# CURL API EXTENSION DOC

In order to support new Curl options the following steps can be used as a guide. Note this is particular to version 7.30.0 and the steps might change with higher versions. The example is for adding CURLOPT\_KEYTAB\_LOCATION as a string option but steps are almost identical for different types of option additions including new Boolean values.

Now in order to create the option I made the following changes:-

1. Added CINIT(KEYTAB\_LOCATION, OBJECTPOINT, 186) in curl.h. This allows a new definition called CURLOPT\_KEYTAB\_LOCATION that can be used in a client
2. #define MAX\_CURL\_KEYTAB\_LOCATION\_LENGTH 256 added in urldata.h, gives max size of keytab location length
3. char keytab[MAX\_CURL\_KEYTAB\_LENGTH] added in url.c for keytab location
4. char \*keytab\_location added in connectdata stucture in urldata.h(needed for use in input\_negoitate function in http\_negotiate
5. STRING\_KEYTAB\_LOCATION added in enum dupstring in urldata.h in last line. Typically this is used to get the option from curlopt\_keytab\_location
6. if(data->set.str[STRING\_KEYTAB\_LOCATION] != NULL) {

strncpy(keytab, data->set.str[STRING\_KEYTAB\_LOCATION], MAX\_CURL\_KEYTAB\_LOCATION\_LENGTH);

user[MAX\_CURL\_USER\_LENGTH-1] = '\0'; /\*To be on safe side\*/

}

conn->keytab\_location = strdup(keytab);

added in create\_conn after set\_userpassword() call in the same function. This would set the keytab location passed in the option into the required connecdata data (i.e keytab\_location)

7. Create a Boolean value in conn->bits object of urldata.h called user\_keytab and write the following logic to populate it inside allocate\_conn function of url.c. This Boolean value can be used later on to judge if keytab option is set or not.

8. case CURLOPT\_KEYTAB\_LOCATION:

/\*

\*authentication keytab location to use in operation, added by arunav sanyal

\*/

result = setstropt(&data->set.str[STRING\_KEYTAB\_LOCATION], va\_arg(param, char \*));

break;

added in function curl\_setopt in url.c order to pass keytab location into data object, which is then used in step 6.

9. Add the appropriate Curl\_Safefree call inside url.c to remove it in remove\_conn and reuse\_conn. E.g. Curl\_safefree(conn->keytab\_location);

Typically the changes for other types of arguments are different and this should be used as a guideline for any further curl changes.

Also an important change that is required is when the same connection is reused. Typically if you add a new option, in the function reuse\_conn in url.c, add the following code:-

conn->bits.user\_keytab = old\_conn->bits.user\_keytab;

/\* reuse keytab location for kerberos authentication \*/

if(conn->bits.user\_keytab) {

Curl\_safefree(conn->keytab\_location);

conn->keytab\_location = old\_conn->keytab\_location;

old\_conn->keytab\_location = NULL;

}

For any option you deem fit to be reusable.

10. Define conn->bits.user\_keytab = ((NULL != (data->set.str[STRING\_USERNAME])) && (NULL != data->set.str[STRING\_KEYTAB\_LOCATION]))?TRUE:FALSE; inside url.c