Elektrobit

EB tresos® AutoCore

EBtresosAutoCore: Cryptographic Primitive Library (Cpl)

Module Version 1.3.2

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential and proprietary information

Table of Contents

1. Introduction

1.1. Overview

The Cpl module provides access to several cryptographic primitives. It acts as a lower interface to the Cal module, which in turn provides cryptographic functionalities to the user.

Some Cpl primitives may use other Cpl primitves themeselves. Therefore these Cpl primitves call the service functions of the Cal module, which interact with the used Cpl primitves. The following Cpl primitives call API functions from the Cal:

- presently none

2. Integration

2.1. Cal

To be able to use the cryptographic functionality provided by this Cpl module, a Cal module is needed. In that Cal module, it is necessary to create configurations which link to Cpl configurations.

2.1.1. Mapping of Cryptographic Primitives to Cal Services

The following is a list of the cryptographic primitives which are accessible via the Cpl interface. It also contains the string which has to be entered into the field "Primitive name" in the Cal configuration created for the service.

Functionality	Cal Service	Cpl Primitive Name
Decompression (via LZMA)	CalDecompress	LZMADecompress
Compression (via LZSS)	CalCompress	LZSSCompress
Decompression (via LZSS)	CalDecompress	LZSSDecompress
Compression (via RLE)	CalCompress	RLECompress
Decompression (via RLE)	CalDecompress	RLEDecompress
Decompression (via ARLE)	CalDecompress	ARLEDecompress

Table 2.1. Cpl primitives

2.2. Cpl

To be able to use the cryptographic functionality provided by this Cpl module, it is necessary to create Cpl configurations to the primitives needed and linked to the Cal.

2.2.1. LZMADecompress

In order to configure the LZMA Decompress first a configuration must be added. This is achieved clicking on tab "Lzma Decompress configurations" inside the Cpl module in Tresos. To add a configuration button "add new element" must be pressed (green + button). Depending how many LZMA compressed arrays there are, that many configurations must be added.

The LZMA algorithm is defined by the following parameters:

LZMA LP parameter. Number of literal position bits LP affects what kind of alignment in the uncompressed data is assumed when encoding literals.

LZMA LC parameter. Number of literal context bits How many of the highest bits of the previous uncompressed byte (the literal) are taken into account when predicting the bits of the next literal.

LZMA PB parameter. PB affects what kind of alignment in the uncompressed data is assumed in general.

LZMA Dictionary Size parameter. The dictionary size affects what how many pairs of length and distance can be stored at one time.

2.2.2. LZSSCompress

Via `CplLzssCompressLengthBitsPerBlock` it is required to specify the number of bits used to encode the 'length' of a sequence reference. References are represented by a block of 2 bytes in the compressed data.

NOTE

Cal parameter `'CalCompressMaxCtxBufByteSize'` vs. Cpl LZSSCompress configuration

The context buffer configured for the Cal compression service must be at least 48 bytes. Depending on µC architecture and compiler options it may be necessary to specify a larger value.

2.2.3. LZSSDecompress

Via `CplLzssDecompressLengthBitsPerBlock` it is required to specify the number of bits used to encode the 'length' of a sequence reference. References are represented by a block of 2 bytes in the compressed data.

NOTE

Cal parameter `'CalDecompressMaxCtxBufByteSize'` vs. Cpl LZSSDecompress configuration

The context buffer configured for the Cal decompression service must be at least `(((1 << (16-CplLzssDecompressLengthBitsPerBlock)) - 1) + 29)` bytes. Depending on µC architecture and compiler options it may be necessary to specify a larger value.

2.2.4. RLECompress

Via `CplRleCompressVariant` it is required to specify the variant of RLE algorithm to use. The choices are

- CPL_RLE_VARIANT_8BIT
- CPL_RLE_VARIANT_CTRLBIT

NOTE

Cal parameter `'CalCompressMaxCtxBufByteSize'` vs. Cpl RLECompress configuration

The context buffer configured for the Cal compression service must be at least 13 bytes. Depending on µC architecture and compiler options it may be necessary to specify a larger value.

2.2.5. RLEDecompress

Via `CplRleDecompressVariant` it is required to specify the variant of RLE algorithm to use. The choices are

- CPL_RLE_VARIANT_8BIT
- CPL_RLE_VARIANT_CTRLBIT

NOTE

Cal parameter `'CalDecompressMaxCtxBufByteSize'` vs. Cpl RLEDecompress configuration

The context buffer configured for the Cal decompression service must be at least 14 bytes. Depending on µC architecture and compiler options it may be necessary to specify a larger value.

2.2.6. ARLEDecompress

The ARLE decompression algorithm is not configurable.

NOTE

Cal parameter `'CalDecompressMaxCtxBufByteSize'` vs. Cpl ARLEDecompress configuration

The context buffer configured for the Cal decompression service must be at least 12 bytes. Depending on µC architecture and compiler options it may be necessary to specify a larger value.

2.3. MemMap.h

The module expects the file "MemMap.h" to exist. In this file, memory mapping can be performed for the different memory blocks of this module:

- CPL_START_SEC_CODE/CPL_STOP_SEC_CODE
- CPL_START_SEC_CONST_UNSPECIFIED/CPL_STOP_SEC_CONST_UNSPECIFIED

2.4. Handling of small output buffer in LZMA algorithm

The LZMA algorithm has the following behaviour in case the Update call returns CAL_E_SMALL_BUFFER;

- inputBuf: address of the new data to be decompressed
- inputBufLen: length of the successfully decompressed data
- outputBuf: location where the new output data can be written
- outputBufLen: length of the decopressed data done in the current Update

In case the Update primitive returns CAL_E_SMALL_BUFFER a new call to the Update primitive is needed, to get the rest of the decopressed data. The error CAL_E_SMALL_BUFFER occured while the provided outputBuf was not enough to write the decopressed data.

The Update primitive needs to be called until it returns CAL_E_OK(all output data from the previous calls was written). In case the Update primitive is called with new input data, it would be processed only after all previously unwritten data was written to the output buffer.

3. API specification

3.1. Application programming interface (API)

3.1.1. Type definitions

3.1.1.1. CProbType

Purpose	Data type to hold the probability for one symbol.
Type	`uint16`

3.1.2. Functions

3.1.2.1. Cpl_ARLEDecompressFinish

Purpose	Finish ARLE decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_ARLEDecompressFinish`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the ARLE decompression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the ARLE decompression is stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of decompressed data (the result)

		which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
	`CAL_E_SMALL_BUFFER`	The provided buffer oputBuf is too small to store the complete result.
Description	This function requests the finishing of the ARLE decompression computation. The finish function of the configured primitive is called and its return value is returned.	

3.1.2.2. Cpl_ARLEDecompressStart

Purpose	Start ARLE decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_ARLEDecompressStart`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the ARLE decompression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the ARLE decompression is stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
Description	This function requests the start of the ARLE decompression for the given configuration. The start function of the configured primitive is called and its return value is returned.	

3.1.2.3. Cpl_ARLEDecompressUpdate

Purpose	Update ARLE decompression computation.

Synopsis	`Cal_ReturnType` **`Cpl_ARLEDecompressUpdate`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , const uint8 * iputBuf , uint32 * iputBufLen , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the ARLE decompression computation.
	`iputBuf`	Holds a pointer to the data that shall be processed.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the ARLE decompression is stored.
	`iputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by iputBuf. On returning from this function the length of data from buffer iputBuf that was already processed/decompressed shall be stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
	`CAL_E_SMALL_BUFFER`	The provided buffer oputBuf is too small to store the complete result. This means that not all data in iputBuf could be processed/decompressed. Only iputBufLen bytes

		of iputBuf (on returning) were decompressed.
Description	This function requests the update of the ARLE decompression computation for the given data. The update function of the configured primitive is called and its return value is returned.	

3.1.2.4. Cpl_LZMADecompressFinish

Purpose	Finish LZMA decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZMADecompressFinish`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , uint8 * outputBuf , uint32 * outputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZMA decompression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZMA decompression is stored.
	`outputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by outputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer outputBuf shall be stored.
Parameters (out)	`outputBuf`	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
Description	This function requests the finishing of the LZMA decompression computation. The finish function of the configured primitive is called and its return value is returned.	

3.1.2.5. Cpl_LZMADecompressStart

Purpose	Start LZMA decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZMADecompressStart`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZMA decompression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZMA decompression is stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
Description	This function requests the start of the LZMA decompression for the given configuration. The start function of the configured primitive is called and its return value is returned.	

3.1.2.6. Cpl_LZMADecompressUpdate

Purpose	Update LZMA decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZMADecompressUpdate`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , const uint8 * inputBuf , uint32 * inputBufLen , uint8 * outputBuf , uint32 * outputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZMA decompression computation.
	`inputBuf`	Holds a pointer to the data that shall be processed.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZMA decompression is stored.
	`inputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by

		inputBuf. On returning from this function the length of data from buffer inputBuf that was already processed/decompressed shall be stored.
	`outputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by outputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer outputBuf shall be stored.
Parameters (out)	`outputBuf`	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
	`CAL_E_SMALL_BUFFER`	The provided buffer outputBuf is too small to store the complete result. This means that not all data in inputBuf could be processed/decompressed. Only inputBufLen bytes of inputBuf (on returning) were decompressed.
Description	This function requests the update of the LZMA decompression computation for the given data. The update function of the configured primitive is called and its return value is returned.	

3.1.2.7. Cpl_LZSSCompressFinish

Purpose	Finish LZSS compression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZSSCompressFinish`** `( const void * cfgPtr , Cal_CompressCtxBufType contextBuffer , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZSS compression computation.

Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZSS compression is stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of compressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the compressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
Description	This function requests the finishing of the LZSS compression computation. The finish function of the configured primitive is called and its return value is returned.	

3.1.2.8. Cpl_LZSSCompressStart

Purpose	Start LZSS compression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZSSCompressStart`** `( const void * cfgPtr , Cal_CompressCtxBufType contextBuffer );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZSS compression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZSS compression is stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
Description	This function requests the start of the LZSS compression for the given configuration. The start function of the configured primitive is called and its return value is returned.	

3.1.2.9. Cpl_LZSSCompressUpdate

Purpose	Update LZSS compression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZSSCompressUpdate`** `( const void * cfgPtr , Cal_CompressCtxBufType contextBuffer , const uint8 * iputBuf , uint32 * iputBufLen , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZSS compression computation.
	`iputBuf`	Holds a pointer to the data that shall be processed.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZSS compression is stored.
	`iputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by iputBuf. On returning from this function the length of data from buffer iputBuf that was already processed/compressed shall be stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of compressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the compressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.

	CAL_E_SMALL_BUFFER	The provided buffer oputBuf is too small to store the complete result. This means that not all data in iputBuf could be processed/ compressed. Only iputBufLen bytes of iputBuf (on returning) were compressed.
Description	This function requests the update of the LZSS compression computation for the given data. The update function of the configured primitive is called and its return value is returned.	

3.1.2.10. Cpl_LZSSDecompressFinish

Purpose	Finish LZSS decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZSSDecompressFinish`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	cfgPtr	Pointer to Cal module configuration which has to be used during the LZSS decompression computation.
Parameters (in,out)	contextBuffer	Holds a pointer to the buffer in which the context of the service that calls the LZSS decompression is stored.
	oputBufLen	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	oputBuf	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	CAL_E_OK	Request successful.
	CAL_E_NOT_OK	Request failed.
Description	This function requests the finishing of the LZSS decompression computation. The finish function of the configured primitive is called and its return value is returned.	

3.1.2.11. Cpl_LZSSDecompressStart

Purpose	Start LZSS decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZSSDecompressStart`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZSS decompression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZSS decompression is stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
Description	This function requests the start of the LZSS decompression for the given configuration. The start function of the configured primitive is called and its return value is returned.	

3.1.2.12. Cpl_LZSSDecompressUpdate

Purpose	Update LZSS decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_LZSSDecompressUpdate`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , const uint8 * iputBuf , uint32 * iputBufLen , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the LZSS decompression computation.
	`iputBuf`	Holds a pointer to the data that shall be processed.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the LZSS decompression is stored.
	`iputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by

		iputBuf. On returning from this function the length of data from buffer iputBuf that was already processed/decompressed shall be stored.
	oputBufLen	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	oputBuf	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	CAL_E_OK	Request successful.
	CAL_E_NOT_OK	Request failed.
	CAL_E_SMALL_BUFFER	The provided buffer oputBuf is too small to store the complete result. This means that not all data in iputBuf could be processed/ decompressed. Only iputBufLen bytes of iputBuf (on returning) were decompressed.
Description	This function requests the update of the LZSS decompression computation for the given data. The update function of the configured primitive is called and its return value is returned.	

3.1.2.13. Cpl_RLECompressFinish

Purpose	Finish RLE compression computation.	
Synopsis	Cal_ReturnType **Cpl_RLECompressFinish** (const void * cfgPtr , Cal_CompressCtxBufType contextBuffer , uint8 * oputBuf , uint32 * oputBufLen);	
Parameters (in)	cfgPtr	Pointer to Cal module configuration which has to be used during the RLE compression computation.

Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the RLE compression is stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of compressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the compressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
	`CAL_E_SMALL_BUFFER`	The provided buffer oputBuf is too small to store the complete result.
Description	This function requests the finishing of the RLE compression computation. The finish function of the configured primitive is called and its return value is returned.	

3.1.2.14. Cpl_RLECompressStart

Purpose	Start RLE compression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_RLECompressStart`** `( const void * cfgPtr , Cal_CompressCtxBufType contextBuffer );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the RLE compression computation.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the RLE compression is stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.

Description	This function requests the start of the RLE compression for the given configuration. The start function of the configured primitive is called and its return value is returned.

3.1.2.15. Cpl_RLECompressUpdate

Purpose	Update RLE compression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_RLECompressUpdate`** `( const void * cfgPtr , Cal_CompressCtxBufType contextBuffer , const uint8 * iputBuf , uint32 * iputBufLen , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	`cfgPtr`	Pointer to Cal module configuration which has to be used during the RLE compression computation.
	`iputBuf`	Holds a pointer to the data that shall be processed.
Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the RLE compression is stored.
	`iputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by iputBuf. On returning from this function the length of data from buffer iputBuf that was already processed/compressed shall be stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of compressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the compressed data shall be stored.
Return Value	Error value.	

	CAL_E_OK	Request successful.
	CAL_E_NOT_OK	Request failed.
	CAL_E_SMALL_BUFFER	The provided buffer oputBuf is too small to store the complete result. This means that not all data in iputBuf could be processed/ compressed. Only iputBufLen bytes of iputBuf (on returning) were compressed.
Description	This function requests the update of the RLE compression computation for the given data. The update function of the configured primitive is called and its return value is returned.	

3.1.2.16. Cpl_RLEDecompressFinish

Purpose	Finish RLE decompression computation.	
Synopsis	`Cal_ReturnType` **`Cpl_RLEDecompressFinish`** `( const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , uint8 * oputBuf , uint32 * oputBufLen );`	
Parameters (in)	cfgPtr	Pointer to Cal module configuration which has to be used during the RLE decompression computation.
Parameters (in,out)	contextBuffer	Holds a pointer to the buffer in which the context of the service that calls the RLE decompression is stored.
	oputBufLen	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	oputBuf	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	CAL_E_OK	Request successful.
	CAL_E_NOT_OK	Request failed.

	CAL_E_SMALL_BUFFER	The provided buffer oputBuf is too small to store the complete result.
Description	This function requests the finishing of the RLE decompression computation. The finish function of the configured primitive is called and its return value is returned.	

3.1.2.17. Cpl_RLEDecompressStart

Purpose	Start RLE decompression computation.	
Synopsis	Cal_ReturnType **Cpl_RLEDecompressStart** (const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer);	
Parameters (in)	cfgPtr	Pointer to Cal module configuration which has to be used during the RLE decompression computation.
Parameters (in,out)	contextBuffer	Holds a pointer to the buffer in which the context of the service that calls the RLE decompression is stored.
Return Value	Error value.	
	CAL_E_OK	Request successful.
	CAL_E_NOT_OK	Request failed.
Description	This function requests the start of the RLE decompression for the given configuration. The start function of the configured primitive is called and its return value is returned.	

3.1.2.18. Cpl_RLEDecompressUpdate

Purpose	Update RLE decompression computation.	
Synopsis	Cal_ReturnType **Cpl_RLEDecompressUpdate** (const void * cfgPtr , Cal_DecompressCtxBufType contextBuffer , const uint8 * iputBuf , uint32 * iputBufLen , uint8 * oputBuf , uint32 * oputBufLen);	
Parameters (in)	cfgPtr	Pointer to Cal module configuration which has to be used during the RLE decompression computation.
	iputBuf	Holds a pointer to the data that shall be processed.

Parameters (in,out)	`contextBuffer`	Holds a pointer to the buffer in which the context of the service that calls the RLE decompression is stored.
	`iputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by iputBuf. On returning from this function the length of data from buffer iputBuf that was already processed/decompressed shall be stored.
	`oputBufLen`	Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by oputBuf. On returning from this function the size of decompressed data (the result) which was written to the buffer oputBuf shall be stored.
Parameters (out)	`oputBuf`	Holds a pointer to the memory location where the decompressed data shall be stored.
Return Value	Error value.	
	`CAL_E_OK`	Request successful.
	`CAL_E_NOT_OK`	Request failed.
	`CAL_E_SMALL_BUFFER`	The provided buffer oputBuf is too small to store the complete result. This means that not all data in iputBuf could be processed/ decompressed. Only iputBufLen bytes of iputBuf (on returning) were decompressed.
Description	This function requests the update of the RLE decompression computation for the given data. The update function of the configured primitive is called and its return value is returned.	

4. Configuration specification

4.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
CplGeneral	1..1	
CplLzssCompress	1..1	The LZSS configuration for compression.
CplLzssDecompress	1..1	The LZSS configuration for decompression.
CplLzmaDecompress	1..1	The LZMA configuration for decompression.
CplRleCompress	1..1	The RLE configuration for compression.
CplRleDecompress	1..1	The RLE configuration for decompression.
CplArleDecompress	1..1	The ARLE configuration for decompression.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	**IMPLEMENTATION_CONFIG_VARIANT**
Label	Config Variant
Description	Select the configuration variant. Currently only PreCompile is supported.
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile

Range	VariantPreCompile

4.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version

Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	206
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

4.1.2. CplGeneral

Parameters included	
Parameter name	**Multiplicity**
CplVersionInfoApi	1..1

Parameter Name	**CplVersionInfoApi**	
Label	Version info API	
Description	Version info API	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

4.1.3. CplLzssCompress

Containers included		
Container name	**Multiplicity**	**Description**
CplLzssCompressConfig	0..32	

4.1.4. CplLzssCompressConfig

Parameters included	
Parameter name	**Multiplicity**
CplLzssCompressLengthBitsPerBlock	1..1

Parameters included	
CplLzssCompressIterationsBeforeInterruption	1..1

Parameter Name	CplLzssCompressLengthBitsPerBlock	
Label	Number of bits to encode the Length	
Description	The number of bits used to encode the 'length' in a two-byte block. The two-byte block references a previous sequence of bytes in the input stream. 2^2 lteq 'length' lteq 2^8. The remaining bits are used to encode the 'offset'.	
Multiplicity	1..1	
Type	INTEGER	
Default value	6	
Range	>=2 --- <=8	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	CplLzssCompressIterationsBeforeInterruption	
Label	Number of iterations before interruption	
Description	The number of prefix match loop iterations before the execution is interrupted.	
Multiplicity	1..1	
Type	INTEGER	
Default value	10	
Range	>=2 --- <=4294967295	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

4.1.5. CplLzssDecompress

Containers included		
Container name	**Multiplicity**	**Description**
CplLzssDecompressConfig	0..32	

4.1.6. CplLzssDecompressConfig

Parameters included	
Parameter name	**Multiplicity**
CplLzssDecompressLengthBitsPerBlock	1..1

Parameter Name	CplLzssDecompressLengthBitsPerBlock	
Label	BitsOfLength	
Description	The number of bits of 'length' in a two-byte (lenght/offset) encoding which denote a compressed sequence of data. Note: 2 greater than or equal to 'length' greater than or equal to 8.	
Multiplicity	1..1	
Type	INTEGER	
Default value	6	
Range	>=2 --- <=8	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

4.1.7. CplLzmaDecompress

Containers included		
Container name	**Multiplicity**	**Description**
CplLzmaDecompressConfig	0..32	

4.1.8. CplLzmaDecompressConfig

Parameters included	
Parameter name	**Multiplicity**
CplLzmaDecompressDictionarySize	1..1

Parameters included	
CplLzmaDecompressLP	1..1
CplLzmaDecompressLC	1..1
CplLzmaDecom-pressPB	1..1

Parameter Name	CplLzmaDecompressDictionarySize	
Label	DictionarySize	
Description	LZMA dictionary size. Valid values between: 2^12 - 2^32	
Multiplicity	1..1	
Type	INTEGER	
Default value	65536	
Range	>=4096 --- <=4294967296	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	CplLzmaDecompressLP	
Label	LP	
Description	LZMA LP parameter. Number of literal position bits LP affects what kind of alignment in the uncompressed data is assumed when encoding literals.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	>=0 --- <=4	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	CplLzmaDecompressLC

Label	LC
Description	LZMA LC parameter. Number of literal context bits How many of the highest bits of the previous uncompressed byte (the literal) are taken into account when predicting the bits of the next literal.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=0 --- <=8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**CplLzmaDecompressPB**
Label	PB
Description	LZMA PB parameter. PB affects what kind of alignment in the uncompressed data is assumed in general.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	>=0 --- <=4
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

4.1.9. CplRleCompress

Containers included		
Container name	**Multiplicity**	**Description**
CplRleCompressConfig	0..32	

4.1.10. CplRleCompressConfig

Parameters included	
Parameter name	**Multiplicity**
CplRleCompressVariant	1..1

Parameter Name	CplRleCompressVariant	
Label	Variant	
Description	The variant of the RLE.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	CPL_RLE_VARIANT_CTRLBIT	
Range	CPL_RLE_VARIANT_8BIT	
	CPL_RLE_VARIANT_CTRLBIT	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

4.1.11. CplRleDecompress

Containers included		
Container name	**Multiplicity**	**Description**
CplRleDecompressConfig	0..32	

4.1.12. CplRleDecompressConfig

Parameters included	
Parameter name	**Multiplicity**
CplRleDecompressVariant	1..1

Parameter Name	CplRleDecompressVariant
Label	Variant

Description	The variant of the RLE.
Multiplicity	1..1
Type	ENUMERATION
Default value	CPL_RLE_VARIANT_CTRLBIT
Range	CPL_RLE_VARIANT_8BIT CPL_RLE_VARIANT_CTRLBIT
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

4.1.13. CplArleDecompress

Containers included		
Container name	**Multiplicity**	**Description**
CplArleDecompress-Config	0..1	

4.1.14. CplArleDecompressConfig

4.1.15. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the Cpl can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	