

LifeCheck Technical Note

ID: TN008
Created: 19-February 2021
By: Chris Drew
Description: LifeCheck Web Services
Version: v5.4.4

Overview

The LifeCheck living data is held within a Microsoft SQL Server database and while access is possible using standard SQL queries, this approach is not recommended as the database format may be changed at any time to add new functionality.

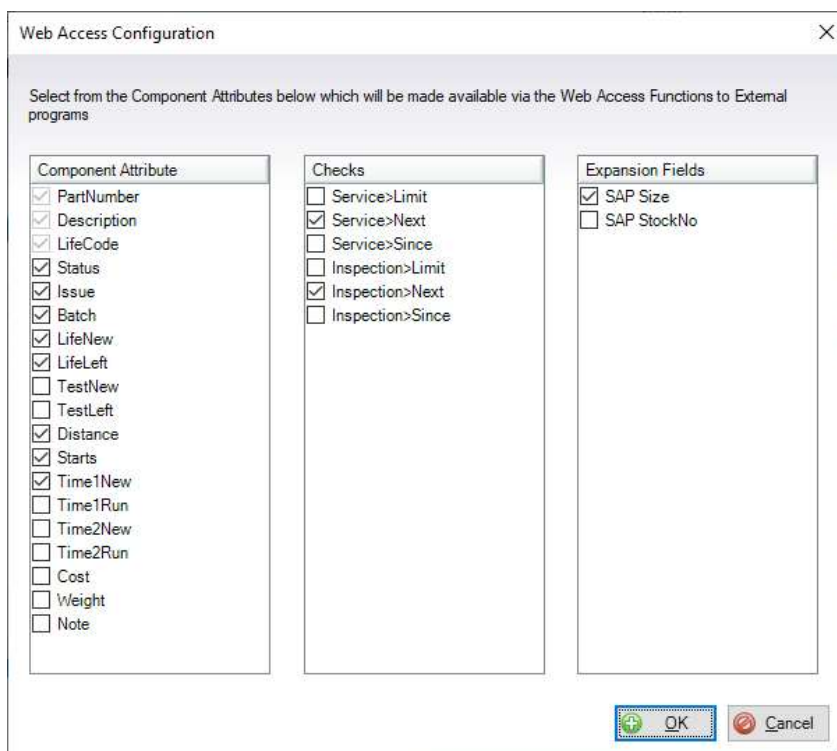
LifeCheck does however provide a **Web Service** by which the database may be accessed both to query and modify the living data. The range of functionality supported by this WebService is constantly being expanded however the following details the current functionality.

The LifeCheckWeb Service is called **LifeCheckWebService** and is implemented using C# .NET. Being a web service it may be called easily from HTML / JavaScript or alternatively (and recommended) may be called directly from a Microsoft .NET application. Information on how to call this web service together with coding samples may be found at the end of this document.

Configuring the Web API

As of v5.4.4 it is now possible to configure the data which is passed by the Web API from LifeCheck to the caller. Previously component information was limited to a fixed set of fields, this new mechanism aims to allow this to be defined dynamically and extended as required.

The Web API configuration is accessed from within LifeCheck using the **Administration>Web Api Configuration** menu option which displays the following window (subject to being licensed).



The image shows a Windows-style dialog box titled "Web Access Configuration". It contains three columns of checkboxes for selecting component attributes to be made available via the Web Access Functions to external programs. The first column, "Component Attribute", lists various attributes with checkboxes. The second column, "Checks", lists specific check types with checkboxes. The third column, "Expansion Fields", lists expansion fields with checkboxes. At the bottom right, there are "OK" and "Cancel" buttons.

Component Attribute	Checks	Expansion Fields
☒ PartNumber	☐ Service>Limit	☒ SAP Size
☒ Description	☒ Service>Next	☐ SAP StockNo
☒ LifeCode	☐ Service>Since	
☒ Status	☐ Inspection>Limit	
☒ Issue	☒ Inspection>Next	
☒ Batch	☐ Inspection>Since	
☒ LifeNew		
☒ LifeLeft		
☐ TestNew		
☐ TestLeft		
☒ Distance		
☒ Starts		
☒ Time1New		
☐ Time1Run		
☐ Time2New		
☐ Time2Run		
☐ Cost		
☐ Weight		
☐ Note		

The component attributes are divided into 3 distinct categories, Component Attributes, Component Check Attributes and Component Extension Fields. As a minimum, Part Number, Description and Life Code must be selected however the remainder of the fields are optional. Check those fields required and click OK to write the settings to the database. These will be immediately available to the (new) Web API calls.

Web API Functions

Before any other calls are made to the web service, **GetLoginContext** must be called. This function both validates the login credentials supplied and returns a context which must be passed to all subsequent calls to identify the department being used within the calls.

The following functions are included in the LifeCheck Web Services

GetLoginContext	Recover a login context for subsequent calls
CreatePart	Create a new PART in the database
GetDepartments	Return a list of the Departments defined within the database
GetLocations	Return a list of the locations defined for the specified Department
GetChecks	Return a list of periodic checks / services defined
CreateComponent	Create a component in the database
ResetCheck	Reset a check for a component
GetComponent	Return details of the specified component

GetLoginContext

```
int GetLoginContext (int version, String department, String password);
```

Description

Validate the login credentials and return a login context on success.

Parameters

Version	Integer version number of the API to use
department	Name of the department to log in under
password	Password for the above department (may be empty)

Returns

int32 Returned Login Context

If -1 is returned an error has occurred during the login process. This may be either that the login credential do not match those held in the LifeCheck database or that a connection to the LifeCheck database could not be established.

CreatePart

```
string CreatePart (int loginContext,  
                  String partNumber ,  
                  String mfrPartNumber ,  
                  String description ,  
                  bool lifed ,  
                  bool isAssembly)
```

Description

Create a new part definition in the database.

Parameters

loginContext	Login context returned from GetLoginContext
partNumber	Textual part number for part being created

mfrPartNumber	Textual manufacturers part number
description	Textual description of the part
lifer	True if the part is lifer, false otherwise
isAssembly	True if the part is an assembly, false otherwise

Returns

string Empty string if no error, error text otherwise

GetPart

List<string> GetPart (int loginContext, string partNumber)

Description

Return information about the part.

Parameters

loginContext	Login context returned from GetLoginContext
partNumber	Textual part number for part in question.

Returns

List of string attributes of the part if no error, empty string otherwise

GetDepartments

List<String> GetDepartments (int loginContext)

Description

Return a list of departments defined in the database.

Parameters

loginContext	Login context returned from GetLoginContext
--------------	---

Returns

String Array Array of Strings one per department

GetLocations

List<String> GetLocations (int loginContext)

Description

Return a list of departments defined in the database.

Parameters

loginContext	Login context returned from GetLoginContext
--------------	---

Returns

List<String> Array of Strings one per location in department

GetChecks

List<String> GetChecks (int loginContext)

Description

Return a list of Checks defined in the database.

Parameters

loginContext Login context returned from GetLoginContext

Returns

String Array of Strings one per defined check

CreateComponent (Depraved, use CreateComponentEx instead)

ResetCheck

```
string ResetCheck (int loginContext,  
                  String checkName,  
                  String partNumber,  
                  String lifecode,  
                  String serviceDate)
```

Description

Reset a check for the specified component

Parameters

loginContext	Login context returned from GetLoginContext
checkName	Name of check to be reset
partNumber	Part number of the component to have check reset
Lifecode	Unique life code { serial number} for the component to have its check reset
serviceDate	Date of check reset specify as YYYY-MM-DD

Returns

String Empty string if no error, error text otherwise

GetComponents

```
List<WSComponent> GetComponents (int loginContext, String partNumber)
```

Description

Get a list of component instances of the specified part

Parameters

loginContext	Login context returned from GetLoginContext
partNumber	Part number to return component instances of

Returns

List<WSComponent> List of Components (instances of the specified part).
See Appendix A for a definition of the WSComponent object.

GetComponent (Depraved, use GetComponentEx instead)

ModifyPart

```
string ModifyPart (int loginContext,
```

```
String oldPartNumber,  
String newPartNumber,  
String newDescription );
```

Parameters

oldPartNumber	Existing Part Number of part to be modified
newPartNumber	New Part Number to be assigned to the above, leave blank if not being changed
newDescription	New Description to be assigned to the above, leave blank if not being changed

Returns

string Empty string if no error, error text otherwise

ModifyComponentLifeCode

```
string ModifyComponentLifeCode (int loginContext,  
String partNumber,  
String oldLifeCode,  
String newLifeCode);
```

Parameters

partNumber	Existing Part Number of part to be modified
lifeCode	Life code of existing component
newLifeCode	New Life Code to be assigned to the above, leave blank if not being changed

Returns

string Empty string if no error, error text otherwise

ModifyComponentLifeLimits

```
string ModifyComponentLifeLimits (int loginContext,  
String partNumber,  
String lifeCode,  
Float lifeNewRace,  
Float lifeNewTest,  
Float serviceNewRace,  
Float serviceNewTest,  
String reason,  
Bool isLifed,  
Bool relifeAll);
```

Parameters

partNumber	Existing Part Number of part to be modified
lifeCode	Life code of existing component
lifeNewRace	New Race Limit to be assigned to the above, leave blank if not being changed
lifeNewTest	New Test Limit to be assigned to the above, leave blank if not being changed
serviceNewRace	New Service Race Limit to be assigned to the above, leave blank if not being changed
serviceNewTest	New Service Test Limit to be assigned to the above, leave blank if not being changed
reason	textual reason for the changes – this will be logged in the history of the component
islifed	

relifAll If true ALL instances of this component will have the changes applied, otherwise if false just the specified component will be re-lifed,

Returns

string Empty string if no error, error text otherwise

GetUpdatedComponents

This function should be called to return a list of components which have been modified since the specified date passed to the function.

Calling Format

List<int> GetUpdatedComponents (int loginContext, DateTime sinceDate)

Description

Get a list of component id's for components which have been updated since the specified date

Parameters

loginContext	Login context returned from GetLoginContext
sinceDate	Date to return updates after

Returns

List<int> List of Component Ids (for use in GetComponentEx below)

GetComponentEx

Calling Format

List<string,string> GetComponentEx (int context, int componentId, string partNumber, string lifecode)

Description

Return details of the specified component. If componentId is set to 0 then part number and lifecode will be used to identify the required component otherwise the Component Id will be used. Component Id is returned by GetUpdatedComponents above. Either componentId OR part number / lifecode should be specified.

Parameters

context	Login context returned from GetLoginContext
componentId	Id of the component as returned from GetUpdatedComponents
partNumber	Part number of component to return
lifeCode	Life Code of the component to return

Returns

List<int> List of Component Ids (for use in GetComponentEx below)

CreateComponentEx

Calling Format

string CreateComponent (int loginContext, List<string, string> attributes, bool createPart);

Description

This function may be used in preference to the existing CreateComponent function to allow additional fields to be set. The attributes list is in the form 'name|value' where 'name' is the name of the attribute to set and 'value' is the associated value for that attribute. As a minimum you must pass the attributes defined as mandatory in the 'ComponentAttributes.cs' file.

If the part specified does NOT exist it may be created based on the setting for the 'createPart' field

Parameters

context	Login context returned from GetLoginContext
attributes	List of key/value attributes and their values to be set
createPart	True if the part should be created if it does not already exist

Returns

string Empty string if no error, error text otherwise

CreateComponentEx

Calling Format

```
string CreateComponentEx (int loginContext,  
                          string inLocation,  
                          string partNumber,  
                          string lifecode,  
                          List<string, string> attributes);
```

Description

This function may be used in preference to the existing CreateComponent function to allow additional fields to be specified. The attributes list is in the form 'name|value' where 'name' is the name of the attribute to be set and 'value' is the associated value for that attribute.

Parameters

context	Login context returned from GetLoginContext
inLocation	Name of the location (for the logged in department) within which the component will be created UNLESS the attributes CA_parentpart and CA_parentlifecode are specified, in which case this attribute is ignored and the new component is created as a child of the parent specified and in the same location as the parent component. If inLocation is blank or not found, the component will be created in the 'default' location for the logged in department.
partNumber	Part number of component to create
lifeCode	Life Code of the component to create
attributes	List of key/value attributes and their values to be set (see Appendix A)

Returns

string Empty string if no error, error text otherwise

Note:

Some attributes cannot be set in this way – these are

- CA_lifeleft
- CA_testleft
- CA_lastnote
- Check > name > CA_checktonext
- Check > name > CA_checksince

UpdateComponentEx

Calling Format

```
string UpdateComponentEx (int loginContext,  
                          int componentId,  
                          string partNumber,  
                          string lifecode,  
                          List<string, string> attributes);
```

Description

This function may be used in preference to the existing CreateComponent function to allow additional fields to be specified. The attributes list is in the form 'name|value' where 'name' is the name of the attribute to be set and 'value' is the associated value for that attribute. If componentId is set to 0 then part number and lifecode will be used to identify the required component otherwise the Component Id will be used

Parameters

context	Login context returned from GetLoginContext
componentId	Id of the component as returned from GetUpdatedComponents
partNumber	Part number of component to return
lifeCode	Life Code of the component to return
attributes	List of key/value attributes and their values to be set

Returns

string Empty string if no error, error text otherwise

Note:

Some attributes cannot be set in this way – these are

- CA_lifeleft
- CA_testleft
- CA_lastnote
- Check > name > CA_checktonext
- Check > name > CA_checkssince

Appendix A

The WSComponent Object

The WSComponent is a simple object intended to provide a wrapper around the various properties associated with a Component. On including a reference to the LifeCheckWebService, the required definition will be held within the Reference.cs file generated during the include. The following properties are defined within the WSComponent object.

string PartNumber	Part number of the Part of which this is a component instance
string Description	Description for the Part of which this is a component instance
string LifeCode	Life Code / Serial Number of this component
string iSsueNumber	Issue Number of the component
string BatchNumber	Batch Number of the Component
float LifeNew	Life Limit of the component in Race conditions (if a lifed component)
float LifeLeft	Remaining life of this component in Race conditions (if a lifed component)
float LifeTestNew	Life Limit of the component in Test conditions (if a lifed component)
float LifeTestLeft	Remaining life of this component in Test conditions (if a lifed component)
float LifeWeightedLimit	Life Limit of the component after (any) weighting factor has been applied
float LifeWeightedRun	Distance run by this component after (any) weighting factor adjustments.
bool Accident	True if this component has been flagged as potentially accident damaged.
int StartCount	Count of starts recorded.
DateTime DateAdded	Date and time the component was created within LifeCheck
string Status	Textual Status for the component. May be one of "Active"; "Accident Damaged"; "FAILED"; Retired"; "To Service"; "Out to Service";
int Count	Count of the instances of this component only applicable to Sundries where the item is counted.

Appendix B

Component Attribute Strings

The ComponentAttributes.cs file – this file contains a number of definitions which should be used to ensure smooth communication between LifeCheck and the WebAPI. It defines the names of the various attributes which may be passed to and from LifeCheck. The content of the file is shown below...

```
// Delimiter which will be used between attribute names
public static string CA_delimiter = "|";

// Component Attribute Names
public static string CA_partnumber = "PartNumber";
public static string CA_description = "Description";
public static string CA_lifecode = "LifeCode";
public static string CA_location = "Location";
public static string CA_status = "Status";
public static string CA_issuenumber = "Issue";
public static string CA_batchnumber = "Batch";
public static string CA_lifenew = "LifeNew";
public static string CA_lifefirst = "LifeLeft";
public static string CA_testnew = "TestNew";
public static string CA_testleft = "TestLeft";
public static string CA_startdist = "StartDistance";
public static string CA_distrun = "Distance";
public static string CA_starts = "Starts";
public static string CA_time1new = "Time1New";
public static string CA_time1run = "Time1Run";
public static string CA_time2new = "Time2New";
public static string CA_time2run = "Time2Run";
public static string CA_cost = "Cost";
public static string CA_weight = "Weight";
public static string CA_lastnote = "Note";
public static string CA_parentpart = "ParentPart";
public static string CA_parentlifecode = "ParentLifeCode";

/// <summary>
/// These are the prefix and suffix for checks and Component extension fields
/// For example a service check limit attribute will be 'Check>Service>Limit'
/// </summary>
public static string CA_extensionfield = "Ext>";
public static string CA_check = "Check>";
public static string CA_checklimit = ">Limit";
public static string CA_checktonext = ">Next";
public static string CA_checksince = ">Since";

/// <summary>
/// These are the names of the default component attributes which will be selected initially.
/// </summary>
public static string[] DefaultAttributes = new string[]
{
    CA_partnumber, CA_description, CA_lifecode, CA_location, CA_status, CA_issuenumber, CA_batchnumber, CA_lifenew, CA_lifefirst,
    CA_distrun
};

/// <summary>
/// These are the name of the MANDATORY component attributes
/// </summary>
public static string[] MandatoryAttributes = new string[]
{
    CA_partnumber, CA_description, CA_lifecode, CA_location
};

/// <summary>
/// This is a list of the attribute names which will be shown on the Web API configuration screen
/// </summary>
public static string[] Names = new string[]
{
    CA_partnumber, CA_description, CA_lifecode, CA_location, CA_status, CA_issuenumber, CA_batchnumber, CA_lifenew, CA_lifefirst,
```

```
CA_testnew, CA_testleft, CA_startdist, CA_distrun, CA_starts, CA_time1new, CA_time1run, CA_time2new, CA_time2run, CA_cost,  
CA_weight, CA_lastnote, CA_parentpart, CA_parentlifecode  
};
```

Appendix C Visual Studio 2010 C# Examples

To make use of the LifeCheckWebService within a C# .NET application using Visual Studio 2010, you must first include a Web Reference to the web service which must have been previously deployed to a Web Site or Web Application with connectivity to the LifeCheck Database.

The LifeCheck database is defined within the web.config file shipped with the web service and must be configured by hand to allow connection to the LifeCheck database. The file may either be edited within IIS or may be manually edited using any text editor including Notepad.

The following shows the settings which must be defined for your specific LifeCheck installation.

```
<appSettings>
<add key="DatabaseServer" value="localhost\sqlexpress" />
<add key="DatabaseName" value="lifecheckv4" />
<add key="DatabaseTrustedConnection" value="0" />
<add key="DatabaseUser" value="lifecheck" />
<add key="DatabasePassword" value="lifecheck999" />
</appSettings>
```

Database Server is the Instance Name of the Microsoft SQL Server holding the LifeCheck database
Database Name should not be modified from the default value of lifecheckv4

DatabaseTrustedConnection should be 0 when using SQL Server Authentication (recommended) or 1 if using Windows Authentication.

DatabaseUser is the name of the SQL Server user that is to be used to access the LifeCheck database. It is only required where DatabaseTrustedConnection above has been set to 0.

DatabasePassword is the password for the SQL Server user defined above. It is only required where DatabaseTrustedConnection above has been set to 0.

Establishing a Database Connection

Before any other calls made be made to the Web Service, it is necessary to login to LifeCheck and obtain a login context. This is done using the following code snippet.

```
LifeCheckWebService.LifeCheckWebService ws = new
LifeCheckWebService.LifeCheckWebService();
int loginContext = ws.GetLoginContext(100, username, password);
```

Where 100 is the version of the LifeCheckWebService required, username is the user or department to login as and password is (any) password required for that department. The Web Service will return a login context if successful or 0 if the call failed. The most likely cause of a failure here is invalid credentials or an error in the database configuration held in the web.config file.

The login context should be saved as it will be required in all future calls to the Web Service.

Further Programming

All of the other LifeCheckWebService functions are called in the same way as the GetLoginContext call detailed previously. Note however that some calls will require information which may only be recovered by a previous call. For example, when creating a Component it is necessary to pass a valid location name as inLocation. The names of the locations available for the currently logged in department are returned by the GetLocations function.