The Lockpath API implements RESTful interface

HTTP verbs: GET, POST, DELETE

Responses: JSON, XML

# Access Configuration

All API requests are made using the base URL.

http://[instancename]:[port]/

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

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>

**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

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 Lockpath Platform. 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.

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.

### Data Structure

A key component of the Lockpath 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 Lockpath 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 Lockpath 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.

**Login**

Accepts an account username and password, verifies them within Lockpath 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 Lockpath application account Password (Text) Password for the Lockpath 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 Lockpath API. Users must also have appropriate permissions for any data they wish to access or manipulate.

**RESPONSE Success:**

**True**

Response Failed:

???

**Ping**

Refreshes a valid Lockpath 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 Lockpath API.

**RESPONSE Success:**

**True**

Response Failed:

???

# Logout

Terminates a Lockpath 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 Lockpath API.

**RESPONSE Success:**

**True**

Response Failed:

???

# 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 Lockpath 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 Lockpath Platform, the Preferred Locale field value in My Profile preferences is set for the user. In the Lockpath 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 Lockpath Setup > Multilingual > Languages area of the Lockpath Platform to reveal the language ID. For a list of languages and corresponding language IDs available in the Lockpath Platform, see Language IDs in the appendix.

Response Success

Id: <int>

FullName: <string>

Username: <string>

IsActive: <boolean>

IsLocked: <boolean>

IsDeleted: <boolean>

AccountType": 1

FirstName: <string>

MiddleName: <string>

LastName: <string>

Title: <string>

Language: <int>

EmailAddress: <string>

HomePhone: <string>

WorkPhone: <string>

MobilePhone: <string>

Fax: <string>

IsSAML: <boolean>

IsLDAP: <boolean>

SecurityConfiguration: <Arrary> [Id: <int>, DisplayName: <string>]

APIAccess: <boolean>

Groups: <Arrary> [Id: <int>, Name: <string>] ??? check array keys

SecurityRoles: <Arrary> [Id: <int>, Name: <string>]

FunctionalRoles: <Arrary> [Id: <int>, Name: <string>] ??? check array keys

## GetUsers

Returns a list of users and supporting fields. The list does not include Deleted users and can include non-Lockpath 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-Lockpath 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>

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"

}

]

}

RESPONSE SUCCESS

[

{

"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 Lockpath users. The count does not include Deleted users and can include non-Lockpath 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-Lockpath 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 |

**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>

RESPONSE SUCCESS

#

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 Lockpath 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 Lockpath Platform, the Preferred Locale field value in My Profile preferences is set for the user. In the Lockpath 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 Lockpath Setup > Multilingual > Languages area of the Lockpath Platform to reveal the language ID. For a list of languages and corresponding language IDs available in the Lockpath 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) |

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": ""

}

RESPONSE SUCCESS

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 Lockpath 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 Lockpath Platform, the Preferred Locale field value in My Profile preferences is set for the user. In the Lockpath 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 Lockpath Setup > Multilingual > Languages area of the Lockpath Platform to reveal the language ID. For a list of languages and corresponding language IDs available in the Lockpath 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) |

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"

}

]

}

]

**RESPONSE SUCCESS**

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.

RESPONSE SUCCESS

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.

RESPONSE SUCCESS

{

"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.

JSON REQUEST (GetGroups.json)

{

"pageIndex": "0",

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

[

{

"Field":

{

"ShortName": "BusinessUnit"

},

"FilterType": "5",

"Value": "False"

}

]

}

RESPONSE SUCCESS

[

{

"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.

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"

}

]

}

RESPONSE SUCCESS

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.

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"

}

]

}

]

**RESPONSE SUCCESS**

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.

RESPONSE SUCCESS

true

3: Component Services API

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

RESPONSE SUCCESS

{

"Id": 10050,

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

}

**GetComponentList**

Returns a complete list of all Lockpath 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.

RESPONSE SUCCESS

[

{

"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.

RESPONSE SUCCESS

{

"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.

RESPONSE SUCCESS

{

"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.

RESPONSE SUCCESS

[

{

"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.

JSON REQUEST (GetAvailableLookupRecords.json)

{

"fieldId": "12345",

"pageIndex": "0",

"pageSize": "1000",

"recordId": "123"

}

RESPONSE SUCCESS

[

{

"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)

RESPONSE SUCCESS

[

{

"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)

RESPONSE SUCCESS

{

"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

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.

JSON REQUEST (GetRecordCount.json)

{

"componentId": "10001",

"pageIndex": "0",

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

{

"FieldPath": [ 11

],

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

}

]

}

RESPONSE SUCCESS

[

{

"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.

JSON REQUEST (GetRecordCount.json)

{

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

{

"FieldPath": [ 11

],

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

}

]

}

RESPONSE SUCCESS

#

# 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)

RESPONSE SUCCESS

{

"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)

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]

}

RESPONSE SUCCESS

[

{

"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)

**Response Success**

{

"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)

RESPONSE SUCCESS

[

{

"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.

RESPONSE SUCCESS

{

"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.

RESPONSE SUCCESS

[

{

"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.

JSON REQUEST (TransitionRecord.json)

{

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

"transitionId": "42"

}

RESPONSE SUCCESS

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.

JSON REQUEST (VoteRecord.json)

{

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

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

}

RESPONSE SUCCESS

true

# CreateRecord

Create a new record within the specified component of the Lockpath 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 Lockpath 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: Lockpath 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 |

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"

}

]

}

]

}

}

RESPONSE SUCCESS

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 Lockpath 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

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"

}

]

}

]

}

}

RESPONSE SUCCESS

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 (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)

JSON REQUEST (UpdateRecordAttachments.json)

{

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

"Id": "2",

"FieldValues": [

{

"key": "34",

"value": [

{

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

"VGV4dCxudW0scmljaHksaXB2NCxpcHY2LGRhdGUsZGF0ZXRpbWUNCnIxLDUxMCw8Yj5ib2xkIHRoaW5nPC9iPiwxOTI uMTY4LjIuNCwwMDE6MGRiODowMDAwOjAwNDI6MDAwMDo4YTJlOjAzNzA6NzMzNCwyLzE4LzE5ODYsMi8xOS8xOTg2DQo

="

}

]

}

]

}

}

RESPONSE SUCCESS

[

{

"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="

}

]

}

]

}

}

RESPONSE SUCCESS

[

{

"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 Lockpath 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.

JSON REQUEST (ImportFile.json)

{

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

"VGV4dCxudW0scmljaHksaXB2NCxpcHY2LGRhdGUsZGF0ZXRpbWUNCnIxLDUxMCw8Yj5ib2xkIHRoaW5nPC9i PiwxOTIuMTY4LjIuNCwwMDE6MGRiODowMDAwOjAwNDI6MDAwMDo4YTJlOjAzNzA6NzMzNCwyLzE4LzE5ODYsM i8xOS8xOTg2DQo=",

"runAsSystem": "false"

}

]

**RESPONSE SUCCESS**

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 Lockpath 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

JSON REQUEST (DeleteRecord.json)

{

"componentId": "10001",

"recordId": "1"

}

RESPONSE SUCCESS

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)

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"

}

]

}

]

}

}

RESPONSE SUCCESS

[

{

"OperationSucceeded": true,

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

"RecordId": 2, "AttachmentInfo": {

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

"DocumentId": 20

}

}

]

# Field Types

## Unique Identifiers for Field Types

The Lockpath 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 |

Filtering

**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 Lockpath component column that will be sorted on. It describes the column ID where the data is stored. Because the Lockpath 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 Records**   |  |  |  | | --- | --- | --- | | **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 | | | | |
| **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>