UseAssignments": true, "UseAssignmentValues": [

{

"AssignmentFieldPath": [

{

"Id": 8182,

"Name": "Created By", "SystemName": "CreatedBy", "FieldType": 5

}

],

"CanAssigneeEdit": true, "CanAssigneeTransition": true

},

{

"AssignmentFieldPath": [

{

"Id": 8185,

"Name": "Updated By", "SystemName": "UpdatedBy", "FieldType": 5

}

],

"CanAssigneeEdit": true, "CanAssigneeTransition": true

}

],

"Transitions": [

{

"Id": 75,

"Label": "Approve", "ToStage": {

"Id": 300,

"Name": "Published"

}

},

{

"Id": 74,

"Label": "Reject", "ToStage": {

"Id": 301,

"Name": "Stage 1"

}

}

],

"CanAutoApprove": false

}

]

}

## GetWorkflows

#### Retrieves all workflows for a component specified by its Alias. A component is a user-defined data object such as a custom content table. The component Alias may be found by using GetComponentList (ShortName).

URL: http://[instancename]:[port]/ComponentService/GetWorkflows?componentalias=[Alias] Method: GET

Input: Alias (String): The Alias of the desired component

Permissions: The authentication account must have: Read Administrative Access permission for the specific component enabled.

**Examples**

This sample obtains a list of the workflows for a specified component within the Keylight Platform. GetWorkflows uses the default GET method, so the method does not need to be specified in the request. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" http://keylight.lockpath.com:4443/ComponentService/GetWorkflows?componentalias=Devices

**XML RESPONSE**

<WorkflowList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Workflow>

<Id>2</Id>

<Name>Default</Name>

<IsActive>true</IsActive>

<IsDefault>true</IsDefault>

</Workflow>

</WorkflowList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "Accept: application/json" http://keylight.lockpath.com:4443/ComponentService/GetWorkflows?componentalias=Devices

### JSON RESPONSE

[

{

"Id": 2,

"Name": "Default", "IsActive": true, "IsDefault": true

}

]

## TransitionRecord

#### Transition a record in a workflow stage.

URL: http://[instance name]:[port]/ComponentService/TransitionRecord Method: POST

Input: tableAlias (string): The Alias for the table

recordId (integer): The ID of the record to be transitioned

transitionId (integer): The ID of the workflow stage transition, which can be retrieved

with GetWorkflow

Permissions: The authentication account must have Read and Update General Access permissions to the defined table, View and Transition workflow stage permissions, and record permission.

**Examples**

TransitionRecord transitions a record in a workflow stage. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @TransitionRecord.xml

"http://keylight.lockpath.com:4443/ComponentService/TransitionRecord"

**XML REQUEST (TransitionRecord.xml)**

<TransitionRecord>

<tableAlias>Devices</tableAlias>

<recordId>2</recordId>

<transitionId>42</transitionId>

</TransitionRecord>

### XML RESPONSE

true

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @TransitionRecord.json http://keylight.lockpath.com:4443/ComponentService/TransitionRecord

### JSON REQUEST (TransitionRecord.json)

{

"tableAlias": "Devices", "recordId": "2",

"transitionId": "42"

}

### JSON RESPONSE

true

## VoteRecord

#### Cast a vote for a record in a workflow stage.

URL: http://[instance name]:[port]/ComponentService/VoteRecord Method: POST

Input: tableAlias (string): The Alias for the table

recordId (integer): The ID of the record to be transitioned

transitionId (integer): The ID of the workflow stage voting rule, which can be retrieved

with GetWorkflow votingComments (string): Voting comments

Permissions: The authentication account must have Read and Update General Access permissions to the defined table, View and Vote workflow stage permissions, and record permission.

**Examples**

VoteRecord casts a vote for a record in a workflow stage. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @VoteRecord.xml

"http://keylight.lockpath.com:4443/ComponentService/VoteRecord"

**XML REQUEST (VoteRecord.xml)**

<VoteRecord>

<tableAlias>Devices</tableAlias>

<recordId>4</recordId>

<transitionId>46</transitionId>

<votingComments>idk</votingComments>

</VoteRecord>

### XML RESPONSE

true

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @VoteRecord.json http://keylight.lockpath.com:4443/ComponentService/VoteRecord

### JSON REQUEST (VoteRecord.json)

{

"tableAlias": "Devices", "recordId": "4",

"transitionId": "46", "votingComments": "idk"

}

### JSON RESPONSE

true

## CreateRecord

#### Create a new record within the specified component of the Keylight application.

**NOTE:** The Required option for a field is only enforced through the user interface, not through the API. Therefore, CreateRecord does not enforce the Required option for fields.

URL: http://[instance name]:[port]/ComponentService/CreateRecord Method: POST

Input: componentId (int): The ID of the desired component dynamicRecord (DynamicRecordItem): Fields for the component to be created.

Dynamic Record:

A dynamic record is defined by key-value pairs that contain field ID (that data will be entered into) and value contains the type (string, decimal, etc.) of the data and the data itself. Sample xml is shown below with an entry for each applicable field type.

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<FieldValues>

<!--Text-->

<KeyValuePair>

<key>3664</key>

<value i:type="a:**string**">See Spot Run</value>

</KeyValuePair>

<!--Numeric-->

<KeyValuePair>

<key>3667</key>

<value i:type="a:**decimal**">25.13</value>

</KeyValuePair>

<!--IP Address-->

<KeyValuePair>

<key>3668</key>

<value i:type="a:**string**">192.168.1.3</value>

</KeyValuePair>

<!--Yes/No-->

<KeyValuePair>

<key>3670</key>

<value i:type="a:**boolean**">true</value>

</KeyValuePair>

<!--Date-->

<KeyValuePair>

<key>3719</key>

<value i:type="a:**dateTime**">2014-09-19T11:02:46</value>

</KeyValuePair>

<!--1:1 Lookup-->

<KeyValuePair>

<key>3667</key>

<value i:type='**DynamicRecordItem**'><Id>18</Id></value>

</KeyValuePair>

<!--1:M Lookup-->

<KeyValuePair>

<key>3720</key>

<value i:type='DynamicRecordList'>

<Record><Id>13</Id></Record>

<Record><Id>20</Id></Record>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

#### Empty Fields: If a field is not included in the input for CreateRecord, the field remains empty/null for the created record.

Another method to create empty/null data for a field that is defined in the input is to use:

<value i:nil="true" /> as the **value**

Matrix: Matrix records can only be created after the parent record has been created. The Matrix componentId can be retrieved with GetComponentList and the Matrix Column fields can be retrieved with GetFieldList, using the Matrix componentId. Matrix Row ID’s must be retrieved from the table in the Keylight database.

Master/Detail: Master/Detail records can only be created after the parent record has been created. The Master/Detail componentId can be retrieved with GetComponentList and the master/detail subfields can be retrieved with GetFieldList, using the Master/Detail componentId.

Workflow Stage The key is the Workflow Stage field ID and the Id is Workflow Stage ID. ID:

<KeyValuePair>

<key>3849</key>

<value i:type="DynamicRecordItem"><Id>109</Id></value>

</KeyValuePair>

#### Permissions: The authentication account must have:

* + Create General Access permission to the selected component
  + Edit permission to any field into which data is to be entered

System Fields: Keylight Platform tracks system fields for each record. The user should enter values for their custom created fields and the platform will populate the system fields. The values for system fields will be returned in the response.

|  |  |
| --- | --- |
| **System Fields** | |
| **Field** | **Value Source** |
| CreatedAt | TimeStamp Record Created |
| CreatedBy | User Id (API logon) |
| Id | Unique Record Identifier |
| CurrentRevision | Tracks Revision History Starts at 0 |
| UpdatedAt | TimeStamp Record Updated |
| UpdatedBy | User Id (API logon) |
| Workflow Stage | Workflow Stage for Record Defined in the platform |

**Examples**

The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @CreateRecordInput.xml http://keylight.lockpath.com:4443/ComponentService/CreateRecord

**XML REQUEST (CreateRecord.xml)**

<CreateRecord>

<componentId>10001</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<FieldValues>

<KeyValuePair>

<key>11</key>

<value i:type="a:string">192168001001</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</CreateRecord>

### XML RESPONSE

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>1</Id>

<DisplayName/>

<FieldValues>

<KeyValuePair>

<key>10</key>

<value i:nil="true"/>

</KeyValuePair>

<KeyValuePair>

<key>1628</key>

<value i:type="DynamicRecordList"/>

</KeyValuePair>

<KeyValuePair>

<key>301</key>

<value i:type="a:dateTime">2014-07-11T10:38:17.8901765-06:00</value>

</KeyValuePair>

<KeyValuePair>

<key>303</key>

<value i:type="DynamicRecordItem">

<Id>11</Id>

<DisplayName>Last, First</DisplayName>

<FieldValues/>

</value>

</KeyValuePair>

</FieldValues>

</DynamicRecordItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @CreateRecord.json http://keylight.lockpath.com:4443/ComponentService/CreateRecord

### JSON REQUEST (CreateRecord.json)

{

"componentId": "10001", "dynamicRecord": {

"FieldValues": [

{

"key": "3",

"value": "API Example DNS Name"

},

{

"key": "9",

"value": 123

},

{

"key": "10",

"value": "12/25/2017"

},

{

"key": "11",

"value": "1.2.3.4"

},

{

"key": "4879",

"value": true

},

{

"key": "23",

"value":

{

"Id": "1"

}

},

{

"key": "294",

"value":

[

{

"Id": "2"

},

{

"Id": "3"

}

]

}

]

}

}

### JSON RESPONSE

GetRecord

### Master/Detail Examples

#### The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @CreateMDRecordInput.xml http://keylight.lockpath.com:4444/ComponentService/CreateRecord

### XML REQUEST (CreateMDRecord.xml)

<CreateRecord>

<!--Master/Detail-->

<componentId>10199</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<FieldValues>

<!--Parent Record Field-->

<KeyValuePair>

<key>3750</key>

<value i:type="a:**int**">5</value>

</KeyValuePair>

<!--Master/Detail Text-->

<KeyValuePair>

<key>3664</key>

<value i:type="a:**string**">See Spot Run</value>

</KeyValuePair>

<!--Master/Detail Numeric-->

<KeyValuePair>

<key>3667</key>

<value i:type="a:**decimal**">25.13</value>

</KeyValuePair>

<!--Master/Detail Date-->

<KeyValuePair>

<key>3719</key>

<value i:type="a:**dateTime**">2014-09-19T11:02:46</value>

</KeyValuePair>

<!--IP Address-->

<KeyValuePair>

<key>3668</key>

<value i:type="a:**string**">192.168.1.3</value>

</KeyValuePair>

<!--Master/Detail Yes/No-->

<KeyValuePair>

<key>3670</key>

<value i:type="a:**boolean**">true</value>

</KeyValuePair>

<!--Master/Detail 1:1 Lookup-->

<KeyValuePair>

<key>3667</key>

<value i:type='**DynamicRecordItem**'><Id>18</Id></value>

</KeyValuePair>

<!--Master/Detail 1:M Lookup-->

<KeyValuePair>

<key>3720</key>

<value i:type='**DynamicRecordList**'>

<Record><Id>13</Id></Record>

<Record><Id>20</Id></Record>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</CreateRecord>

## UpdateRecord

#### Update fields in a specified record.

**NOTE:** The Required option for a field is only enforced through the user interface, not through the API. Therefore, UpdateRecord does not enforce the Required option for fields. The response will include the complete set of fields for the specified record.

URL: http://[instance name]:[port]/ComponentService/UpdateRecord Method: POST

Input: componentId (int): The ID of the desired component dynamicRecord (DynamicRecordItem): Fields for the component to be created.

Dynamic Record:

A dynamic record is defined by key-value pairs that contain field ID (that data will be entered into) and value contains the type (string, decimal, etc.) of the data and the data itself. Sample xml is shown below with an entry for each applicable field type.

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<FieldValues>

<!--Text-->

<KeyValuePair>

<key>3664</key>

<value i:type="a:**string**">See Spot Run</value>

</KeyValuePair>

<!--Numeric-->

<KeyValuePair>

<key>3667</key>

<value i:type="a:**decimal**">25.13</value>

</KeyValuePair>

<!--Date-->

<KeyValuePair>

<key>3719</key>

<value i:type="a:**dateTime**">2014-09-19T11:02:46</value>

</KeyValuePair>

<!--IP Address-->

<KeyValuePair>

<key>3668</key>

<value i:type="a:**string**">192.168.1.3</value>

</KeyValuePair>

<!--Yes/No-->

<KeyValuePair>

<key>3670</key>

<value i:type="a:**boolean**">true</value>

</KeyValuePair>

<!--1:1 Lookup-->

<KeyValuePair>

<key>3667</key>

<value i:type='**DynamicRecordItem**'><Id>18</Id></value>

</KeyValuePair>

<!--1:M Lookup-->

<KeyValuePair>

<key>3720</key>

<value i:type='DynamicRecordList'>

<Record><Id>13</Id></Record>

<Record><Id>20</Id></Record>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

#### Empty Fields: To empty/null data for a field that is defined in the input, use <value i:nil="true" /> as the value.

Matrix: The Matrix componentId can be retrieved with GetComponentList and the Matrix Column fields can be retrieved with GetFieldList, using the Matrix componentId. Matrix Row ID’s must be retrieved from the table in the Keylight database.

Master/Detail: The Master/Detail componentId can be retrieved with GetComponentList and the master/detail subfields can be retrieved with GetFieldList, using the Master/Detail componentId.

Workflow Stage The key is the Workflow Stage field ID and the Id is Workflow Stage ID. ID:

<KeyValuePair>

<key>3849</key>

<value i:type="DynamicRecordItem"><Id>109</Id></value>

</KeyValuePair>

#### Permissions: The authentication account must have:

* + Read/Update General Access permission to the selected component
  + Edit permission for fields to be updated
  + Read permission to the selected record

**Examples**

The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @UpdateRecordInput.xml http://keylight.lockpath.com:4443/ComponentService/UpdateRecord

**XML REQUEST (UpdateRecord.xml)**

<UpdateRecord>

<componentId>10001</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<Id>1</Id>

<FieldValues>

<KeyValuePair>

<key>9</key>

<value i:type="a:decimal">100.00</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</UpdateRecord>

### XML RESPONSE

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>1</Id>

<DisplayName>192.168.1.84</DisplayName>

<FieldValues>

<KeyValuePair>

<key>10</key>

<value i:nil="true"/>

</KeyValuePair>

<KeyValuePair>

<key>9</key>

<value i:type="a:decimal">100.00</value>

</KeyValuePair>

</FieldValues/>

</DynamicRecordItem>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @UpdateRecord.json http://keylight.lockpath.com:4443/ComponentService/UpdateRecord

### JSON REQUEST (UpdateRecord.json)

{

"componentId": "10001", "dynamicRecord": {

"Id": "2",

"FieldValues": [

{

"key": "3",

"value": "API Example DNS Name updated"

},

{

"key": "9",

"value": 1234

},

{

"key": "10",

"value": "12/25/2018"

},

{

"key": "11",

"value": "1.2.3.5"

},

{

"key": "4879",

"value": false

},

{

"key": "23",

"value":

{

}

"Id": "1"

},

{

"key": "294",

"value":

[

{

"Id": "4"

},

{

"Id": "5"

}

]

}

]

}

}

### JSON RESPONSE

GetRecord

### Matrix Fields

#### Matrix fields can only be updated after the parent record is created. The essential components of a Matrix field update are:

componentId: The ID of the Matrix field found in GetComponentList. KeyValue Pairs: Creating a matrix row:

* + Parent Record Id, found in GetFieldList for the matrix field componentId
  + Value, parent record Id KeyValue Pairs: Updating a matrix row:
  + The component matrix cell Id for that entry
  + Matrix Row Id, found in the get field list for the matrix component
  + Value, matrix row signifier from within the matrix, found in the get field list for the parent record component
  + Matrix Column Id, found in the get field list for the matrix component
  + Value(s), actual entry into the intended matrix cell(s)

Only one Matrix Row Id/record is updated per script and the script must change between updating a matrix that has no values in a cell versus updating a matrix that holds an existing value.

**Matrix Examples**

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @MatrixUpdate.xml http://keylight.lockpath.com:4444/ComponentService/UpdateRecord

### XML REQUEST (MatrixUpdate.xml)

**Updating an empty Matrix field:**

<UpdateRecord>

<componentId>10164</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<FieldValues>

<KeyValuePair>

<key>2918</key>

<value i:type="a:int">5</value>

</KeyValuePair>

<KeyValuePair>

<key>2919</key>

<value i:type="a:int">101</value>

</KeyValuePair>

<KeyValuePair>

<key>2921</key>

<value i:type="a:decimal">8888</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</UpdateRecord>

**Updating a Matrix field with an existing value:**

<UpdateRecord>

<componentId>10164</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<Id>7</Id>

<FieldValues>

<KeyValuePair>

<key>2921</key>

<value i:type="a:decimal">333</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</UpdateRecord>

### XML RESPONSE

<DynamicRecordItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<Id>11</Id>

<DisplayName>11</DisplayName>

<FieldValues>

<KeyValuePair>

<key>2911</key>

<value i:type="a:int">11</value>

</KeyValuePair>

<KeyValuePair>

<key>2912</key>

<value i:type="a:dateTime">2014-05-06T11:34:27.8017443-05:00</value>

</KeyValuePair>

<KeyValuePair>

<key>2913</key>

<value i:type="a:dateTime">2014-05-06T11:34:27.8017443-05:00</value>

</KeyValuePair>

<KeyValuePair>

<key>2914</key>

<value i:type="DynamicRecordItem">

<Id>72</Id>

<DisplayName>Frank, Irma</DisplayName>

<FieldValues/>

</value>

</KeyValuePair>

<KeyValuePair>

<key>2915</key>

<value i:type="DynamicRecordItem">

<Id>72</Id>

<DisplayName>Frank, Irma</DisplayName>

<FieldValues/>

</value>

</KeyValuePair>

<KeyValuePair>

<key>2918</key>

<value i:type="DynamicRecordItem">

<Id>6</Id>

<DisplayName/>

<FieldValues/>

</value>

</KeyValuePair>

<KeyValuePair>

<key>2919</key>

<value i:type="DynamicRecordItem">

<Id>100</Id>

<DisplayName>Row 1</DisplayName>

<FieldValues/>

</value>

</KeyValuePair>

<KeyValuePair>

<key>2921</key>

<value i:type="a:decimal" xmlns:a="<http://www.w3.org/2001/XMLSchema>">2222</value>

</KeyValuePair>

<KeyValuePair>

<key>3587</key>

<value i:type="a:int" xmlns:a="<http://www.w3.org/2001/XMLSchema>">1</value>

</KeyValuePair>

</FieldValues>

</DynamicRecordItem>

### Master/Detail Examples

#### The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @UpdateMDRecordInput.xml http://keylight.lockpath.com:4444/ComponentService/UpdateRecord

### XML REQUEST (UpdateMDRecord.xml)

<UpdateRecord>

<!--Master/Detail-->

<componentId>10199</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<Id>5</Id>

<FieldValues>

<!--Master/Detail Text-->

<KeyValuePair>

<key>3664</key>

<value i:type="a:**string**">See Spot Run</value>

</KeyValuePair>

<!--Master/Detail Numeric-->

<KeyValuePair>

<key>3667</key>

<value i:type="a:**decimal**">25.13</value>

</KeyValuePair>

<!--Master/Detail Date-->

<KeyValuePair>

<key>3719</key>

<value i:type="a:**dateTime**">2014-09-19T11:02:46</value>

</KeyValuePair>

<!--IP Address-->

<KeyValuePair>

<key>3668</key>

<value i:type="a:**string**">192.168.1.3</value>

</KeyValuePair>

<!--Master/Detail Yes/No-->

<KeyValuePair>

<key>3670</key>

<value i:type="a:**boolean**">true</value>

</KeyValuePair>

<!--Master/Detail 1:1 Lookup-->

<KeyValuePair>

<key>3667</key>

<value i:type='**DynamicRecordItem**'><Id>18</Id></value>

</KeyValuePair>

<!--Master/Detail 1:M Lookup-->

<KeyValuePair>

<key>3720</key>

<value i:type='**DynamicRecordList**'>

<Record><Id>13</Id></Record>

<Record><Id>20</Id></Record>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</UpdateRecord>

## UpdateRecordAttachments

#### Adds new attachments and/or updates existing attachments to the provided Documents field(s) on a specific record, where the FileData is represented as a Base64 string. The maximum data size of the request is controlled by the maxAllowedContentLength and maxReceivedMessageSize values in the API web.config.

URL: http://[instance name]:[port]/ComponentService/UpdateRecordAttachments Method: POST

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component fieldId (Integer): The ID for the individual field within the component

Master/Detail: The Master/Detail componentId can be retrieved with GetComponentList and the master/detail subfields can be retrieved with GetFieldList, using the Master/Detail componentId.

Permissions: The authentication account must have Read and Update General Access permissions to:

* + Selected component
  + Selected record
  + Applicable fields in the component (table)

**Examples**

The UpdateRecordAttachments adds new attachments and/or updates existing attachments to the provided document fields on a record. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @UpdateRecordAttachments.xml

http://keylight.lockpath.com:4443/ComponentService/UpdateRecordAttachments

**XML REQUEST (UpdateRecordAttachments.xml)**

<UpdateRecordAttachments>

<componentId>12345</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<Id>1</Id>

<FieldValues>

<KeyValuePair>

<key>1234</key>

<value i:type='RecordAttachmentList'>

<Attachment>

<FileName>Attachment1.txt</FileName>

<FileData>SGVsbG8gV29ybGQ=</FileData>

</Attachment>

<Attachment>

<FileName>Attachment2.xml</FileName>

<FileData>SGVsbG8gV29ybGQ=</FileData>

</Attachment>

</value>

</KeyValuePair>

<KeyValuePair>

<key>5678</key>

<value i:type='RecordAttachmentList'>

<Attachment>

<FileName>Attachment3.txt</FileName>

<FileData>SGVsbG8gV29ybGQ=</FileData>

</Attachment>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</UpdateRecordAttachments>

### XML RESPONSE

<AttachmentOperationResultList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<AttachmentOperationResult>

<OperationSucceeded>true</OperationSucceeded>

<Message>Attachment was successfully added to the Documents field.</Message>

<ComponentId>12345</ComponentId>

<RecordId>1</RecordId>

<AttachmentInfo>

<FileName>Attachment1.txt</FileName>

<FieldId>1234</FieldId>

<DocumentId>1</DocumentId>

</AttachmentInfo>

</AttachmentOperationResult>

<AttachmentOperationResult>

<OperationSucceeded>true</OperationSucceeded>

<Message>Attachment was successfully added to the Documents field.</Message>

<ComponentId>12345</ComponentId>

<RecordId>1</RecordId>

<AttachmentInfo>

<FileName>Attachment2.xml</FileName>

<FieldId>1234</FieldId>

<DocumentId>2</DocumentId>

</AttachmentInfo>

</AttachmentOperationResult>

</AttachmentOperationResultList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @UpdateRecordAttachments.json http://keylight.lockpath.com:4443/ComponentService/UpdateRecordAttachments

### JSON REQUEST (UpdateRecordAttachments.json)

{

"componentId": "10001", "dynamicRecord": {

"Id": "2",

"FieldValues": [

{

"key": "34",

"value": [

{

"FileName": "import\_temp.csv", "FileData":

"VGV4dCxudW0scmljaHksaXB2NCxpcHY2LGRhdGUsZGF0ZXRpbWUNCnIxLDUxMCw8Yj5ib2xkIHRoaW5nPC9iPiwxOTI uMTY4LjIuNCwwMDE6MGRiODowMDAwOjAwNDI6MDAwMDo4YTJlOjAzNzA6NzMzNCwyLzE4LzE5ODYsMi8xOS8xOTg2DQo

="

}

]

}

]

}

}

### JSON RESPONSE

[

{

"OperationSucceeded": true,

"Message": "Attachment was successfully added to the Documents field.", "ComponentId": 10001,

"RecordId": 2, "AttachmentInfo": {

"FileName": "import\_temp.csv", "FieldId": 34,

"DocumentId": 20

}

}

]

### Master/Detail Examples

#### The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @UpdateRecordAttachmentsMDInput.xml https://keylight.lockpath.com:4443/ComponentService/UpdateRecordAttachments

### XML REQUEST (UpdateRecordAttachmentsMDInput.xml)

<UpdateRecordAttachments>

<componentId>10199</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<Id>36</Id>

<FieldValues>

<KeyValuePair>

<key>5045</key>

<value i:type='RecordAttachmentList'>

<Attachment>

<FileName>helloworld.txt</FileName>

<FileData>SGVsbG8gV29ybGQ=</FileData>

</Attachment>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</UpdateRecordAttachments>

### XML RESPONSE

<AttachmentOperationResultList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<AttachmentOperationResult>

<OperationSucceeded>true</OperationSucceeded>

<Message>Attachment was successfully added to the Documents field.</Message>

<ComponentId>10199</ComponentId>

<RecordId>36</RecordId>

<AttachmentInfo>

<FileName>helloworld.txt</FileName>

<FieldId>5045</FieldId>

<DocumentId>45</DocumentId>

</AttachmentInfo>

</AttachmentOperationResult>

</AttachmentOperationResultList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @UpdateRecordAttachmentsMDInput.json https://keylight.lockpath.com:4443/ComponentService/UpdateRecordAttachments

### JSON REQUEST (UpdateRecordAttachmentsMDInput.json)

{

"componentId": "10199", "dynamicRecord": {

"Id": "37",

"FieldValues": [

{

"key": "5045",

"value":

[

{

"FileName": "helloworld.txt", "FileData": "SGVsbG8gV29ybGQ="

}

]

}

]

}

}

### JSON RESPONSE

[

{

"OperationSucceeded": true,

"Message": "Attachment was successfully added to the Documents field.", "ComponentId": 10199,

"RecordId": 37, "AttachmentInfo": {

"FileName": "helloworld.txt", "FieldId": 5045,

"DocumentId": 46

}

}

]

## ImportFile

#### Queue a job to import a file for a defined import template.

URL: http://[instance name]:[port]/ComponentService/ImportFile Method: POST

Input: tableAlias (string): The Alias for the table to import into importTemplateName (string): The Name of the import template fileName (string): The Name of the import file

fileData (string): Base64 encoded string of file contents

runAsSystem (boolean): Run import as the Keylight System account rather than authentication account

Permissions: The authentication account must have Read, Create, Update, and Import/Bulk General Access permissions to the defined table. To enable the Run As System option, the authentication account must have also have Read, Create, and Update Administrative Access permissions to the defined table.

**Examples**

ImportFile queues a job to import a file for a defined import template. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @ImportFile.xml

"http://keylight.lockpath.com:4443/ComponentService/ImportFile"

**XML REQUEST (ImportFile.xml)**

<ImportFile>

<tableAlias>\_dubs</tableAlias>

<importTemplateName>CSV Import</importTemplateName>

<fileName>import\_temp.csv</fileName>

<fileData>VGV4dCxudW0scmljaHksaXB2NCxpcHY2LGRhdGUsZGF0ZXRpbWUNCnIxLDUxMCw8Yj5ib2xkIHRoaW5nPC 9iPiwxOTIuMTY4LjIuNCwwMDE6MGRiODowMDAwOjAwNDI6MDAwMDo4YTJlOjAzNzA6NzMzNCwyLzE4LzE5ODYsMi8xOS 8xOTg2DQo=</fileData>

<runAsSystem>false</runAsSystem>

</ImportFile>

### XML RESPONSE

true

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @ImportFile.json http://keylight.lockpath.com:4443/ComponentService/ImportFile

### JSON REQUEST (ImportFile.json)

{

"tableAlias": "\_dubs", "importTemplateName": "CSV Import", "fileName": "import\_temp.csv", "fileData":

"VGV4dCxudW0scmljaHksaXB2NCxpcHY2LGRhdGUsZGF0ZXRpbWUNCnIxLDUxMCw8Yj5ib2xkIHRoaW5nPC9i PiwxOTIuMTY4LjIuNCwwMDE6MGRiODowMDAwOjAwNDI6MDAwMDo4YTJlOjAzNzA6NzMzNCwyLzE4LzE5ODYsM i8xOS8xOTg2DQo=",

"runAsSystem": "false"

}

#### ]

**JSON RESPONSE**

true

## Issue Assessments

#### Assessments can be initiated via the API into fields on DCF tables and on Master Detail records. Assessments require specific data to be issued via a Request XML file. Note that only XML request examples are available.

**Issue Assessments**

**Examples: Issue Assessments REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d

@Request.xml https://keylight.lockpath.com:4443/AssessmentService/IssueAssessment > RESULTS.xml

**0\_base REQUEST (XML)**

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<ProjectId/>

<TableId/>

<FieldId/>

<ContentId/>

<VendorId/>

<IsVendorInternalMode/>

<Name/>

<TemplateId/>

<VendorContactId/>

<UserIds>x, y</UserIds>

<GroupIds>x, y</GroupIds>

<AllowDelegation/>

<AssignedUserOnly/>

<ReviewerId/>

<ShowUserScores/>

<IssuanceScheduleType/>

<IssueDate/>

<BeginningDate/>

<EndingDate/>

<RepeatUnit/>

<RepeatInterval/>

<RepeatsSunday/>

<RepeatsMonday/>

<RepeatsTuesday/>

<RepeatsWednesday/>

<RepeatsThursday/>

<RepeatsFriday/>

<RepeatsSaturday/>

<DueDate/>

<DueUnit/>

<DueInterval/>

<PrepopulatePriorAnswers/>

<EmailSubject/>

<EmailBody/>

<SendReviewerOrIssuerEmail/>

<SendCategoryEmail/>

<AdministrativeEmailSubject/>

<AdministrativeEmailBody/>

</assessmentIssuance>

</IssueAssessment>

### Immediate REQUEST (XML)

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<ProjectId>2</ProjectId>

<Name>Immediate Scheduled Assessment</Name>

<TemplateId>13</TemplateId>

<UserIds>6, 28</UserIds>

<GroupIds>7</GroupIds>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>28</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>immediate</IssuanceScheduleType>

<DueDate>12/31/2018</DueDate>

<PrepopulatePriorAnswers>false</PrepopulatePriorAnswers>

<EmailSubject>Immediate Scheduled Email Subject</EmailSubject>

<EmailBody>Immediate Scheduled Email Body</EmailBody>

</assessmentIssuance>

</IssueAssessment>

### Onetime REQUEST (XML)

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<ProjectId>2</ProjectId>

<Name>One-Time Scheduled Assessment</Name>

<TemplateId>13</TemplateId>

<UserIds>6, 28</UserIds>

<GroupIds>7</GroupIds>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>28</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>onetime</IssuanceScheduleType>

<IssueDate>07/01/2018</IssueDate>

<DueDate>12/31/2018</DueDate>

<PrepopulatePriorAnswers>true</PrepopulatePriorAnswers>

<EmailSubject>One-Time Scheduled Email Subject</EmailSubject>

<EmailBody>One-Time Scheduled Email Body</EmailBody>

</assessmentIssuance>

</IssueAssessment>

### Recurring REQUEST (XML)

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<TableId>10001</TableId>

<FieldId>35</FieldId>

<ContentId>64</ContentId>

<Name>Recurring Scheduled Assessment API</Name>

<TemplateId>53</TemplateId>

<UserIds>27</UserIds>

<GroupIds>17</GroupIds>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>10</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>recurring</IssuanceScheduleType>

<BeginningDate>03/10/2018</BeginningDate>

<RepeatUnit>monthly</RepeatUnit>

<EndingDate>11/05/2018</EndingDate>

<RepeatInterval>1</RepeatInterval>

<DueUnit>daily</DueUnit>

<DueInterval>1</DueInterval>

<EmailSubject>Recurring Scheduled Email Subject API</EmailSubject>

<EmailBody>Recurring Scheduled Email Subject API</EmailBody>

<SendReviewerOrIssuerEmail>true</SendReviewerOrIssuerEmail>

<AdministrativeEmailSubject>Recurring Reviewer Email Subject API</AdministrativeEmailSubject>

<AdministrativeEmailBody>Recurring Reviewer Email Body API</AdministrativeEmailBody>

</assessmentIssuance>

</IssueAssessment>

### Vendor REQUEST (XML)

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<TableId>10066</TableId>

<FieldId>1439</FieldId>

<ContentId>15</ContentId>

<VendorId>15</VendorId>

<Name>vendor assessment API [VendorName]</Name>

<TemplateId>53</TemplateId>

<VendorContactId>44</VendorContactId>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>10</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>recurring</IssuanceScheduleType>

<BeginningDate>03/08/2018</BeginningDate>

<RepeatUnit>Monthly</RepeatUnit>

<EndingDate>11/05/2018</EndingDate>

<RepeatInterval>1</RepeatInterval>

<DueUnit>Weeks</DueUnit>

<DueInterval>1</DueInterval>

<EmailSubject>Recurring Scheduled Email Subject API Vendor</EmailSubject>

<EmailBody>Recurring Scheduled Email Subject API [VendorContact]</EmailBody>

<SendReviewerOrIssuerEmail>true</SendReviewerOrIssuerEmail>

<AdministrativeEmailSubject>Recurring Reviewer Email Subject API Vendor</AdministrativeEmailSubject>

<AdministrativeEmailBody>Recurring Reviewer Email Body API Vendor</AdministrativeEmailBody>

</assessmentIssuance>

</IssueAssessment>

### Vendor Immediate Assessment email format REQUEST (XML)

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<TableId>10066</TableId>

<FieldId>1439</FieldId>

<ContentId>15</ContentId>

<VendorId>15</VendorId>

<Name>Immediate Vendor Assessment API 2 [VendorName]</Name>

<TemplateId>53</TemplateId>

<VendorContactId>72</VendorContactId>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>10</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>immediate</IssuanceScheduleType>

<DueDate>03/30/2018</DueDate>

<PrepopulatePriorAnswers>false</PrepopulatePriorAnswers>

<EmailSubject>Immediate Vm Scheduled Email Subject</EmailSubject>

<EmailBody>Immedate Vm Scheduled Email Take this assessment immediately [VendorContact].

Table Id = Vendor Profiles Field Id = Assessments Content Id = record id

Vendor Id = New Vendor Profile-May Be Deleted Quickly

Template Id = Generate Findings with Attachments [AssessmentUrl] [AssessmentName]

Vendor Contact Id = Bob Jones Reviewer Id = Betty Barnes [VendorContact],&lt;br /&gt; &lt;br /&gt;

An assessment has been issued to [VendorName] with you as the specified contact. To begin working on the assessment, log into the Keylight Vm portal and enter the credentials issued in a previous email. Submit the assessment for review once you have finished answering all questions.

&lt;ul&gt;

&lt;li&gt;Keylight Vm Portal URL: [SiteUrl]&lt;/li&gt; &lt;li&gt;Assessment Name: [AssessmentUrl]&lt;/li&gt; &lt;li&gt;Assessment Due Date: [DueDate]&lt;/li&gt; &lt;/ul&gt;

Thank you,&lt;br /&gt; &lt;br /&gt;

Keylight Vm team</EmailBody>

<SendReviewerOrIssuerEmail>true</SendReviewerOrIssuerEmail>

<AdministrativeEmailSubject>Reviewer Email for Immediate Vm Scheduled Email Subject</AdministrativeEmailSubject>

<AdministrativeEmailBody>Reviewer Email for Immediate Vm Scheduled Email Subject Table Id = Vendor Profiles

Field Id = Assessments Content Id = record id

Vendor Id = New Vendor Profile-May Be Deleted Quickly

Template Id = Generate Findings with Attachments [AssessmentUrl] [AssessmentName]

Vendor Contact Id = Bob Jones

Reviewer Id = Betty Barnes</AdministrativeEmailBody>

</assessmentIssuance>

</IssueAssessment>

### RESULTS (XML)

<AssessmentIssuanceItem xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<ProjectId>1</ProjectId>

<TableId i:nil="true"/>

<FieldId i:nil="true"/>

<ContentId i:nil="true"/>

<VendorId i:nil="true"/>

<IsVendorInternalMode i:nil="true"/>

<Name>Immediate Scheduled Assessment API CM</Name>

<TemplateId>11</TemplateId>

<UserIds>27</UserIds>

<VendorContactIdi:nil="true"/>

<GroupIds>17</GroupIds>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>10</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>immediate</IssuanceScheduleType>

<BeginningDate i:nil="true"/><IssueDate i:nil="true"/>

<DueDate>03/15/2018</DueDate>

<EndingDate i:nil="true"/>

<RepeatUnit i:nil="true"/>

<RepeatInterval i:nil="true"/>

<RepeatsSunday i:nil="true"/>

<RepeatsMonday i:nil="true"/>

<RepeatsTuesday i:nil="true"/>

<RepeatsWednesday i:nil="true"/>

<RepeatsThursday i:nil="true"/>

<RepeatsFriday i:nil="true"/>

<RepeatsSaturday i:nil="true"/>

<DueUnit i:nil="true"/>

<DueInterval i:nil="true"/>

<PrepopulatePriorAnswers>false</PrepopulatePriorAnswers>

<EmailSubject>Immediate Scheduled Email Subject</EmailSubject>

<EmailBody>Immediate Scheduled Email Take this assessment immediately.</EmailBody>

<SendReviewerOrIssuerEmail i:nil="true"/>

<SendCategoryEmail i:nil="true"/>

<AdministrativeEmailSubject i:nil="true"/>

<AdministrativeEmailBody i:nil="true"/>

</AssessmentIssuanceItem>

### Special Case: Assessment Issued from Master Detail level record

#### Table Id: Id for the Master Detail table in the master record Field Id: Id for the Assessment field in the MD record Content Id: Id for the MD record

**Vendor Immediate Assessment on Master Detail record REQUEST (XML)**

<IssueAssessment>

<assessmentIssuance xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<TableId>10145</TableId>

<FieldId>2816</FieldId>

<ContentId>4</ContentId>

<VendorId>1</VendorId>

<Name>vendor assessment API [VendorName] test it to MD field</Name>

<TemplateId>53</TemplateId>

<VendorContactId>61</VendorContactId>

<AllowDelegation>false</AllowDelegation>

<AssignedUserOnly>true</AssignedUserOnly>

<ReviewerId>10</ReviewerId>

<ShowUserScores>true</ShowUserScores>

<IssuanceScheduleType>immediate</IssuanceScheduleType>

<DueDate>04/30/2018</DueDate>

<PrepopulatePriorAnswers>false</PrepopulatePriorAnswers>

<EmailSubject>Immediate Vendor Assessment API MD VM Email Subject</EmailSubject>

<EmailBody>Immediate Vendor Assessment API MD VM Take this assessment immediately [VendorContact].</EmailBody>

</assessmentIssuance>

</IssueAssessment>

## DeleteRecord

#### Delete a selected record from within a chosen component.

**IMPORTANT:** DeleteRecord will update the record, making it so it will no longer be viewable within Keylight Platform. Records are soft-deleted to maintain any historical references to the record and can be restored with a database script.

URL: http://[instance name]:[port]/ComponentService/DeleteRecord Method: DELETE

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component Permissions: The authentication account must have:

* + Read/Delete General Access permission to the selected component
  + Read permissions to the selected record

**Examples**

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X DELETE –d @DeleteRecord.xml

http://keylight.lockpath.com:4443/ComponentService/DeleteRecord

**XML REQUEST (DeleteRecord.xml)**

<DeleteRecord>

<componentId>10001</componentId>

<recordId>1</recordId>

</DeleteRecord>

### XML RESPONSE

true

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X DELETE-d @DeleteRecord.json

http://keylight.lockpath.com:4443/ComponentService/DeleteRecord

### JSON REQUEST (DeleteRecord.json)

{

"componentId": "10001",

"recordId": "1"

}

### JSON RESPONSE

true

## DeleteRecordAttachments

#### Deletes the specified attachments from the provided document fields on a specific record.

URL: "http://[instance name]:[port]/ComponentService/DeleteRecordAttachments" Method: POST

Input: componentID (Integer): The ID of the desired component

recordId (Integer): The ID for the individual record within the component fieldId (Integer): The ID for the individual field within the component attachmentId (Integer): The ID for the individual attachment within the component

Permissions: The authentication account must have Read and Delete General Access permissions to:

* + Selected component
  + Selected record
  + Applicable fields in the component (table)

**Examples**

The DeleteRecordAttachments deletes one or more attachments from the provided document fields on a record. The cURL -b option is used to provide authentication.

**XML REQUEST (cURL)**

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @DeleteRecordAttachments.xml

"http://keylight.lockpath.com:4443/ComponentService/DeleteRecordAttachments"

**XML REQUEST (DeleteRecordAttachments.xml)**

<DeleteRecordAttachments>

<componentId>12345</componentId>

<dynamicRecord xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:a="<http://www.w3.org/2001/XMLSchema>">

<Id>1</Id>

<FieldValues>

<KeyValuePair>

<key>1234</key>

<value i:type='DynamicRecordList'>

<Record>

<Id>1</Id>

</Record>

<Record>

<Id>2</Id>

</Record>

</value>

</KeyValuePair>

</FieldValues>

</dynamicRecord>

</DeleteRecordAttachments>

### XML RESPONSE

<AttachmentOperationResultList xmlns:i="<http://www.w3.org/2001/XMLSchema-instance>">

<AttachmentOperationResult>

<OperationSucceeded>true</OperationSucceeded>

<Message>Attachment was successfully deleted from the Documents field.</Message>

<ComponentId>12345</ComponentId>

<RecordId>1</RecordId>

<AttachmentInfo>

<FileName>Attachment1.txt</FileName>

<FieldId>1234</FieldId>

<DocumentId>1</DocumentId>

</AttachmentInfo>

</AttachmentOperationResult>

<AttachmentOperationResult>

<OperationSucceeded>true</OperationSucceeded>

<Message>Attachment was successfully deleted from the Documents field.</Message>

<ComponentId>12345</ComponentId>

<RecordId>1</RecordId>

<AttachmentInfo>

<FileName>Attachment2.xml</FileName>

<FieldId>1234</FieldId>

<DocumentId>2</DocumentId>

</AttachmentInfo>

</AttachmentOperationResult>

</AttachmentOperationResultList>

### JSON REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/json" -H "Accept: application/json" -X POST

-d @DeleteRecordAttachments.json http://keylight.lockpath.com:4443/ComponentService/DeleteRecordAttachments

### JSON REQUEST (DeleteRecordAttachments.json)

{

"componentId": "10001", "dynamicRecord": {

"Id": "2",

"FieldValues": [

{

"key": "34",

"value": [

{

"Id": "20"

}

]

}

]

}

}

### JSON RESPONSE

[

{

"OperationSucceeded": true,

"Message": "Attachment was successfully deleted from the Documents field.", "ComponentId": 10001,

"RecordId": 2, "AttachmentInfo": {

"FileName": "import\_temp.csv", "FieldId": 34,

"DocumentId": 20

}

}

]

Appendices

# A: Field Types

#### This appendix provides the unique identifiers for the field type in a Keylight table.

**Unique Identifiers for Field Types**

The Keylight Platform uses several field types each represented by a unique ID. This unique ID is used to describe the data in GetComponent and GetField.

|  |  |
| --- | --- |
| **Field Types** | |
| **ID** | **Type** |
| 1 | Text |
| 2 | Numeric |
| 3 | Date |
| 4 | IP Address |
| 5 | Lookup |
| 6 | Master/Detail |
| 7 | Matrix |
| 8 | Documents |
| 9 | Assessments |
| 10 | Yes/No |

B: cURL Command Switches

This appendix includes the basis cURL command switches.

**Basic cURL Command Switches**

Some basic cURL command switches that may be helpful:

|  |  |  |
| --- | --- | --- |
| **Switches** | | |
| **Switch** | **Option** | **Use** |
| -b | cookie | Include cookie |
| -c | cookie-jar | Store cookie |
| -H | header | Include xml header information, for example, Content-Type |
| -X |  | Use to switch to POST or DELETE |
| -d | data | XML body request, use @filename.xml to specify an input file |

C: Filtering

This appendix provides the filters for search criteria.

**Search Filters**

Search filters may be used with the GetRecords or GetRecordCount methods. A variety of filters are available and an unlimited number of search criteria may be applied to each transaction that supports filtering. A filter is composed of a path, type, and value. The format of a filter is shown below.

<filters>

<SearchCriteriaItem>

<Field Path>

<int>path</int>

</Field Path>

<FilterType>ID</FilterType>

<Value>value</Value>

</SearchCriteriaItem>

</filters>

**Field Path**

#### Field Path is the Keylight component column that will be sorted on. It describes the column ID where the data is stored. Because the Keylight Platform supports multiple lookup field reference data types, multiple points may be required to describe the path to the columnId. The GetFields() action described in the API can provide the Field ID number for any DCF component.

In this example, the field path for Model is 8676. The filter input would be as follows:

<Field Path>

<int>8676</int>

</Field Path>

|  |  |  |
| --- | --- | --- |
| **Get Fields (Equipment)** | | |
| **Id** | **Name** | **Field Type** |
| 8684 | Serial | Text |
| 8670 | Make | Text |
| 8676 | Model | Text |
| 8683 | Building | Lookup |

|  |  |  |
| --- | --- | --- |
| **Get Fields (Facility)** | | |
| **Id** | **Name** | **Field Type** |
| 9658 | Building | Text |
| 9661 | Address | Text |
| 9663 | State | Lookup |
| 9671 | Zip | Numeric |

|  |  |  |  |
| --- | --- | --- | --- |
| **Get Records** | | | |
| **Serial** | **Make** | **Model** | **Room** |
| 67847 | Dell | M4600 | C27 |
| A1234567 | HP | H45000 | C27 |
| 67849 | Dell | M6600 | A123 |
| B78888998889 | Lenovo | T420 | C28 |

#### In the Equipment table, the Room is a lookup value to the rooms field of the Facility table. The value for the field is referenced in the Equipment table but actually stored in the Facility table. The field path search filter parameter must describe the relationship. If a search filter is filtering equipment based on the building in which it is located, the path will be:

<Field Path>

<int>9658</int>

<int>8683</int>

</Field Path>

### Filter Types

#### The Filter Type describes the integer ID for the filter being applied. The table below describes the available filter types. The xml syntax for "contains" is:

<FilterType>1</FilterType>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Filter Types** | | | | | |
| **ID** | **Filter** | **ID** | **Filter** | **ID** | **Filter** |
| 1 | Contains | 8 | < | 15 | Is Null |
| 2 | Excludes | 9 | >= | 16 | Is Not Null |
| 3 | Starts With | 10 | <= | 10001 | Offset |
| 4 | Ends With | 11 | Between | 10002 | Contains Any |
| 5 | = | 12 | Not Between | 10003 | Contains Only |
| 6 | <> | 13 | Is Empty | 10004 | Contains None |
| 7 | > | 14 | Is Not Empty | 10005 | Contains At Least |

### Value

#### Value describes the comparison that the column will be measured against. For example, to filter all Dell computers the value would be "Dell". And appear as:

<Value>Dell</Value>

#### For the filter options 11 and 12, two values are required that are delimited by a pipe (|).

<Value>100|200</Value>

#### For 13-16 no values will be required.

Contains filters apply only to One-to-Many Lookup fields.

**SearchCriteriaItem**

By combining Field Path, Filter, and Value, a complete filter (denoted in XML as SearchCriteriaItem) is created. Multiple filters can be stacked to create the desired search parameters.

<filters>

<SearchCriteriaItem>

<Field Path>

<int>path</int>

</Field Path>

<FilterType>ID</FilterType>

<Value>value</Value>

</SearchCriteriaItem>

</filters>

### Examples REQUEST (cURL)

curl -b cookie.txt -H "content-type: application/xml;charset=utf-8" -X POST -d @GetRecordsInput.xml http://keylight.lockpath.com:4443/ComponentService/GetRecords

### REQUEST (XML)

**Sample Filter on a Lookup Field:**

<GetRecords>

<componentId>10001</componentId>

<pageIndex>0</pageIndex>

<pageSize>100</pageSize>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>307</int>

<int>23</int>

</FieldPath>

<FilterType>5</FilterType>

<Value>4</Value>

</SearchCriteriaItem>

</filters>

</GetRecords>

#### The above request will get 100 records with the Device Type Id of 4 (FIREWALL) from the 10001 (Devices) component. It is showing a filter on the Device Type field. <int>307</int> is the Field id for Device Type field on the Devices Table. <int>23</int> is the Field id for the Id field from the LU table, Device Types.

**Sample Filter on a Workflow Stage:**

<GetRecords>

<componentId>10001</componentId>

<pageIndex>0</pageIndex>

<pageSize>100</pageSize>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>351</int>

<int>232</int>

</FieldPath>

<FilterType>1</FilterType>

<Value>Publish</Value>

</SearchCriteriaItem>

</filters>

</GetRecords>

#### The above request will pull 100 records from the 10001 (Devices) table that contain (FilterType 1) the value “Publish”. The WorkflowStage component does not appear on the API components list. The Workflow Stage Component Fields are 231 for the Id field and 232 for the Name field. See table for additional system field identifiers for Components not provided via the API Component request. Note that these system components do not offer record create, edit or delete access.

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | **Component Id** | **Field Name** | **Field Id** |
| Users | 100 | Id | 321 |
|  |  | First Name | 322 |
|  |  | Middle Name | 323 |
|  |  | Last Name | 324 |
|  |  | Full Name | 1335 |
|  |  | Vendor | 281 |
|  |  | Title | 265 |
|  |  | email | 266 |
| Groups | 101 | Id | 325 |
|  |  | Name | 326 |
| Workflow Stage | 57 | Id | 231 |
|  |  | Name | 232 |
| Workflow | 69 | Id | 570 |
|  |  | Name | 571 |

**Sample Filter on a Yes/No Field:**

<GetRecords>

<componentId>10001</componentId>

<pageIndex>0</pageIndex>

<pageSize>100</pageSize>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>5130</int>

</FieldPath>

<FilterType>5</FilterType>

<Value>True</Value>

</SearchCriteriaItem>

</filters>

</GetRecords>

### Sample multiple filter requests:

<GetRecords>

<componentId>10001</componentId>

<pageIndex>0</pageIndex>

<pageSize>100</pageSize>

<filters>

<SearchCriteriaItem>

<FieldPath>

<int>305</int>

<int>321</int>

</FieldPath>

<FilterType>5</FilterType>

<Value>72</Value>

</SearchCriteriaItem>

<SearchCriteriaItem>

<FieldPath>

<int>11</int>

</FieldPath>

<FilterType>15</FilterType>

</SearchCriteriaItem>

</filters>

</GetRecords>

# D: Language Identifiers

#### This appendix includes language IDs that can be utilized in the GetUser, CreateUser, and UpdateUser methods.

**Language IDs**

The Language object of the GetUser, CreateUser, and UpdateUser methods determines the language in the Keylight Platform. In addition to English, which is the default language, the Keylight Platform provides other languages that can be utilized in the language object.

This table shows a list of available language names and language IDs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Language Name** | **Language ID** | **Language Name** | **Language ID** | **Language Name** | **Language ID** |
| Afrikaans | 54 | Hindi | 57 | Polish | 21 |
| Albanian | 28 | Hungarian | 14 | Portuguese | 22 |
| Alsatian | 132 | Icelandic | 15 | Punjabi | 70 |
| Amharic | 94 | Igbo | 112 | Quechua | 107 |
| Arabic | 1 | Indonesian | 33 | Romanian | 24 |
| Armenian | 43 | Inuktitut | 93 | Romansh | 23 |
| Assamese | 77 | Irish | 60 | Russian | 25 |
| Azerbaijani | 44 | isiXhosa | 52 | Sakha | 133 |
| Bangla | 69 | isiZulu | 53 | Sami (Northern) | 59 |
| Bashkir | 109 | Italian | 16 | Sanskrit | 79 |
| Basque | 45 | Japanese | 17 | Scottish Gaelic | 145 |
| Belarusian | 35 | Kannada | 75 | Serbian | 31770 |
| Bosnian | 30746 | Kazakh | 63 | Sesotho sa Leboa | 108 |
| Breton | 126 | Khmer | 83 | Setswana | 50 |
| Bulgarian | 2 | K'iche | 134 | Sinhala | 91 |
| Catalan | 3 | Kinyarwanda | 135 | Slovak | 27 |
| Chinese | 30724 | Kiswahili | 65 | Slovenian | 36 |
| Corsican | 131 | Konkani | 87 | Spanish | 10 |
| Croatian | 26 | Korean | 18 | Swedish | 29 |
| Czech | 5 | Kyrgyz | 64 | Syriac | 90 |
| Danish | 6 | Lao | 84 | Tajik | 40 |
| Dari | 140 | Latvian | 38 | Tamazight | 95 |
| Divehi | 101 | Lithuanian | 39 | Tamil | 73 |
| Dutch | 19 | Luxembourgish | 110 | Tatar | 68 |
| English | 9 | Macedonian (FYROM) | 47 | Telugu | 74 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Language Name** | **Language ID** | **Language Name** | **Language ID** | **Language Name** | **Language ID** |
| Estonian | 37 | Malay | 62 | Thai | 30 |
| Faroese | 56 | Malayalam | 76 | Tibetan | 81 |
| Filipino | 100 | Maltese | 58 | Turkish | 31 |
| Finnish | 11 | Maori | 129 | Turkmen | 66 |
| French | 12 | Mapudungun | 122 | Ukrainian | 34 |
| Frisian | 98 | Marathi | 78 | Upper Sorbian | 46 |
| Galician | 86 | Mohawk | 124 | Urdu | 32 |
| Georgian | 55 | Mongolian | 80 | Uyghur | 128 |
| German | 7 | Nepali | 97 | Uzbek | 67 |
| Greek | 8 | Norwegian | 20 | Vietnamese | 42 |
| Greenlandic | 111 | Occitan | 130 | Welsh | 82 |
| Gujarati | 71 | Odia | 72 | Wolof | 136 |
| Hausa | 104 | Pashto | 99 | Yi | 120 |
| Hebrew | 13 | Persian | 41 | Yoruba | 106 |

E: Troubleshooting Tips

This appendix provides troubleshooting tips for API calls and FAQs.

**API Troubleshooting**

The following are problems you may encounter using the Keylight API and a suggested solution for each.

|  |  |
| --- | --- |
| **Problem** | **What to do** |
| Enable API | 1. Confirm the Keylight API is licensed. Navigate to Setup > Keylight Platform > System > Subscription / License Details. Keylight Data API should be listed within Connectors. 2. Confirm the given user has API access. Navigate to Setup > Keylight Platform > Security > Users. Select the user in question and confirm the API Access field is set to Yes.   Note Keylight REST API is available only with Keylight Enterprise Edition or with an additional license in the Keylight Standard Edition. |
| User access permissions | Confirm the user has appropriate rights to the component (table), workflow stage for the given record, the record itself (restrict record access) and the specified fields. |
| Server encountered error processing request. The incoming message has an unexpected message format 'Raw'.  Expected formats are xml or json | Verify the message format is correct. |
| Using the right method (GET vs POST vs DELETE) | Consult this guide for the method appropriate to your request. |
| Capitalization of fields (for example, Login) | API methods are case sensitive. Verify the case for the given method in this guide. |
| Having set-cookie not cookie | Verify the correct cookie file is specified from the login. |
| A valid session is required for API request | If you previously had a valid cookie, the session has now expired. Consider creating a custom security configuration with an extended timeout and assign the new configuration to your API user(s) if the session is expiring inappropriately. |
| Input string is not in the correct format | Review the syntax of your command. |

|  |  |
| --- | --- |
| **Problem** | **What to do** |
| cURL command results in error | API methods are case sensitive. Verify that you are entering the proper syntax and case for the cURL command as documented in this guide. |
| Order of switches for cURL command | Type *curl –help* and review the commands in the online help or *curl – man>Curl\_Manual.txt* and review the text file for the complete curl manual.  See http[s]://<machine\_name>:<port>/SecurityService/help for the published list of call templates. |
| Certificate problem | Use the -k or --insecure option of the curl command when an SSL certificate warning occurs.  Type *curl –help* and review the commands in the online help or *curl – man>Curl\_Manual.txt* and review the text file for the complete curl manual. |

**Keylight API FAQ**

The following are frequently asked questions about the Keylight API.

**In what Keylight editions are the REST API available?**

Keylight Enterprise Edition and with an additional license in the Keylight Standard Edition. The REST API is not available with the Keylight Team Edition.

**Why should I choose to use the Keylight API?**

The Keylight API is extremely useful for building automated linkages to your internal systems for performing bi- directional updates. You can also use the REST API methods to update/import one-to-many lookup fields for fully managing the record.

**Can I modify user information using REST API?**

Yes, you can add, update, and assign security roles to users and groups using API calls.

**Does an API account take up a license?**

Users with Portal access may have access to the Keylight API without using an additional user license. If a user account is established specifically for API access, there must be a full user license available for granting the appropriate access.

**What methods are supported?**

The REST API supports the following HTTP verbs: GET, POST, and DELETE.

**Is there a limit to the number of calls I can make?**

The API is limited to 10 active sessions at one time and up to 20 requests can be processed per second. When retrieving multiple (bulk) records, up to 1,000 records can be requested at once. Messaging is provided if a request is unable to be completed to allow for subsequent attempts.

**Why can my user not make API calls?**

The API access option may not be enabled in the user profile. Verify that the user account for the user you want to use in an API call has API access.

**Will Lockpath program against the API on our behalf?**

You can engage Lockpath Professional Services for a statement of work, who will gather requirements and establish the expected deliverables. The Professional Services consultant will estimate, schedule, and complete the project to meet your needs.