PKCS#11 Asynchronous Operations

Overall concept:

Open a session in asynchronous mode that allows any function to return CKR\_PENDING to indicate that the operation is still running. Use a new function to determine if the call has finished and obtain the result.

1. Section 3.3 add CKF\_ASYNC\_SESSION to flags to table 7

|  |  |  |
| --- | --- | --- |
| CKF\_ASYNC\_SESSION | 0x00000008 | Ture if the session is asynchronous; false if the session is synchronous |

1. Section 5.1.6 add CKR\_PENDING: This value is returned if the operation is running asynchronously.
2. Add CKR\_PENDING as an allowed return code to all functions in section 5 that require a session handle. Add CKR\_OPERATION\_ACTIVE active to any function in section 5 that requires a session handle and does not already include CKR\_OPERATION\_ACTIVE as a return code.
3. A token MAY return CKR\_PENDING if the session is asynchronous and the token determines that the operation will take a long time to conclude.
4. Add a new structure

typedef struct CK\_ASYNC\_DATA {

CK\_ULONG ulVersion;

CK\_BYTE\_PTR pValue;

CK\_ULONG ulValue;

CK\_OBJECT\_HANDLE\_PTR phObject;

CK\_OBJECT\_HANDLE\_PTR phAdditionalObject;

} CK\_ASYNC\_DATA;

The fields of the CK\_ASYNC\_DATA structure have the following meaning:

|  |  |
| --- | --- |
| ulVersion | version of this structure; always 0 for this version of Cryptoki |
| pValue | pointer to a buffer to contain the result of the operation |
| ulValue | size of the buffer pointed to by pValue |
| phObject | pointer to receive the handle for an object resulting from the operation |
| phAdditionalObject | Pointer to receive the handle for an additional object resulting from the operation |

1. Add a new function

CK\_DECLARE\_FUNCTION(CK\_RV, C\_AsyncComplete) (

CK\_SESSION\_HANDLE hSession,

CK\_UTF8CHAR\_PTR pFunctionName,

CK\_ASYNC\_DATA\_PTR pResult

) ;

**C\_AsyncComplete** checks if the function identified by pFunctionName has completed an asynchronous operation and, if so, returns the associated result(s). *hSession* is the session’s handle; *pFunctionName* is the name of the function whose state is being queried; *pResult* is a pointer to a structure to contain the result if the function has completed.

Return values: This function’s return values are as returned by the function identified by *pFunctionName*.

Example:

|  |
| --- |
| CK\_SESSION\_HANDLE hSession;  CK\_OBJECT\_HANDLE hKey;  CK\_MECHANISM mechanism = {  CKM\_DES\_MAC, NULL\_PTR, 0  };  CK\_BYTE data[] = {...};  CK\_BYTE mac[4];  CK\_ULONG ulMacLen;  CK\_RV rv;  .  .  rv = C\_SignInit(hSession, &mechanism, hKey);  if (rv == CKR\_OK) {  rv = C\_SignUpdate(hSession, data, sizeof(data));  while (rv == CKR\_PENDING)  {  rv = C\_AsyncComplete(hSession, (CK\_UTF8CHAR\_PTR)"C\_SignUpdate", NULL\_PTR);  /\* rv will contain CKR\_PENDING if the operation is still running or it will contain the  return code from the C\_SignUpdate operation \*/  }  .  .  ulMacLen = sizeof(mac);  rv = C\_SignFinal(hSession, mac, &ulMacLen);    if (rv == CKR\_PENDING)  {  CK\_ASYNC\_DATA result;  result.ulVersion = 0;  result.pValue = NULL\_PTR;  result.ulValue = 0;  rv = C\_AsyncComplete(hSession, (CK\_UTF8CHAR\_PTR)"C\_SignFinal", &result);  if (rv == CKR\_BUFFER\_TOO\_SMALL)  {  result.pValue = (CK\_BYTE\_PTR)malloc(result.ulValue);  rv = C\_AsyncComplete(hSession, (CK\_UTF8CHAR\_PTR)"C\_SignFinal", &result);  /\* if rv is CKR\_OK, result will contain the mac value \*/  .  .  }  }  } |

1. C\_CloseSession and C\_CloseAllSessions cancel all pending operations.
2. C\_SessionCancel works as usual and will cancel a pending operation matching the input flags.
3. 5.6.6 C\_GetOperationsState will pause all pending operations and serialize them.
4. 5.6.7 C\_SetOperationsState will reestablish pending operations and resume them.