

VOLKSWAGEN

AKTIENGESELLSCHAFT

Dokumentation und Anleitung ODXCreate

Autor	Schertler, Jens
Abt./OE	I/EE-87
Telefon	+49-841-8939682
E-Mail	Jens.Schertler@audi.de
Erstausgabe	14.08.2013
Änderungsstand	19.05.2020
Programmversion	2.38

Inhaltsverzeichnis

1	Vorbemerkung.....	4
2	Die Funktionen des ODXCreate	6
1	Wichtige Anwendungshinweise	7
2	Starten des ODXCreate.....	8
2.1	Select.....	8
2.2	Festlegung der Basiskonfiguration	9
3	Allgemeine Elemente des ODXCreate.....	11
3.1	Menüleiste.....	11
3.2	Werkzeugleiste.....	12
4	General Informations.....	14
4.1	Supplier/Creator	14
4.2	General	15
4.3	Initial values	17
4.4	External tools	18
5	Logistic and ECU-Specific für ECU-MEMSpec V3.0.....	19
5.1	Logistics Informations.....	19
5.2	Security Properties	20
5.3	AdressAndLengthFormatIdentifier	20
5.4	Expected-Ids	21
5.5	Links to ODX-Diaglayer Container.....	22
5.6	Programming conditions.....	23
5.7	User Input Fenster.....	25
6	ECU-MEMORY-Configuration für ECU-MEMSpec V3.0.....	26
6.1	Memory-Layout	26
6.2	Partial Programming.....	27
6.3	Memory block.....	27
6.4	Add new mem block	31
6.5	Segment.....	32
7	XML Output.....	34
8	Logininfo	35
9	Logistic and ECU-Specific für ECU-MEMSpec V2.0.....	36
9.1	AdressAndLengthFormatIdentifier	36
10	ECU-MEMORY-Configuration für ECU-MEMSpec V2.0.....	37
10.1	Memory block.....	37
11	Logistic and ECU-Specific für ECU-MEMSpec V3.1	39
11.1	AdressAndLengthFormatIdentifier	39
12	ECU-MEMORY-Configuration für ECU-MEMSpec V3.1	41
12.1	Memory block.....	41
13	Logistic and ECU-Specific für ECU-MEMSpec V4.0.....	42
13.1	Security Properties	43
13.1.1	Checksum/Signature	43
14	Logistic and ECU-Specific für Bootloader-Datensätze nach V3.1	44
14.1	Logistics Informations.....	44
14.2	Security Properties	45
14.3	AdressAndLengthFormatIdentifier	46
14.4	Expected-Ids	46
14.5	Links to ODX-Diaglayer Container.....	46
14.6	Programming conditions.....	46
15	ECU-MEMORY-Configuration für Bootloader-Datensätze nach V3.1	47
16	Logistic and ECU-Specific für Bootloader-Datensätze nach V3.2	48

16.1	Logistics Informations.....	48
16.2	AdressAndLengthFormatIdentifier	49
17	ECU-MEMORY-Configuration für Bootloader-Datensätze nach V3.2	50
18	Logistic and ECU-Specific für Applikations-Datensätze nach V3.1	51
18.1	Logistics Informations.....	51
18.2	Data	52
18.3	Editor-Funktion	52
19	Logistic and ECU-Specific für Applikations-Datensätze nach V3.2	54
19.1	Logistics Informations.....	54
20	Erstellen eines ASN1-Parser Flashcontainers und Löscontainers.....	55
20.1	Erstellung des ANS1-Parser Flashcontainers mit ODXCreate	55
20.2	Erstellen eines Löscontainers, zum Löschen des ASN1-Parser Zertifikats	56
20.2.1	Allgemein	56
20.2.2	Löschen des ASN1-Parser Zertifikats bei ECUMEM 3.0.0 Steuergerät ohne Löschtreiber	56
20.2.3	Löschen des ASN1-Parser Zertifikats bei ECUMEM 3.0.0 Steuergerät mit Löschtreiber	57
20.2.4	Löschen des ASN1-Parser Zertifikats ab ECUMEM 3.1.0 Steuergerät	57
21	Reengineering von ODXCreate Konfigurationen aus Datencontainern.....	59
21.1	Anleitung für das Reengineering	59
22	Konvertierung von Basiskonfigurationen	60
22.1	Anleitung für das Konvertieren	60
23	Umwandlung von ODX-Flashcontainer in PDX-Flashcontainer und umgekehrt.....	61
23.1	Anleitung für die Umwandlung von Flashcontainer	61
24	Die ocnf-Datei	62
25	Die Kommandozeilensteuerung.....	63
26	Änderungshistorie	64
26.1	Änderungen zur Version 2.....	64
27	Weitere Dokumente, Beschreibungen und Ergänzungen	65
28	Lizenzinformation über ODXCreate-Komponenten.....	66
28.1	Kcontrols	66
28.2	NativeXML.....	66
28.3	Jedi	66
28.4	TurboPower Abbrevia.....	66
28.5	Erläuterte Beispielkonfiguration mit Rückflashschutzblock (ECUMEM-4.3)	67
28.5.1	Erstellen eines neuen Flashcontainers	67
28.5.2	Auswahl der allgemeinen Informationen des Flashcontainers	67
28.5.3	Anlegen eines Rückflashschutzblocks	68
28.5.4	Konfiguration des Rückflashschutztreiberblocks.....	69
28.5.5	Konfiguration des ersten Flashdatenblocks	70
28.5.6	Erstellen des Flashcontainers – PDX	71
28.5.7	Inhalt des Flashcontainers.....	72

1 Vorbemerkung

Einzelne Kapitel enthalten Screenshots und teilweise Beschreibungen die nicht der aktuellen Version entsprechen. Im Zweifel zunächst Kapitel „Erläuterte Beispielkonfiguration mit Rückflasheschutzblock (ECUMEM-4.3)“ oder die jeweilige ECUMEM-SPEC konsultieren.

Bezüglich bekannte Funktionseinschränkungen bitte die aktuellen Release-Notes berücksichtigen.

2 Die Funktionen des ODXCreate

Das Tool ODXCreate ist ein lizenzkostenfreies Tool, um unterschiedliche Arten von Software-Containern erstellen zu können, die für ein Steuergeräte Software-Update notwendig sind:

- ODX-Flashcontainer sowohl nach ECUMEM Version 2.0 und Version 3.0., 3.1 und 4.0 erstellt werden.
- Erstellen ASN1-Parser Flashcontainer (temporäre Umschaltung von Serien- auf Entwicklungs-Signaturschlüssel).
- Erstellen ASN1-Parser Löschcontainer (löschen von ASN1-Parser Zertifikat im Steuergerät, damit wieder Umschaltung auf Serien-Signaturschlüssel).
- Bootloader-Datensätzen nach der Version 3.1 und 3.2 des Datensatzdownload Lastenheftes (ODX-Container auf Basis ECUMEM Version 3.0 und 3.1).
- Applikations-Datensätze nach der Version 3.1 und 3.2 des Datensatzdownload Lastenheftes.
- Unterstützung beim Reengineering von ODXCreate Konfigurationen aus Datencontainern.
- Konvertierung von Basiskonfigurationen (z.B. Umwandlung der ECU-MEMSpec Version).
- Umwandlung von ODX-Flashcontainer in PDX-Flashcontainer und umgekehrt (Prototype).

Für die Nutzung des ODXCreate ist ein Lizenz-File notwendig, das nicht Hardware gebunden ist und daher auch beliebig oft kopiert und auf beliebig vielen Rechnern genutzt werden kann. Dieses Lizenz-File ist kostenlos, aber nur immer für eine bestimmte ODXCreate-Version und eine bestimmte Zeitspanne gültig. Damit wird sichergestellt, dass in regelmäßigen Abständen auf die aktuelle ODXCreate-Version aktualisiert werden muss (neue ODXCreate-Version mit neuem Lizenz-File). Diese Anleitung beschreibt die unterschiedlichen Eingabefelder des ODXCreate und stellt nicht eine Anleitung zum Erstellen von Containern dar. Dabei wurde die Anleitung nach den Eingabereitern des ODXCreate ausgerichtet.

1 Wichtige Anwendungshinweise

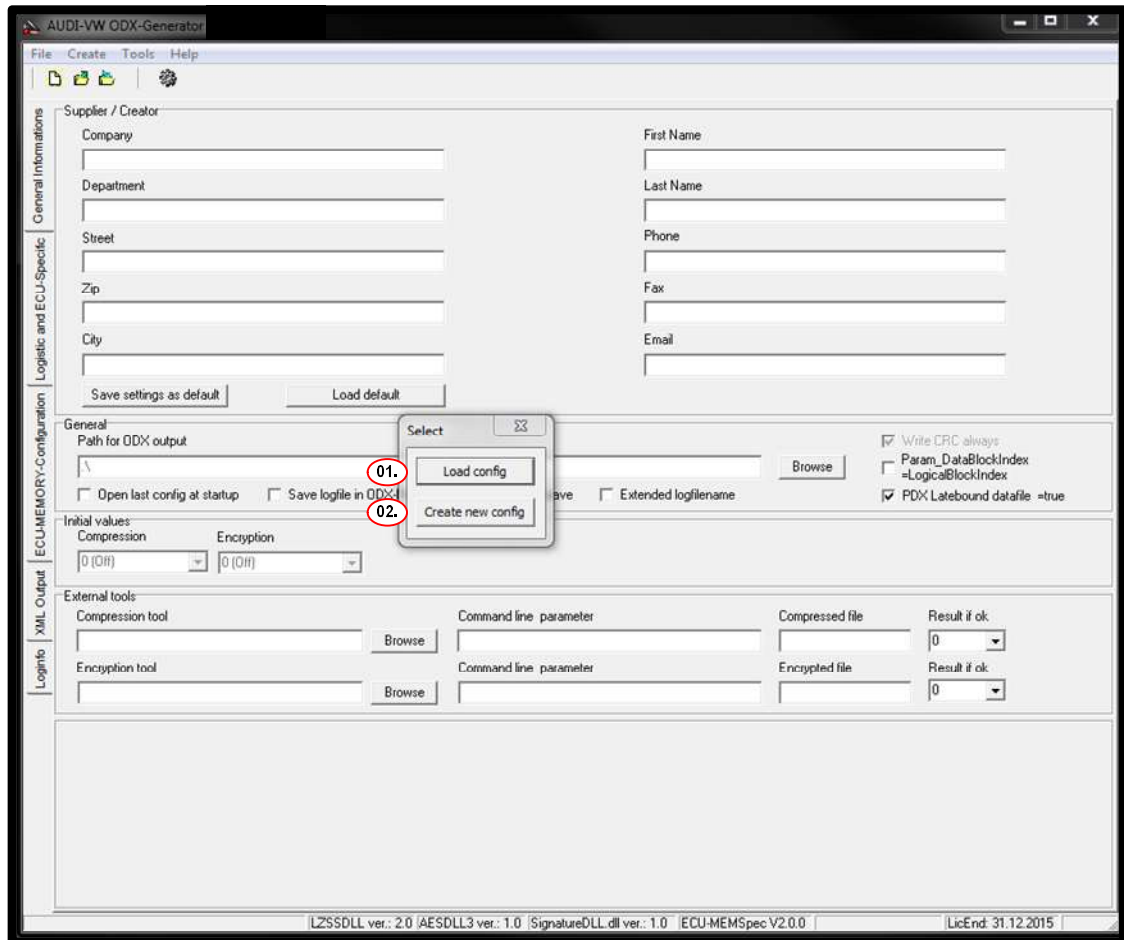
Mit neuen Versionen von ODXCreate werden Verbesserungen und Erweiterungen des Containerformats „OCNF“ erreicht. Aufgrund neuer Funktionalitäten oder Problembehebungen kann daher nicht sichergestellt werden, dass eine Rückwärtskompatibilität zu OCNF-Konfigurationen, welche mit älteren Versionen von ODXCreate erstellt wurden.

Die Konvertierung und Reverse-Funktionalität unterliegt hier bestimmten Einschränkungen und ist eher als Hilfe denn als Vollwertige Konvertierfunktion zu betrachten. Es wird z.B: keine Entschlüsselung oder Dekompression durchgeführt.

2 Starten des ODXCreate

Beim ersten Aufruf der Datei „ODXCreate.exe“ erscheint zunächst das „Select“ Auswahlfenster.

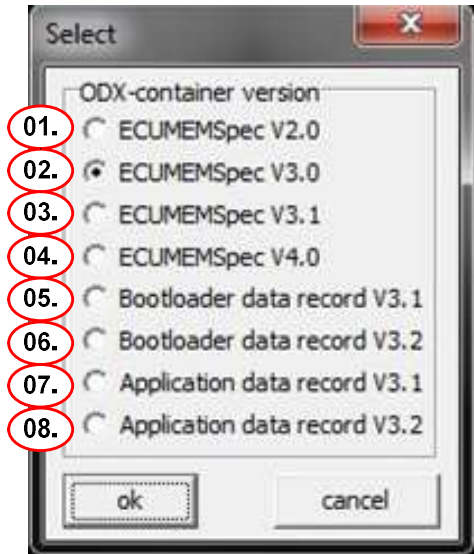
2.1 Select



01. „Load config“ Button – Es kann eine bereits erstellte ODXCreate-Konfiguration (Datei mit der Endung *.ocnf) geladen werden
02. „Create new config“ Button – Es wird ein weiteres Auswahlfenster „Select“ geöffnet für die Festlegung welche Art von Basiskonfiguration erstellt werden soll.

2.2 Festlegung der Basiskonfiguration

Immer wenn eine neue ODXCreate-Konfiguration erstellt wird, muss zunächst festgelegt werden, um welche Art von erzeugten Datencontainern es sich zukünftig handeln soll (Basiskonfiguration). Die festgelegte Entscheidung kann im späteren Prozess nicht mehr revidiert werden (eine bereits erstellte Konfiguration kann nur genutzt werden, um wieder die gleiche Art von Container zu erstellen). Wenn also diese Basiseinstellung falsch vorgenommen wurde, so muss eine neue Konfiguration erstellt werden oder diese Basiskonfiguration muss konvertiert werden. Es ist aber zu beachten, dass eine Konvertierung nur mit Einschränkungen möglich ist.

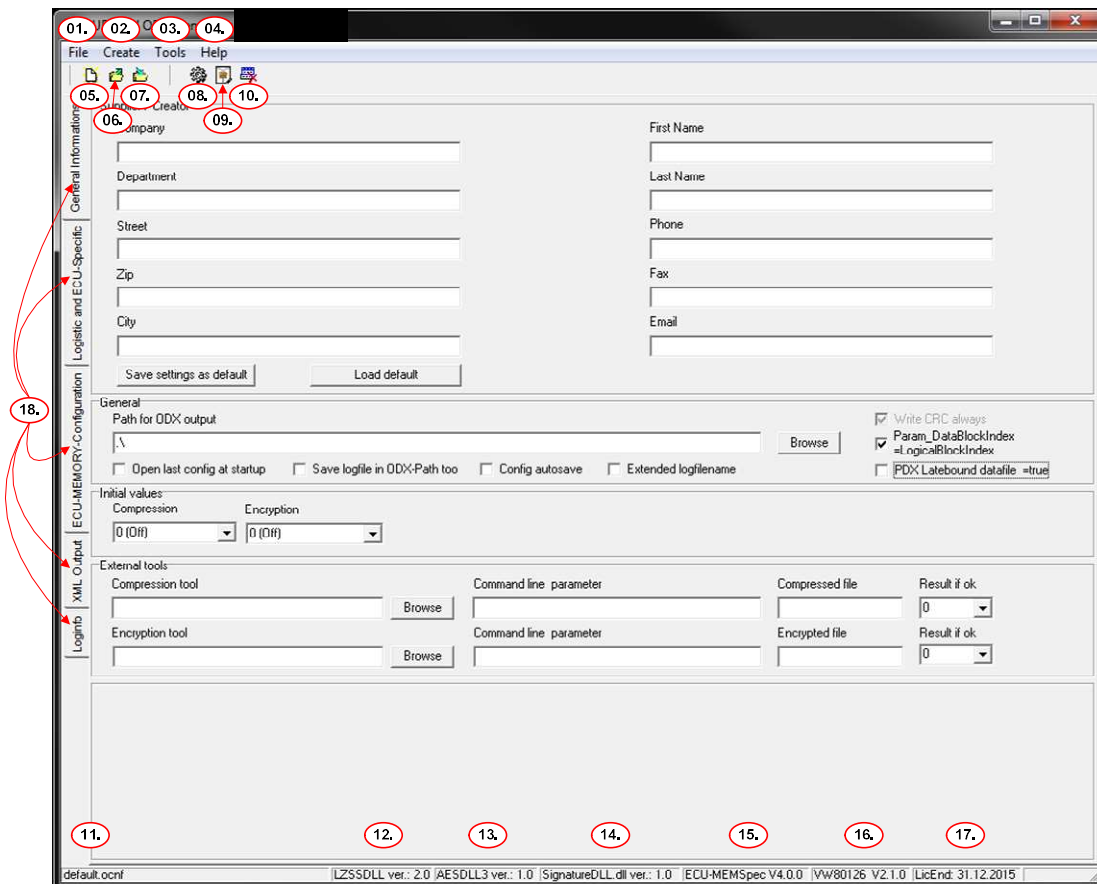


01. „ECUMEMSpec V2.0“ Auswahl – Erzeugt einen Flashcontainer für Steuergeräte, nach der ECUMEM Version 2.0, die nach dem Flashprozesse bis zur VW80126 V1.2 Gültigkeit besitzt.
02. „ECUMEMSpec V3.0“ Auswahl – Erzeugt einen Flashcontainer nach der ECUMEM Version 3.0 (bzw. VW80128 Teil 3) für Steuergeräte, die nach dem Flashprozesse der VW80126 ab V2.0 (MLBevo) umgesetzt sind.
03. „ECUMEMSpec V3.1“ Auswahl – Erzeugt einen Flashcontainer nach der ECUMEM Version 3.1 (bzw. VW80128 Teil 3 V3.1) für Steuergeräte, die nach dem Flashprozesse der VW80126 ab V2.1 umgesetzt sind.
04. „ECUMEMSpec V4.0“ Auswahl – Erzeugt einen Prototypischen Flashcontainer in Vorbereitung der ECUMEMVersion 4.0. Einführung des Flashcontainers im PDX-Format.
05. „Bootloader data record V3.1“ Auswahl – Erzeugt einen Bootloader-Datensatz nach dem Lastenheft Datensatzdownload Generation 2 Version 3.1.
06. „Bootloader data record V3.2“ Auswahl – Erzeugt einen Bootloader-Datensatz nach dem Lastenheft Datensatzdownload Generation 2 Version 3.2.
07. „Application data record V3.1“ Auswahl – Erzeugt einen Applikations-Datensatz nach dem Lastenheft Datensatzdownload Generation 2 Version 3.1.
08. „Application data record V3.2“ Auswahl – Erzeugt einen Applikations-Datensatz nach dem Lastenheft Datensatzdownload Generation 2 Version 3.2.

Da es keine 1 zu 1 Umwandlung zwischen den unterschiedlichen Basiskonfigurationen gibt, kann z.B. eine einmal getroffene Auswahl „ECUMEMSpec V2.0“ nicht in eine „ECUMEMSpec V3.0“ überführt werden. Es gibt aber nun erstmals die Option, dass unter bestimmten Einschränkungen Basiskonfigurationen konvertiert werden können (ECUMEM 3.0 <-> ECUMEM 3.1 <-> ECUMEM 4.0; .Bootloader-Datensätze V3.1 <-> V3.2 und Applikations-Datensätze V3.1 <-> V3.2). Siehe dazu Kapitel 3.1 Menüleiste „Tools“.

3 Allgemeine Elemente des ODXCreate

Die Allgemeinen Elemente des ODXCreate geben entweder Informationen zu der Version des ODXCreate wieder oder dienen zur Steuerung des ODXCreate. Es ist zu beachten, dass manche Elemente nur in Abhängigkeit der gewählten Basiskonfiguration zur Verfügung stehen.



3.1 Menüleiste

01. "File" Menüleiste – Unter diesen Menüpunkt gibt es die Untermenüpunkte:
- „Open config“ – Laden einer bereits existierenden Konfiguration.
 - „Save config“ – Speichern der aktuell bearbeiteten Konfiguration.
 - „Save config as...“ – Speichern der aktuell bearbeiteten Konfiguration unter einem neuen Namen.
 - „New config“ – Erstellen einer neuen Basiskonfiguration.
 - „Import FDS-project“ – Import eines FDS-Konfigurationsfile (FDS = FlashDatenSicherheit im XML-Format), das aus system42 für FDS-SG extrahiert werden kann.
 - „Import programming conditions“ – Hier kann über eine Texttabelle (txt-Format) weitere Programmierbedingungen für die Konfiguration integriert werden.

- „Exit“ – ODXCreate beenden.
02. „Create“ – Je nach der gewählten Basis-Konfiguration gibt es hier die Unterschiedlichen Auswahlmöglichkeiten zum Erzeugen von Daten-Containern in speziellen Ausgabeformaten.
- „ODX/XML“ - Standard in jeder Basiskonfiguration.
- „PDX“ - Nur in der Basiskonfiguration ECUMEMSpec 4.0.
- „ERASE-ASN1“ - (Löschzertifikat) nur in den Basiskonfigurationen 3.1 und 4.0.
Dazu wird die aktuelle Konfiguration verwendet (inklusive seiner Bedatung) und es werden alle logischen Blöcke entfernt, mit Ausnahme der Lösch-Treiberblöcke. Für den logischen Block 0xFE (in diesen Block ist das ASN1-Parser Zertifikat im Steuergerät abgelegt) wird ein Block vom Type „ERASE_ASN1“ erzeugt.
03. „Tools“ – Bietet Optionen zum Bearbeiten/Umwandeln von Containern:
- „Convert config to“ Hier kann eine Basiskonfiguration (je nach geladener Basiskonfiguration) in eine andere Basiskonfiguration umgewandelt werden.
ECUMEM 3.0 <-> ECUMEM 3.1 <-> ECUMEM 4.0
Bootloader data record V3.1 <-> V3.2
Application data record V3.1 <-> V3.2
- „VW80126 version“ Bietet die Option, die Version der VW80126 Version in den Flashcontainern einzutragen (Standard ist die Version 2.0.0)
- „Import config from“ – Reengineering einer Konfiguration aus einem gültigen Datencontainer der mit dem ODXCreate erzeugt wurde.
Dabei gibt es die Unterauswahl:
„ODX-container“ – Flashcontainer/Bootloader-Datensatz (Quelle ist dafür eine *.odx bzw. *.odx-f datei).
„XML-container“ - Applications-Datensatz (Quelle ist dafür eine XML Datei)
Es muss dabei beachtet werden, dass es für das Reengineering Grenzen gibt und daher eine 100 % Reengineering nicht möglich ist. Die Rückgewinnung von Binär-Daten wird nicht unterstützt.
- „Convert container“ – Umwandlung von ODX-Formaten
„PDX -> ODX“ – Konvertieren von PDX-Flashcontainer in ODX-Flashcontainer
„ODX -> PDX“ – Konvertieren von ODX-Flashcontainer in PDX-Flashcontainer
04. „Help“ – Hilfen für den ODXCreate mit folgenden Untermenüpunkten:
- „command line“ – Es werden alle unterstützten Parameter aufgelistet, die ODXCreate über den Aufruf der Kommandozeile unterstützt werden (siehe auch „ODXCreateKommandozeile_Vxxx.pdf“).

3.2 Werkzeugleiste

05. Erstellt eine neue Basiskonfiguration (identisch zu „File“ -> „New config“).
06. Lädt eine bereits gespeicherte Konfiguration (identisch zu „File“ -> „Open config“).
07. Speichert die aktuelle Konfiguration (identisch zu „File“ -> „Save config“).
08. Erstellt einen Datencontainer (identisch zu „Create“->„ODX/XML“).
09. Erstellt einen Datencontainer im PDX-Format (identisch zu „Create“ ->„PDX“)
10. Erzeugt einen reinen Löschcontainer für das ASN1-Parser Zertifikat (identisch zu „Create“->„ERASE-ASN1“).
11. Zeigt den Dateinamen der aktuellen Konfiguration (ocnf-Datei) an.
12. Zeigt die vom ODXCreate eingebundene Version des Standard-SW Modules für die Kompression an.
13. Zeigt die vom ODXCreate eingebundene Version des Standard-SW Modules für die Verschlüsselung an.

14. Zeigt die vom ODXCreate eingebundene Version des krypto42 Standard-SW Modules für die Signatur an.
15. Zeigt die Version der ECUMEM (bzw. VW80128) an, nach welchen Vorschriften dieser Container erzeugt wird (ist nur für Flashcontainer relevant, bei Datensätzen gibt es dafür eine feste Vorgabe).
16. Zeigt die Version der VW80126 an, die in den Datencontainer geschrieben wird (ist nur für Flashcontainer relevant, bei Datensätzen gibt es dafür eine feste Vorgabe).
17. Zeigt das Ablaufdatum der Lizenz für den ODXCreate an.
18. Reiterleiste für die Detail Einstellung einzelner Komponenten im ODXCreate mit den Reitern „General Informations“, „Logistic and ECU-Specific“, ECU-MEMORY-Configuration“, „XML-Output“ und „Loginfo“.

4 General Informations

Im Reiter „General Informations“ werden Administrative Einstellungen und Kontaktdaten des Datencontainererstellers festgelegt.

4.1 Supplier/Creator

01. „Company“ – Firmenname des Container-Erstellers.
02. „Department“ – Abteilung des Container-Autors.
03. „Street“ – Adressangabe der Straße des Firmensitzes.
04. „Zip“ – Adressangabe der Postleitzahl des Firmensitzes.
05. „City“ – Adressangabe des Ortes des Firmensitzes.
06. „FirstName“ – Vorname des Container-Autors.
07. „LastName“ – Nachname des Container-Autors.
08. „Phone“ – Firmenrufnummer, unter der, der Container-Autor erreichbar ist.
09. „Fax“ – Firmenfaxnummer.
10. „Email“ – Firmen E-Mail Adresse des Container-Autors.
11. „Save settings as default“ Button – Die Adressangaben (01 bis 10) können als eine

Default-Vorlage gespeichert werden. Dadurch muss ein Anwender die Adressdaten nicht immer wieder beim Erstellen einer neuer Konfiguration eingeben.

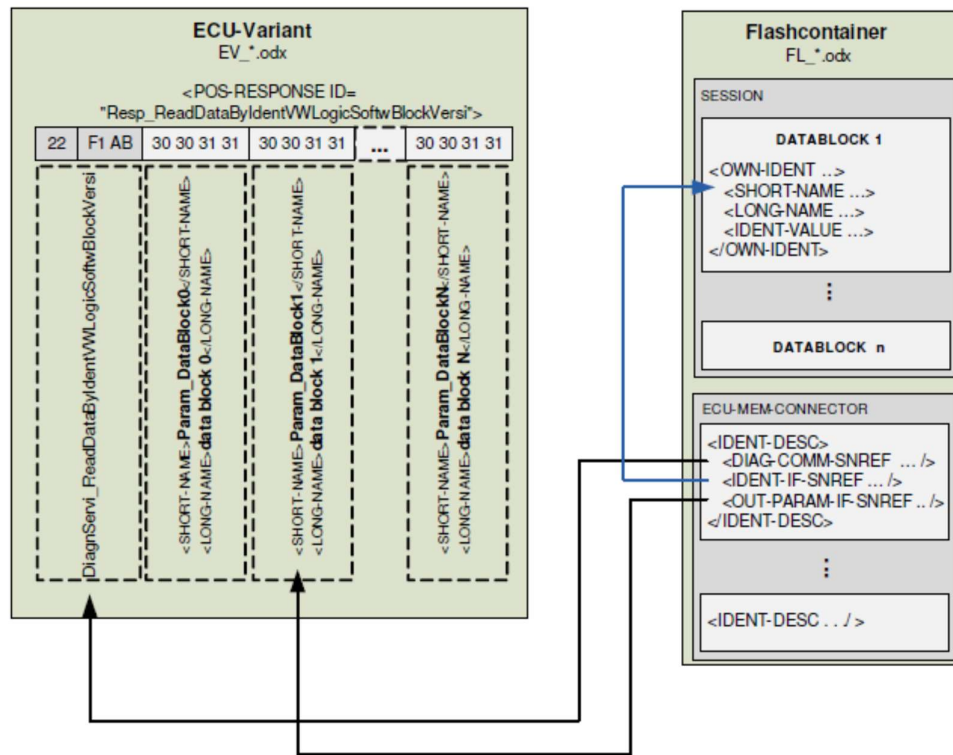
12. „Load default“ Button – Damit können als Default-Vorlage gespeicherte Adressdaten geladen werden (falls eine Default-Vorlage bereits gespeichert wurde).

4.2 General

13. „Path for ODXoutput“ – Festlegung in welches Unterverzeichnis ein neu erstellter Container gespeichert wird. Als Default wird das Unterverzeichnis „Container“ verwendet, das bei einer Installation unter dem Arbeitsverzeichnis erstellt wird.
14. „Browse“ Button – Öffnen des Browserfensters, um über den Explorer das Unterverzeichnis für die Ablage der neu erstellten Container auswählen zu können.
15. „Open last config at startup“ Auswahl – Wenn diese Auswahl aktiviert ist, wird bei jedem Neustart des ODXCreate die zuletzt bearbeitete Konfiguration wieder geladen (das „Select“ Auswahlfenster wird damit nicht mehr angezeigt).
16. „Save logfile in ODXPath too“ Auswahl – Wenn diese Auswahl aktiviert ist, so wird das beim Erstellen eines neuen Containers generierte Logfile nicht nur im Unterverzeichnis „Log“, sondern auch im gleichen Unterverzeichnis abgelegt, in dem der neue Container erstellt wird.
17. „Config autosave“ Auswahl – Durch neuere Features oder Änderungen bei neueren ODXCreate-Versionen kann es notwendig sein, dass auch die ocnf-Datei (die ocnf-Datei ist die gespeicherte Konfigurations-Datei für das Erstellen von Containern) angepasst werden muss (Einfügen neuer Elemente, Umstellung der Struktur etc.). Wenn in einer neuen ODXCreate-Version eine ocnf geladen wird, die ursprünglich mit einer nun „veralteten“ ODXCreate-Version erstellt wurde, so gibt es beim Erzeugen eines neuen Containers eine Warnung (Hinweis, dass an dieser Stelle Ersatzwerte benutzt wurden, da die verwendete ocnf diese Informationen nicht enthält). Diese Warnung wird bei jedem Container Erstellen angezeigt, bis diese „veraltete“ ocnf mit einer aktuellen ODXCreate-Version neu abgespeichert wird (damit werden die Ersatzwerte fest übernommen, wenn diese vom Autor nicht korrigiert wurden). Wenn die Auswahl „Config autosave“ aktiviert ist, so wird vom ODXCreate die ocnf selbstständig im neuesten Format gespeichert, falls ODXCreate eine „veraltete“ ocnf-Datei erkennt. Dadurch werden bei der Containererstellung keine Warnungen mehr generiert, da das ocnf wenn immer nötig aktualisiert wird (und damit die vorgeschlagenen Ersatzwerte fest übernimmt). Dieses Feature kann aber auch den Nachteil haben, dass der Ersatzwert für diesen Container nicht gewünscht ist und man keinen Hinweis erhält, wann Änderungen an der Original ocnf vorgenommen wurden.
18. „Extended logfilename“ Auswahl – Einstellung mit der festgelegt wird, ob das erstellte Logfile „ODXCreate.log“ im Unterverzeichnis „Log“ um den erzeugten Containernamen erweitert wird, inklusive einer zweistelligen laufenden Nummer. Dadurch werden die Logfiles beim Erzeugen eines neuen Containers nicht automatisch wieder überschrieben. Die laufende Nummer wird nur aus den bereits vorhandenen Logfiles des gleichen Namens im Unterordner „Log“ ermittelt.
19. „Write CRC always“ Auswahl – Bei Containern die FDS (FlashDatenSicherheit)

relevant sind (nur für Flashcontainer oder Bootloader-Datensätze), aber keine Kompression nutzen, wurde im Container nur das Element „FW-SIGNATURE“ genutzt und das Element „FW-CHECKSUM“ ist entfallen. Da das Element „FW-CHECKSUM“ immer erlaubt ist und für neue Anforderungen wichtig werden kann (z.B. Absicherung der Partiellen Programmierung über Checksummen Vergleich und nicht nur durch Logischen Software Block Index) wird die Checksumme immer abgelegt (Änderung seit ODXCreate-Version 1.48).

20. „Param_DataBlockIndex=LogicalBlockIndex“ Auswahl – Damit das Feature „Partielle Programmierung“ vom Flashjob genutzt werden kann, muss durch den Flashjob ein Vergleich zwischen den Logistikinformationen im Flashcontainer mit den Logistikdaten, die aus dem Steuergerät stammen, stattfinden. Damit aber die Logistikdaten für jeden einzelnen Datenblock zwischen Steuergerät und Flashcontainer in Beziehung zueinander gebracht werden können, ist im Flashcontainer im Element „ECU-MEMORY-CONNECTOR“ die Beziehung zueinander abgebildet. Dabei wird in dem Element „ECU-MEM-CONNECTOR“ sowohl auf die ECU-Variante (EV) und den dort festgelegten „Param_DataBlock[x] mit dem OWN-IDENT Element des Flashcontainers verwiesen. Da die EV aber unabhängig vom Flashcontainer erstellt wird, kann der Param_DataBlock in dieser EV (Tester-PDX) einen anderen Index haben als der „Logical Block Index“ im Flashcontainer. Aus diesen Grund muss im Flashcontainer dieser Abgleich zu der verwendeten EV möglich sein. Wenn die Auswahl „Param_DataBlockIndex=LogicalBlockIndex“ aktiviert ist, dann wird im Flashcontainer im Element „ECU-MEM-CONNECTOR“ der Param_DataBlockIndex gleich dem LogicalBlockIndex gesetzt, ansonsten muss im Reiter „ECU-MEMORY-Configuration“->“Memory block“ -> 18. für jeden logischen Block auch der Param_DataBlock Index (siehe Kapitel ECU-MEMORY-Configuration) eingetragen werden.



21. „PDX Latebound datafile = true“ Auswahl – Einstellung mit der festgelegt wird, welchen Parameterwert (true/false) in der ODX-F Datei im PDX (ab ECUMEM 4.0.0) für den Parameter PDX Latebound datafile vorgegeben wird. Bei gesetztem Haken wird der Wert = „true“ gesetzt (Default-Einstellung).

4.3 Initial values

Initial values definiert die „Vorschläge“ die beim Anlegen eines neuen logischen Blocks (im Reiter „ECU-MEMORY-Configuration“) als Grundeinstellung verwendet werden sollen. Diese Werte können dort aber jederzeit auch wieder geändert werden. Beim Erstellen eines Containers werden die Werte verwendet, die für jeden logischen Block definiert wurden und nicht die „Initial values“

22. „Compression“ Drop-Down Menü – Einstellung welcher Wert für die Kompression im Flashcontainer stehen soll [0= nicht komprimiert, 1 bis 9 sind durch den Zulieferer definierte Kompressionsverfahren, A bis F sind durch VW/AUDI festgelegt und Standardisierte Kompressionsverfahren).
23. „Encryption“ Drop-Down Menü – Einstellung welcher Wert für die Verschlüsselung im Flashcontainer stellen soll [0= nicht verschlüsselt, 1 bis 9 sind durch den Zulieferer definierte Verschlüsselungsverfahren, A bis F sind durch VW/AUDI festgelegt und Standardisierte Verschlüsselungsverfahren).

4.4 External tools

Bei der Verwendung von Zulieferer eigenen Tools für die Kompression oder Verschlüsselung, kann ODXCreate diese verwenden, wenn diese über einen Kommandozeilenaufruf gesteuert werden können.

24. „Compression tool“ Eingabe – Name des Kompressionsprogramms.
25. „Browse“ Button – Öffnen des Browserfensters, um über den Explorer das Kompressionsprogramm auszuwählen.
26. „Command line parameter“ Eingabe – Übergabe Parameter, wie dieser beim Aufruf über die Kommandozeile des Kompressionstools angesprochen werden muss.
Hier ist zu beachten, dass der ODXCreate ein Binär-File an das Kompressionstool übergibt und als Rückgabewerte auch wieder ein Binär-File zur weiter Verarbeitung erwartet.
27. „Compressed file“ Eingabe – Benennung, wie das Output File heißt, das vom ODXCreate weiter verarbeitet werden muss.
28. „Result if ok“ Festlegung – Wenn das Kompressionstool nach dem Aufruf über einen Rückgabewert verfügt, der angibt, ob die Kompression erfolgreich durchgeführt werden konnte, so kann dieser vom ODXCreate interpretiert werden. Mit diesem Parameter, wird festgelegt welcher Rückgabewert des Kompressionstools als erfolgreich verwendet wird.
29. „Encryption tool“ Eingabe – Name des Verschlüsselungsprogramms.
30. „Browse“ Button – Öffnen des Browserfensters um über den Explorer das Verschlüsselungsprogramm auszuwählen.
31. „Command line parameter“ Eingabe – Übergabeparameter, wie dieser beim Aufruf über die Kommandozeile des Verschlüsselungstools angesprochen werden muss.
Hier ist zu beachten, dass der ODXCreate ein Binär-File an das Verschlüsselungstool übergibt und als Rückgabewerte auch wieder ein Binär-File zur weiter Verarbeitung erwartet.
32. „Compressed file“ Eingabe – Benennung, wie das Output File heißt, das vom ODXCreate weiter verarbeitet werden muss.
33. „Result if ok“ Festlegung – Wenn das Verschlüsselungstool nach dem Aufruf über einen Rückgabewert verfügt, der angibt, ob die Verschlüsselung erfolgreich durchgeführt werden konnte, so kann dieser vom ODXCreate interpretiert werden. Mit diesem Parameter wird festgelegt welcher Rückgabewert des Verschlüsselungstools als erfolgreich verwendet wird.

Für die Verwendung von externer Kompressions- und Verschlüsselungssoftware gibt es noch das Dokument ODXCreateExterneTools_Vx.xx.pdf das hier auch Tipps zur Nutzung gibt.

5 Logistic and ECU-Specific für ECU-MEMSpec V3.0

Im Reiter „Logistic and ECU-Specific“ für ECU-MEMSpec V3.0 werden für Flashcontainer (ab MLBevo) die logistischen Steuerungselemente bedatet.

5.1 Logistics Informations

01. „VwSparePatnumber(F187)“ – Vorgabe der VW/AUDITEilenummer, die zu diesem Flashcontainer gehört.
02. „VwSoftwareVersionNumber(F189)“ – Vorgabe des SW-Index, den dieser Flashcontainer erzeugen soll.
03. „Optional Description“ – Optionales Beschreibungsfeld, mit der die Containerbenamung erweitert werden kann.
04. „Container Version“ – Optionale Containerversionierung (Ab der ECUMEM 3.1 Pflichtfeld).

5.2 Security Properties

05. „Security Access - SA 2“ – Eingabe des SA2-MiniBefehlssatz für die Freischaltung in der Programming Session. Achtung! Der Ende-Befehl „+4C“ wird vom ODXCreate automatisch ergänzt.
06. „Enable FDS“ Auswahl – Festlegung, ob das Steuergerät FlashDatenSicherheit unterstützt (Wegfahrsperren-, Komponentenschutz- und Tachomanipulationschutz-Steuergeräte). Wenn die Auswahl „FDS“ gewählt wurde, dann werden die Eingabefelder 07. bis 12. eingeblendet.

Bitte beachten! Es gibt zwei unterschiedliche Typen von Signaturschlüsseln. Zum einen eine personenbezogene Authentifizierung (FDSProject_xxxx_E.p7), sprich Signatur-Datei mit der Endung *.p7 (dazu ist ein Kartenleser, gültige PKI-Karte und das aktuellste EE-Krypt Tool notwendig. Das funktioniert aber auch nur dann, wenn der PKI-Kartenbesitzer für dieses Projekt freigeschaltet wurde) und die nicht personenbezogene Authentifizierung (FDSProject_xxxx_E.p8) - sprich Signatur-Datei mit Endung *.p8.

Die Signaturen werden mit den P8-Dateien direkt von ODXCreate erzeugt. Mit den P7-Dateien kann die Signatur vom ODXCreate nicht erstellt werden. ODXCreate weist aber mit einer Meldung darauf hin und erzeugt den Container (ohne Signatur), aber mit den notwendigen Erweiterungen, damit dieser Container im Nachgang über das Tool EE-Krypt erzeugt werden kann.

Info: Die Festlegung, ob eine personenbezogene Authentifizierung notwendig ist, wird beim Erstellen des FDS-Projekts in system42 getroffen und kann rückwirkend nicht mehr geändert werden.

07. „Keyfile“ Anzeige – Zeigt an, auf welchen Signaturschlüssel das FDS-Project-XML File verweist.
08. „FDS-Project-ID“ Anzeige – Interpretiert aus dem FDS-Project-XML File, welcher FDS-Project-ID diesem Schlüssel zugeordnet ist.
09. „Signature-Methode“ Anzeige – Interpretiert aus dem FDS-Project-XML File, welche Signature-Methode verwendet wird.
10. „Encryption-ALG“ Anzeige - Interpretiert aus dem FDS-Project-XML File, welcher Verschlüsselungs-Algorithmus verwendet wird.
11. „Hash-ALG“ Anzeige - Interpretiert aus dem FDS-Project-XML File, welcher Hash-Algorithmus verwendet wird.
12. „FDS-project config file“ Anzeige – Zeigt das aktuell geladene FDS-Project-XML File an.
13. „Import FDS project“ Button – Startet den Browser, um das FDS-Project-XML File laden zu können. Das FDS-Project-XML File muss aus system42 heruntergeladen werden und enthält unter anderen den Entwicklungs-Signaturschlüssel. Dadurch werden auch die notwendigen Erweiterungen im Container vorbereitet, damit dieser von krypto42 seriensigniert werden kann.

5.3 AdressAndLengthFormatIdentifier

Eingaben der AdressAndLengthFormatIdentifier (ALFID) für den EraseMemory (0x31 01 FF 00 xx xx) und den RequestDownload (0x34 xx xx xx xx ...) Service:

14. „Len of MemSize (EraseMemory)“ Vorgabe – Der ALFID beim EraseMemory

entspricht im Service 0x31 dem #5 Data Byte. Die Bits 4 bis 7 geben die Länge des Parameters für memorySize an (0x31 01 FF 00 xx xx xx...). Im VW-Konzern wird der Parameter nicht genutzt und ist daher als Default Wert mit 0 vorgegeben.

15. „Len of MemAddr (EraseMemory)“ Vorgabe – Der ALFID des EraseMemory entspricht im Service 0x31 dem #5 Data Byte. Die Bits 0 bis 3 geben die Anzahl der Bytes für die memoryAddress an (0x31 01 FF 00 xx xx xx...). Damit bestimmt x die Anzahl der nachfolgenden xx-Pärchen im Service, z.B. 0x31 01 FF 00 01 xx. Da im VW-Konzern als Adressierung die logische Block Adressierung verwendet wird, reicht meist ein Byte aus.
16. „Len of MemSize (RequestDownload)“ Vorgabe – Der ALFID beim RequestDownload entspricht im Service 0x34 dem #3 Data Byte. Die Bits 4 bis 7 geben die Länge des Parameters für die unkomprimierte MemorySize an (0x34 xx xx xx xx...). Damit bestimmt x die Anzahl der letzten xx-Pärchen im Service, z.B. 0x34 00 x4 xx ... 00 01 00 00.
17. „Len of MemAddr (RequestDownload)“ Vorgabe – Der ALFID beim RequestDownload entspricht im Service 0x34 dem #3 Data Byte. Die Bits 0 bis 3 geben die Anzahl der Bytes für die memoryAddress an (0x34 xx xx xx xx...). Damit bestimmt x die Anzahl der xx-Pärchen im Service unmittelbar nach dem ALFID Parameter, z.B. 0x34 00 2x 00 30 xx ...

Beispiel für Len of MemSize + Len of MemAddr.: 0x34 00 24 00 30 00 01 00 00

18. „Default“ Button für die ALFID – Setzt die ALFIDs für den EraseMemory und RequestDownload auf den VW-Konzern Default-Wert (EraseMemory: Len of MemSize = 0; Len of MemAddr = 1; RequestDownload: Len of MemSize = 4, Len of MemAddr = 1).

5.4 Expected-Idents

Expected-Ident wird in der Entwicklung genutzt, um eine „Vorauswahl“ von Containern zu erhalten, die für den Update Vorgang am Tester angeboten werden (dieses Feature kann bei den Entwicklungstestern auch deaktiviert werden). Diese „Vorfilterung“ wird dabei aber nur vom Tester genutzt und hat keine Auswirkungen auf den Flashjob (damit kein Mittel, um in der Produktion oder beim Kundendienst eine Update Programmierung zu verhindern).

19. „VWSparePartNumber (F187)“ Vorgabe – Festlegung der erlaubten VW/AUDI-Teilenummer
20. „VWSoftwareVersionNumber (F189)“ Vorgabe – Festlegung der erlaubten VWApplicationSoftware-Version
21. „VWECUHardwareNumber (F191)“ Vorgabe – Festlegung der erlaubten Referenz-Hardware
22. „VWECUHardwareVersionNumber (F1A3)“ Vorgabe – Festlegung der erlaubten Hardware-Version
23. „+“ Button Expected-Idents – Mit dem „+“ Button wird das „User Input“ Fenster geöffnet. Dazu muss aber zuvor eines der 4 Elemente von VWSparePart-Number, VWSoftwareVersionNumber, VWECUHardwareNumber oder VWECUHardwareVersionNumber mit der Maus oder Cursor markiert worden sein (Die Markierung darf nicht auf ein Unterelement dieses Elementes zei-

gen, sprich auf einen bereits aufgelisteten Wert stehen). Dadurch können für jedes der 4 Elemente weitere Einträge hinzugefügt werden.

24. „-“ Button Expected-Idents – Mit dem „-“ Button kann ein markierter Eintrag (Unterelement) wieder gelöscht werden.

Die „Filterung“ durch die Expected-Idents im Tester entspricht dabei folgender Logik: Der Container wird im Tester angezeigt, wenn der Ausdruck:

```
[VWSparePartNumber(
  Unterelement_1 oder Unterelement_2 oder ...Unterelement_n)]
und
[VWSoftwareVersionNumber(
  Unterelement_1 oder Unterelement_2 oder ...Unterelement_n)]
und
[VWECUHardwareNumber(
  Unterelement_1 oder Unterelement_2 oder ...Unterelement_n)]
und
[VWECUHardwareVersionNumber(
  Unterelement_1 oder Unterelement_2 oder ...Unterelement_n)]
```

wahr ist. Nur dann wird die Flash-Session im Tester zur Anzeige gebracht (wenn es kein Unterelement für ein Element gibt, so ist dieser Teilausdruck erfüllt).

Beispiel:

Folgende Vorgaben wurden getroffen

VWSparePartNumber: 12345678901 und 23456789012

VWSoftwareVersionNumber: 0030 und 0040

Bei folgenden Steuergeräten würde der so erzeugte Container zum Update im Tester angeboten:

12345678901+0030

12345678901+0040

23456789012+0030

23456789012+0040

5.5 Links to ODX-Diaglayer Container

Mit dem ODX-Diaglayer wird festgelegt, auf welche Steuergeräte der Container geflashed werden kann. Dabei wird über den Logical Link (Einstieg im Tester zu dem gewünschten Steuergerät) verglichen, ob der ASAM-ODX-File-Identifizier im Container zum aktuellen Link passt (aufgelistet ist). Dabei kann es beliebig viele Einträge im ODX-Diaglayer geben (auch Kombinationen von BV, EV und GVs). Der Logical Link den der Tester verwendet, um mit dem Steuergerät zu kommunizieren, muss zwingend enthalten sein (und wenn es nur eine Vererbung aus der BV ist), ansonsten wird der Container im Tester für die Update-Programmierung nicht angezeigt und kann auch nicht upgedatet werden (auch bei Produktion und Kundendienst nicht). Falls nicht mindestens eine BV, EV oder GV eingetragen wurde, dann wird vom ODXCreate kein Container erstellt.

25. „BV_LayerRefs“ Vorgabe – Festlegung der BV-Variante zu der dieser Container kompatibel ist. Bei BV wird nur der ASAM-ODX-File-Identifier eingetragen, wie dieser auch im aktuellen Fahrzeug-PDX des Testers aufgelistet ist. Es ist zu beachten, dass mit dem Eintrag einer BV-Variante dieser Container auch auf allen Steuergeräten für die Update-Programmierung angeboten wird, die über Vererbung von dieser BV erreichbar sind. Bei Steuergerätegruppen (z.B. Steuergeräte mit mehreren unterschiedlichen Lieferanten) kann daher ein BV-Eintrag auch dazu führen, dass Software auf nicht kompatible Hardware zum Update angeboten wird. Im Gegenzug gibt es aber keine Abhängigkeit zu Hauptvarianten, die dann dazu führen, dass ein Container nicht mehr auf ein Steuergerät geflashed werden kann, da sich die Hauptvariante geändert hat. Die BV Einträge enthalten keine Hauptvariante.
26. „EV_LayerRefs“ Vorgabe – Festlegung der EV Variante, zu der dieser Container kompatibel ist. Bei der EV wird der ASAM ODX File Identifier eingetragen plus der Hauptvariante des ASAM ODX File Version (die ersten 3 Zeichen der ASAM ODX File Version). Die **Hauptvariante** wird mit **Unterstrich** an den **ASAM ODX File Identifier** angehängt, z.B. **EV_ECM125TDI150KW_001**. Wie bereits bei der BV Variante aufgeführt, kann mit der EV gezielt gesteuert werden (wenn es hier besondere Abhängigkeiten gibt) auf welche EV Variante der Container geflashed werden kann. Dabei ist aber zu beachten, dass wenn sich die Hauptvariante der EV ändert (z.B. Inkompatibilität der ODX-Daten, die aber für das Updateverhalten nicht relevant sind) der Container dadurch auf dieses Steuergerät nicht mehr geflashed werden kann. Das Feature Partielle Programmierung kann auch nur genutzt werden, wenn ein vorhandener EV_Layer Eintrag im Flashcontainer zu dem zu flashenden Steuergerät gehört. Wenn im ODXCreate (Konfiguration) keine EV-Variante eingetragen wird, so wird auch das Feature „Partielle Programmierung“ beim Erstellen eines Flashcontainers deaktiviert.
27. „GV_LayerRefs“ Vorgabe – Festlegung der GV-Variante (Gruppenvariante) zu der dieser Container kompatibel ist. Die GV ist dabei „nur“ eine spezielle Version der „EV“. Diese Variante ist z.B. für das Konzept bei Motor-SGten gedacht. Es gibt mehrere Motor-SGte (z.B. von einem Lieferanten), die man zusammenfassen möchte, um die Abhängigkeit der einzelnen aufgelisteten EVs zu reduzieren, ohne gleich eine BV eintragen zu müssen, die dann dazu führt, dass z.B. ein Container auch Lieferanten übergreifend verwendet werden kann, was dann unweigerlich zu einem Flashabbruch führt.
28. „+“ Button Links to ODX-Diaglayer Container – Mit dem „+“ Button wird das „User Input“ Fenster geöffnet. Dazu muss aber zuvor eines der 3 Elemente von BV_LayerRefs, EV_LayerRefs, GV_LayerRefs mit der Maus oder Cursor markiert worden sein (es darf die Markierung nicht auf einem Unterelement dieses Elementes zeigen, sprich auf einen bereits aufgelisteten Wert stehen). Dadurch können für jedes der 3 Elemente weitere Einträge hinzugefügt werden.
29. „-“ Button Links to ODX-Diaglayer Container – Mit dem „-“ Button kann ein markierter Eintrag (Unterelement) wieder gelöscht werden.

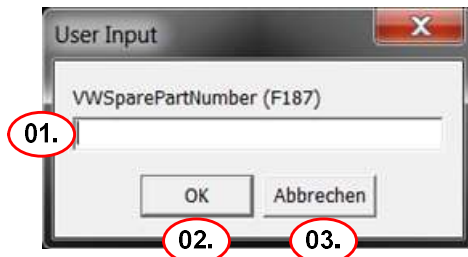
5.6 Programming conditions

Im Fenster Programming conditions werden die möglichen Programmierbedingungen angezeigt. Die Liste der Programmierbedingungen kann jederzeit erweitert werden.

Dazu muss nur die Datei `odxcreate.ini` (befindet sich im gleichen Verzeichnis wie die `odxcreate.exe`) mit einem Texteditor geöffnet werden, neue Einträge hinzugefügt werden und anschließend die Änderungen gespeichert werden.

30. „Auswahl der Programmierbedingungen“ – Durch selektieren (an hacken) aus der Liste der Programmierbedingungen können die Steuergeräte individuellen Programmierbedingungen zusammengestellt werden, die dann auch im Container abgelegt werden (dieses Feature ist nur eine Dokumentation, die tatsächliche Umsetzung der Programmierbedingungen muss in der Applikations-Software erfolgen).
31. „<TOOL>User tool<\TOOL> optional“ Eingabefeld – Optionales Eingabefeld für den Namen und Version eines Zulieferer-Tools, das zur Generierung (Steuerung des ODXCreate) von Daten-Container mit verwendet wird. Der Eintrag wird im ODX-Container unter dem Tag <TOOL> bei der Version des ODXCreate erweitert.

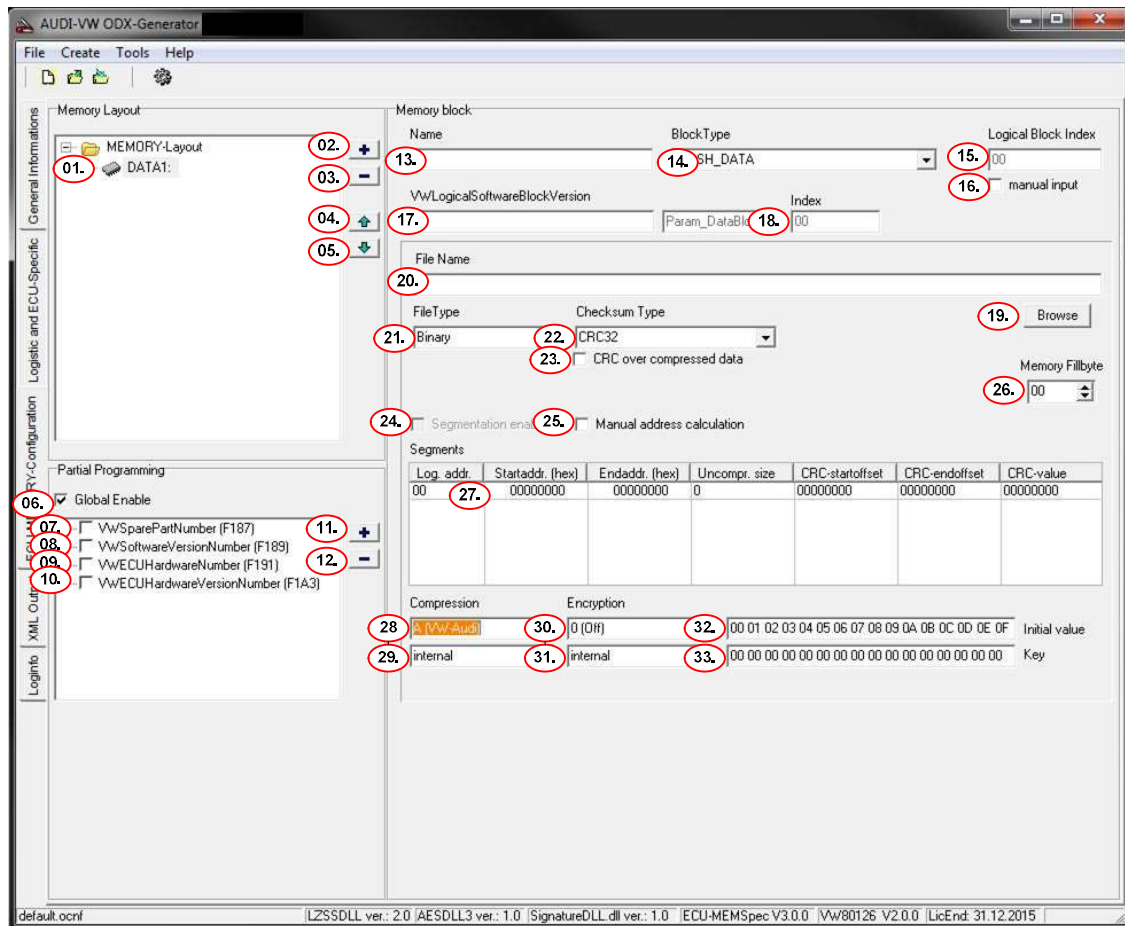
5.7 User Input Fenster



01. „VWSparePartNumber (F187)“ Eingabefeld – Eingabe eines weiteren Eintrages, hier für das gewählte Expected-Ident Element, VWSparePartNumber (F187). Es gibt für jedes Expected-Idents Element (VWSparePartNumber, VWSoftwareVersionNumber, VWECUHardwareNumber und VWECUHardwareVersionNumber) ein eigenes User Input Fenster, die aber hier nicht weiter aufgelistet werden.
02. „OK“ Button – Bestätigen des neuen Eintrages und Aufnahme in die Liste, mit schließen des User Input Fensters.
03. „Abbrechen“ Button – Verwerfen des Eintrages und schließen des User Input Fensters

6 ECU-MEMORY-Configuration für ECU-MEMSpec V3.0

Im ECU-MEMORY-Configuration-Reiter werden der Speicheraufbau, Flashablauf, Vorgaben für die Partielle Programmierung und die zu programmierenden Binärdaten festgelegt.



6.1 Memory-Layout

Im Memory Layout wird das Speicher-Layout erstellt und grafisch angezeigt.

01. „Memory-Layout“ Darstellung – zeigt den grafischen Aufbau der logischen Blöcke in einer Baumstruktur und dient zur Selektion eines logischen Blockes, um dessen Einstellungen im „Memory Block“ zu erstellen bzw. zu ändern. Der „Memory Block“ zeigt dabei immer die Einstellungen des aktuell markierten logischen Blockes.
02. „+“ Button Memory-Layout – Hinzufügen eines weiteren logischen Blockes in der Grafik mit öffnen des „Add new mem block“ - Drop-Down Menüs.
03. „-“ Button Memory-Layout – Löschen des markierten logischen Blockes.

Achtung damit gehen auch alle zugewiesenen „Memory block“ Einstellungen verloren!

- 04. „Pfeil oben“ – Den markierten logischen Block um eine Position nach oben schieben.
- 05. „Pfeil unten“ – Den markierten logischen Block um eine Position nach unten schieben.

6.2 Partial Programming

Im Partial Programming Fenster wird festgelegt, ob das Steuergerät das partielle Flashen unterstützt und wenn ja mit welchen Einschränkungen.

- 06. „Global Enable“ Auswahl – Festlegung, ob das Steuergerät das partielle Flashen unterstützt. Wenn Global Enable aktiviert wurde, werden die Felder 17. bis 18 angezeigt und die Eingabe für die Felder 07. bis 12. wird freigegeben. Wenn in dem Reiter „Logistic and ECU-Specific“ nicht mindestens eine EV-Varinate in die „EV_LayerRefs“ eingetragen wurde, ist dieser Button ausgegraut und kann nicht markiert werden (ohne einer EV ist das Feature „Partial Programming“ technisch nicht möglich).
- 07. VWSparePartNumber (F187) Vorgabe – Festlegung der erlaubten VW/AUDI-Teilenummer.
- 08. VWSoftwareVersionNumber (F189) Vorgabe – Festlegung der erlaubten VWApplicationSoftware-Version.
- 09. VWECUHardwareNumber (F191) Vorgabe – Festlegung der erlaubten Referenz-Hardware.
- 10. VWECUHardwareVersionNumber (F1A3) – Festlegung der erlaubten Hardware-Version.
- 11. „+“ Button Expected-Idents – Mit dem „+“ Button wird das „User Input“ Fenster geöffnet. Dazu muss aber zuvor eines der 4 Elemente von VWSparePartNumber, VWSoftwareVersionNumber, VWECUHardwareNumber oder VWECUHardwareVersionNumber mit der Maus oder dem Cursor markiert worden sein (Die Markierung darf nicht auf ein Unterelement dieses Elementes zeigen, sprich auf einem bereits aufgelisteten Wert stehen). Dadurch können für jedes der 4 Elemente weitere Einträge hinzugefügt werden.
- 12. „-“ Button Expected-Idents – Mit dem „-“ Button kann ein markierter Eintrag (Unterelement) wieder gelöscht werden.

Die Freischaltung des Features „Partielle Programmierung“ im Flashjob entspricht der Logik der Filterung bei den Expected-Idents. Da es aber für die Partielle Programmierung, im Regelfall keine Einschränkung gibt, sollte es auch keine Unterelemente für VWSparePartNumber, VWSoftwareVersionNumber, VWECUHardwareNumber oder VWECUHardwareVersionNumber geben.

6.3 Memory block

Im Memory block werden für jeden logischen Block die Eigenschaften festgelegt. Dabei werden die Eigenschaften des im MEMORY-Layout Fenster markierten logischen Block dargestellt.

- 13. „Name“ Eingabe – Dient zur Bezeichnung eines logischen Blockes
- 14. „BlockType“ Vorgabe – Mit dem BlockType wird der Flashablauf im Flashjob

gesteuert. Dazu gibt es 4 vorgegebene Arten von BlockType (siehe auch VW80128 Teil 3, „DATABLOCK“): ERASE_PROGRAMMING_ROUTINE, ERASE_ROUTINE, PROGRAMMING_ROUTINE und FLASH_DATA.

Die verschiedenen Typen:

ERASE_PROGRAMMING_ROUTINE:

enthält den kompletten Flashtreiber (Löschen und Programmieren) für das Steuergerät

ERASE_ROUTINE:

enthält den reinen Löschtreiber für das Steuergerät

PROGRAMMING_ROUTINE:

enthält den reinen Flashtreiber (ohne Löschtreiber) für das Steuergerät

FLASH_DATA: Default

ASN1_DATA:

Pseudo-BlockType der im Flashcontainer als „FLASH_DATA“ eingetragen wird. Dieser Pseudo-Block Type wurde eingeführt um die ASN1_Parser Zertifikat Erstellung zu vereinfachen. Dieses ASN1_Parser Zertifikat ist ein spezieller Flashcontainer, der keine Applikations-Software Blöcke enthalten darf, sondern nur ein aus krypto42 erzeugtes File (Dateiendung *.der).

Für den Type PROGRAMMING_ROUTINE und FLASH_DATA leitet das Programmiergerät die Download Sequenz beginnend mit einem EraseMemory ein.

Für den Type ERASE_ROUTINE und ERASE_PROGRAMMING_ROUTINE leitet das Programmiergerät die Download-Sequenz mit einem RequestDownload ein. Ein EraseMemory wird für diese beiden Typen nicht gesendet.

15. „Logical Block Index“ Eingabe – Eingabe der logischen Block Adresse. Wenn keine manuelle Eingabe erfolgt, so vergibt ODXCreate die logische Block Adresse selbstständig (beginnend bei 01) in dem jeder neu hinzugefügte logische Block eine um 1 höhere logische Block Adresse erhält.
16. „manual input“ Auswahl – Damit kann die logische Block Adresse manuell Vorgegeben werden. Wenn „manual input“ aktiviert wird, kann das Feld 15. überschrieben werden.
17. „VWLogicalSoftwareBlockVersion“ Eingabe – Feld ist nur sichtbar, wenn 6. Aktiviert ist und dient zur Eingabe der VWLogicalSoftwareBlockVersion. Die VWLogicalSoftwareBlockVersion wird beim partiellen Flashen verwendet, um zu entscheiden, ob dieser Logische Block upgedatet werden muss oder nicht. Dabei führt der Flashjob einen String-Vergleich zwischen der aus dem Steuergerät ausgelesenen VWLogicalSoftwareBlockVersion und der im Container hinterlegten VWLogicalSoftwareBlockVersion durch. Nur wenn beide Versionen übereinstimmen, wird im Flashjob-Ablauf dieser logische Block „übersprungen“.
18. „Index“ Eingabefeld für den Param_Data – Eingabe des Param_Data, siehe dazu auch Reiter „General Information“->„General“->19. inklusive der Erklärung. Das Feld 18. „Index“ kann nur überschrieben werden, wenn in Reiter „General Information“->„General“->19. Das Feld „Param_DataBlockIndex=LogicalBlockIndex“ nicht aktiviert ist.
19. „Browse“ Button – Öffnet den Explorer um das Datenfile hochladen zu können, das

in diesen logischen Block geschrieben werden soll.

20. „File Name“ Ausgabefeld – zeigt an welches Datenfile für diesen logischen Block geladen wurde.
21. „File Type“ Auswahl – Einstellung, von welchem Typ das Datenfile ist. Dabei wird die Voreinstellung auch bei der „Browse“ Datenfile Suche berücksichtigt, so dass nur noch Datenfiles dieses Typs im Browser angezeigt werden.
22. „Checksum Type“ Auswahl – Hier können verschiedene Checksummen Berechnungsalgorithmen ausgewählt werden. Dabei berechnet ODXCreate über das hochgeladene Datenfile die Checksumme nach dem ausgewählten Checksum Type und trägt die Checksumme in den Container ein. Der Checksum Type steuert auch die Auswahlmöglichkeiten in dem Zusatzfenster „segment“, siehe 6.4.
23. „CRC over compressed data“ Auswahl – Wenn bei der Kompression ein Wert gleich 0 gewählt wurde, so ist diese Eingabe ausgeblendet. Bei der Vorgabe eines Kompressionswert ungleich „0“ (der Bootloader erwartet komprimierte Daten) kann mit dieser Auswahl festgelegt werden, ob die Checksumme über die Rohdaten (vor der Kompression) oder über die komprimierten Daten ermittelt werden soll. Die Festlegung für den VW-Konzern ist die Checksumme über die Rohdaten zu berechnen (VW-Konzern Standard Bootloader kann nur die Checksumme über die Rohdaten als i.O. interpretieren).
24. „Segmentation enabled“ Auswahl – Vorhalt für die Zukunft, wenn die Segmentierung von logischen Blöcken im VW-Konzern genutzt werden kann.
25. „Manual address calculation“ Auswahl – ODXCreate ermittelt vor jeder Containererstellung automatisch die Start-, Endadresse, Größe, Checksumme etc..., wenn das aber nicht gewünscht ist, da z.B. aus einem großen Binärfile nur Teile verwendet werden sollen, oder die geladenen Binärfiles kleiner sind als die tatsächliche logische Blockgröße etc. dann gibt es die Option, diese Rahmengrößen manuell festzulegen. Dazu muss die Auswahl „Manual address calculation“ aktiviert sein. Die Festlegung der Rahmengrößen wird später noch in 26. und in „segment“ erklärt.

Wichtig! Wenn die Auswahl „Manual address calculation“ aktiviert worden ist, dann werden diese Vorgaben auch hart verwendet, ohne dass diese Werte beim Einlesen oder Ändern des Source-Datenfiles noch einmal angepasst oder vom ODXCreate geprüft werden.

26. „Memory Fillbyte“ Vorgabe – Festlegung mit welchen Füllbytes (im Hexbereich von 0x00 bis 0xFF) ODXCreate Lücken im Datenstrom auffüllen soll, wenn z.B. bei aktiver „Manual address calculation“ ein größerer Adressbereich vorgegeben wurde, als durch das Datenfile abgedeckt wird.
27. „Segments“ Übersicht – Zeigt die vom ODXCreate ermittelten Rahmengrößen jedes Segments (bisher wird im VW-Konzern nur ein Segment pro logischen Block unterstützt) an, die durch das eingelesene Datafile ermittelt werden. Wie Startadresse, Endadresse, unkomprimierte Größe, Checksummen Start-Offset, Checksummen End-Offset und die Checksumme. Durch einen Doppelklick mit der Maus auf die Segmentzeile wird das Fenster „segment“ geöffnet, mit dem die Werte noch korrigiert werden können.
28. „Compression“+„Methode“ Auswahl – Festlegung ob der Bootloader komprimierte oder unkomprimierte Daten erwartet. Bei der Vorgabe „0“ werden vom Bootloader unkomprimierte Daten erwartet (Feld 23. wird ausgeblendet und ist nicht sichtbar). Ansonsten erwartet der Bootloader komprimierte Daten. Die Auswahl 1 bis 9 sind dabei für Zulieferer eigene Kompressionsverfahren re-

serviert. Die Zahl wird in den Container übernommen und im Flashjob bei dem Service RequestDownload (0x34) im dataFormatIdentifier bei compression-Method gesendet. Die Auswahl „A“ entspricht dabei der vom VW-Konzern im Standard-SW Bootloader integrierten Kompressionsmethode (in zwei Ausführung mit oder ohne Padding-Bytes). ODXCreate kann diese Kompression ohne weiteres externes Tool selbstständig durchführen.

Hinweis: Sollte durch die Kompression, die Größe des Binärfiles für diesen logischen Block größer werden als das Original Binärfile, so gibt der ODXCreate eine Warnung aus:

„Warning: Memoryblock x Compression error: compressed size is larger than uncompressed! Datablock not compressed“.

Dieser logische Block wird unkomprimiert übernommen und als Kompressions-Methode wird im Flashcontainer für diesen logischen Block „0“ eingetragen.

29. „Compression“ + „Tool“ Auswahl – Festlegung wie die Kompression durchgeführt werden soll.

Bei „internal“ in Kombination mit einer der beiden Kompressionsmethoden von „A“ wird das Datenfile vom ODXCreate eingelesen, komprimiert und im Container komprimiert abgelegt

Bei „external tool“ wird vom ODXCreate das Datenfile eingelesen, die Rahmendaten (Start, Endadresse, Checksumme, unkomprimierte Größe etc.) ermittelt und das File anschließend an das angegebene Externe Tool (siehe Reiter „General Information“ die Punkte 23. bis 27.) übergeben und das Ergebnis des externen Kompressionstool im Container abgelegt.

Bei „compressed file“ erwartet ODXCreate bereits ein komprimiertes File, das im Container direkt übernommen wird. Da ODXCreate an der Stelle das Originalfile nicht kennt, kann ODXCreate hier auch keine Rahmendaten ermitteln. Diese müssen entweder über Manual address calculation „aktiv“ fest vorgegeben sein oder im Nachgang im Container manuell eingetragen werden.

30. „Encryption“ + „Methode“ Auswahl – Festlegung ob der Bootloader verschlüsselte oder unverschlüsselte Daten erwartet. Bei der Vorgabe „0“ werden vom Bootloader unverschlüsselte Daten erwartet. Die Auswahl 1 bis 9 sind dabei für Zulieferer eigene Verschlüsselungsverfahren reserviert. Die Zahl wird in den Container übernommen und im Flashjob bei dem Service RequestDownload (0x34) im dataFormatIdentifier bei encryptingMethod gesendet. Die Auswahl „A“ entspricht dabei der vom VW-Konzern im Standard-SW Bootloader integrierten Verschlüsselungsmethode (in zwei Ausführung mit oder ohne Padding-Bytes). ODXCreate kann diese Verschlüsselung ohne ein weiteres, externes Tool selbstständig durchführen.
- Das VW-Konzern Encryption-Modul erwartet aber immer eine Datenmenge Modulo 16/32, daher müssen die Source-Daten gegebenenfalls durch das Padding-Verfahren um die fehlenden Bytes ergänzt werden. Deshalb gibt es sowohl beim Kompressions-Modul als auch beim Verschlüsselungs-Modul die Auswahl „A (VW-AUDI) padding“. Beim Kompressions-Modul wird nach der Kompression der Padding-Algorithmus angewendet und beim Verschlüsse-

lungs-Modul wird der Padding-Algorithmus vor dem Verschlüsseln angewendet.

Bei der Kombination, dass sowohl bei der Kompression eine A-Variante ausgewählt wurde als auch eine A-Variante bei der Verschlüsselung gewählt wurde, so verwendet ODXCreate selbstständig die Kombination:

Kompression: A(VW-AUDI) padding

Verschlüsselung: A(VW-AUDI)

auch wenn diese Kombination in der Konfiguration so nicht vorgegeben oder gespeichert ist.

Wichtig! Das VW-Konzern Verschlüsselungsmodul verlängert das Datenfile um 1 bis maximal 16/32 Byte. Da damit die unkomprimierte Datenfilelänge nicht mehr, mit der im Flashjob tatsächlich zu übertragene Filelänge übereinstimmt, wird beim ODXCreate die tatsächlich zu übertragende Filelänge ermittelt. Diese neue Filelänge wird dann im Flashcontainer als „compressed size“ eingetragen. Die „compressed size“ ist in diesen Fall (Kombination: keine Kompression aber Verschlüsselung gewählt) um 1 bis 16/32 Byte länger, als die unkomprimierte Datenfilelänge. Im Flashjob wird dann die „compressed size“ für die tatsächlich zu übertragenden Datenbytes genutzt (Flashjob wertet, wenn vorhanden die „compressed size“ aus).

31. „Encryption“+„Tool“ Auswahl – Festlegung wie die Verschlüsselung durchgeführt werden soll.

Bei „internal“ in Kombination mit einer der beiden Verschlüsselungsmethoden von „A“ wird das Datenfile vom ODXCreate eingelesen, verschlüsselt und im Container verschlüsselt abgelegt.

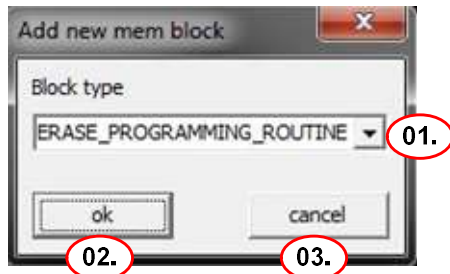
Bei „external tool“ wird vom ODXCreate das Datenfile eingelesen und an das angegebene Externe Tool (siehe Reiter „General Information“ die Punkte 28. bis 32.) übergeben und das Ergebnis des externen Verschlüsselungstool im Container abgelegt.

Bei „encrypted file“ erwartet ODXCreate bereits ein verschlüsseltes File, das im Container direkt übernommen wird.

32. „Initial value“ Eingabe – Hier kann für die Verschlüsselung ein Initialisierungsvektor vorgegeben werden (trifft nur für die „internal“ Verschlüsselung zu, bei beiden A-Varianten).
33. „Key“ Eingabe – Hier kann der Verschlüsselungs-Key festgelegt werden.

6.4 Add new mem block

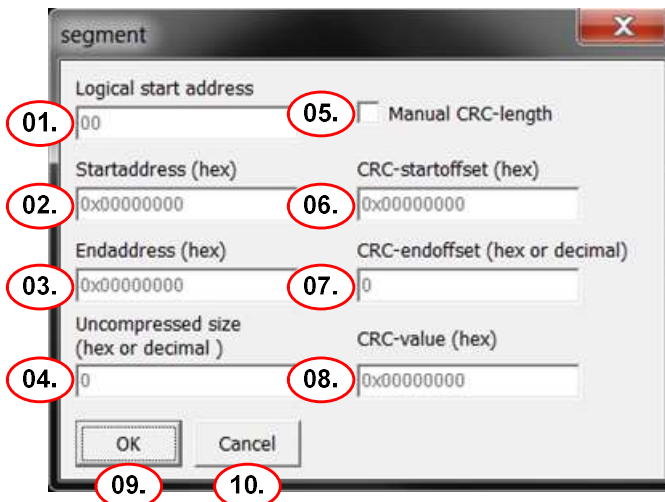
Wie bereits in 02. beschrieben wird beim hin zufügen eines weiteren Memory-Blocks das Auswahlfenster „Add new mem block“ geöffnet.



- 01. „Block type“ Auswahl – Hier können wie bereits unter 14. Beschrieb die verschiedenen Block types ausgewählt werden.
- 02. „ok“ Button – Bestätigung der Auswahl für den Block type.
- 03. „cancel“ Button – Abbruch der Eingabe.

6.5 Segment

Wie bereits in 22. und 27. beschrieben, kann über das segment-Fenster die Rahmengrößen manuell angepasst werden. Beim Einlesen des Datafiles werden dazu zunächst die Rahmengrößen vom ODXCreate ermittelt. Wenn die Option „Manual address calculation“ aktiviert ist, so können die Felder 02. bis 04. überschrieben werden. Ansonsten werden diese ausgegraut und können nur ausgelesen werden. Wenn unter 22. Als Checksum Type nicht „Manual“ oder „Intern“ ausgewählt wurde, dann gibt es folgende Eingabeoptionen.



- 01. „Logical start address“ Anzeige – Die logische Startadresse entspricht dem „Logical Block Index“ und kann im segment-Fenster nur angezeigt aber nicht geändert werden (Änderung nur durch siehe 15.).
- 02. „Startadresse“ Eingabefeld – Hier kann die Startadresse als Hex-Wert angegeben werden, ab welcher Adresse aus dem Datafile die Daten übernommen werden sollen.
- 03. „Endadresse“ Eingabefeld – Hier kann die Endadresse als Hex-Wert angegeben

werden, bis zu welcher Adresse die Daten aus dem Datafile übernommen werden sollen.

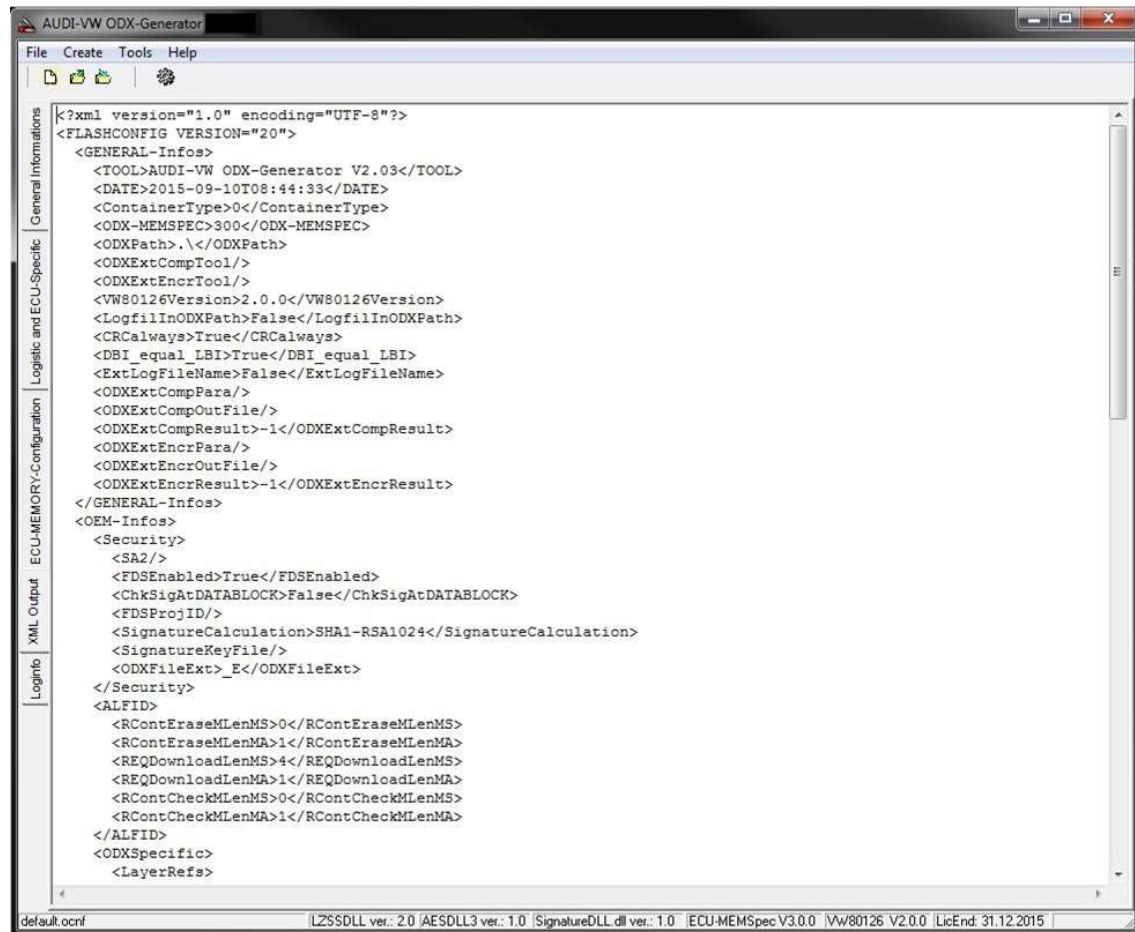
04. „Uncompressed size“ Eingabefeld – Hier kann die unkomprimierte Datengröße als Hex- oder Dezimal-Wert eingetragen werden (bei Hex ist 0x vorzustellen).
05. „Manual CRC-length“ Auswahl – Auswahl, ob die Checksumme über alle Daten dieses logischen Blockes berechnet werden (Standard) oder nicht. Nur wenn „Manual CRC-length“ Auswahl aktiviert ist, lassen sich die Eingabefelder 06. und 07. überschreiben. Dabei gibt es dann die Option über zwei Offsets am Start und Endbereich jeweils Daten aus der Checksummenberechnung nicht einfließen zu lassen
06. „CRC-startoffset (hex)“ Eingabe – Hier kann als Hex-Wert festgelegt werden, ab welchem Datenbyte die Checksummenberechnung starten soll. Das bedeutet aber auch, dass Änderungen in diesem „Offset“-Bereich zu keiner Änderung der Checksumme führen.
07. „CRC-endoffset (hex or decimal)“ Eingabe – Hier kann als Hex- oder Dezimal-Wert festgelegt werden, wie viele Datenbytes am Ende des Datafiles nicht in die Checksummenberechnung einfließen. Änderungen in diesen Datenbytes haben keine Auswirkungen auf die berechnete Checksumme.
08. „CRC-value (hex)“ Ausgabe – Zeigt die berechnete Checksumme über das Datafile an.
09. „OK“ Button – Einträge werden vom ODXCreate übernommen und das segment-Fenster wird wieder geschlossen.
10. „Cancel“ Button – Das segment-Fenster wird geschlossen und die Änderungen werden verworfen.

Wenn unter 22. Als Checksum Type „Manual“ oder „Intern“ ausgewählt wurde, dann entfallen die Eingabeoptionen 06. bis 07. und es gibt dafür das Eingabefeld 11.

11. „Length of CRC value“ Eingabe – Es kann die zu übertragende Checksummenlänge in Byte vorgegeben werden.
12. „CRC-value (hex)“ Vorgabe/Eingabe – Bei Checksum Type „Manual“ kann der Checksummenwert eingetragen werden und bei Checksum Type „Intern“ wird die Checksumme mit „0“ angegeben wie es die „Length of CRC value“ vorgibt.

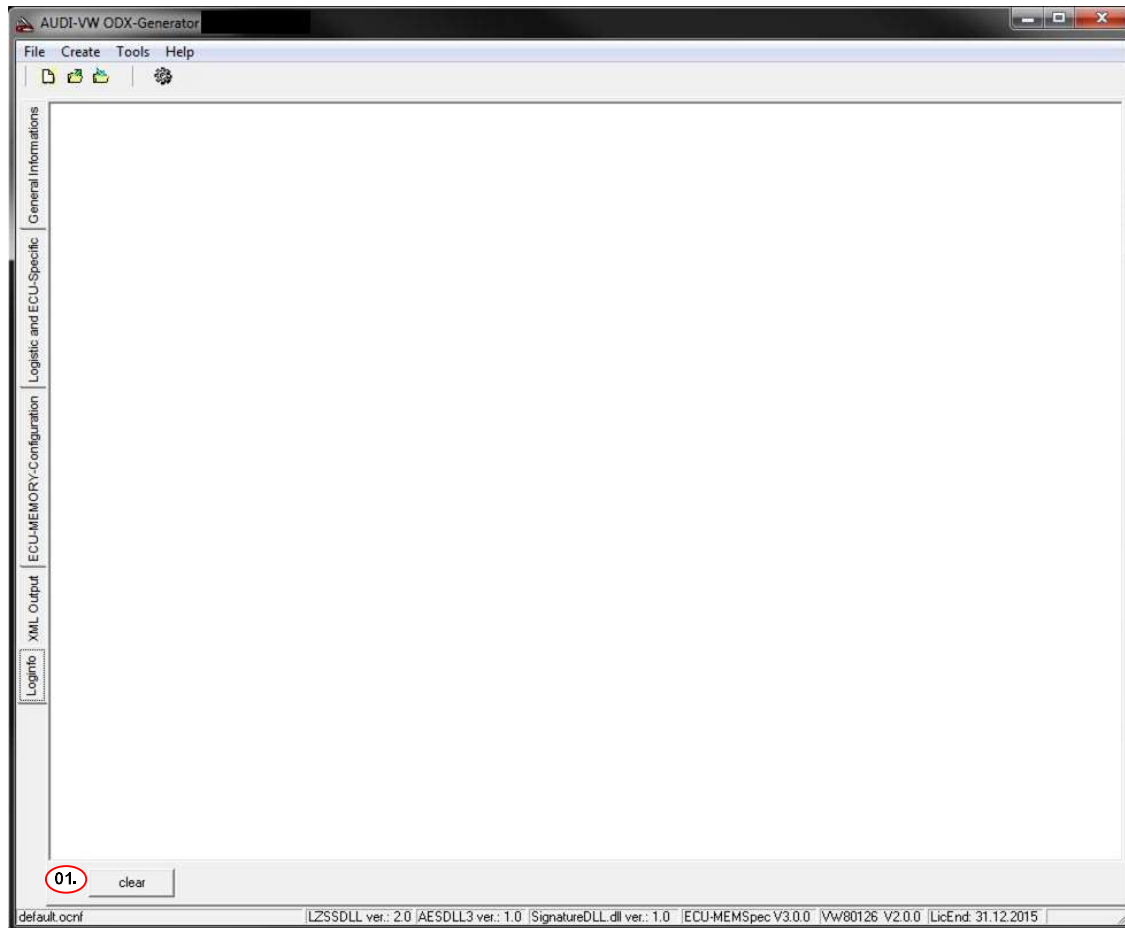
7 XML Output

Im XML Output Fenster wird der erzeugte Container als Vorschau erzeugt und es kann die Umsetzung geprüft werden.



8 Loginfo

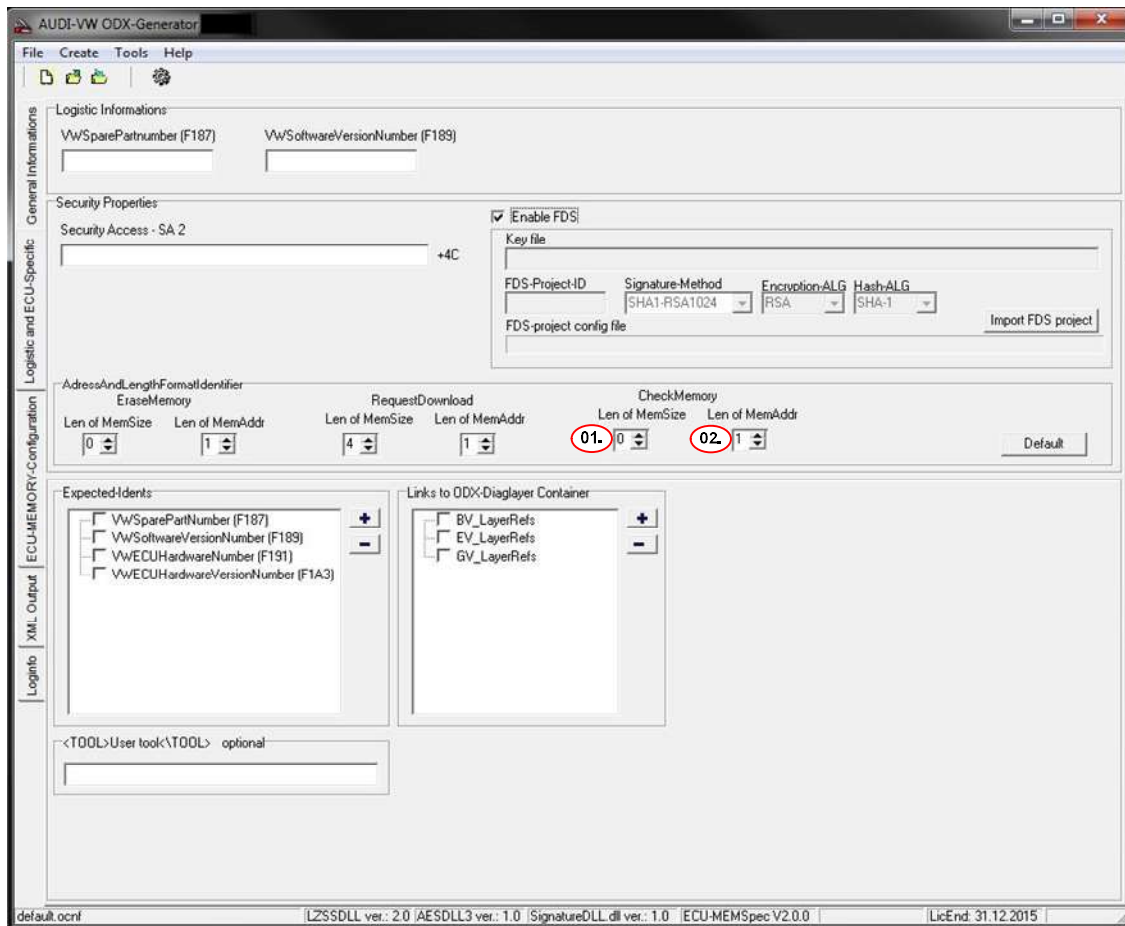
Im Loginfo-Fenster werden die Ergebnisse der Container-Erstellung angezeigt. Dabei wird für jeden Container-Erstellungsvorgang ein zusätzlicher Eintrag erzeugt. Im Loginfo-Fenster werden die gleichen Informationen ausgegeben wie im Logfile.



01. „clear“ Button – Damit können die Einträge im Loginfo-Fenster gelöscht werden.

9 Logistic and ECU-Specific für ECU-MEMSpec V2.0

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel Logistic and ECU-Specific für ECU-MEMSpec V3.0 erläutert. Die restlichen Punkte sind identisch zur V3.0

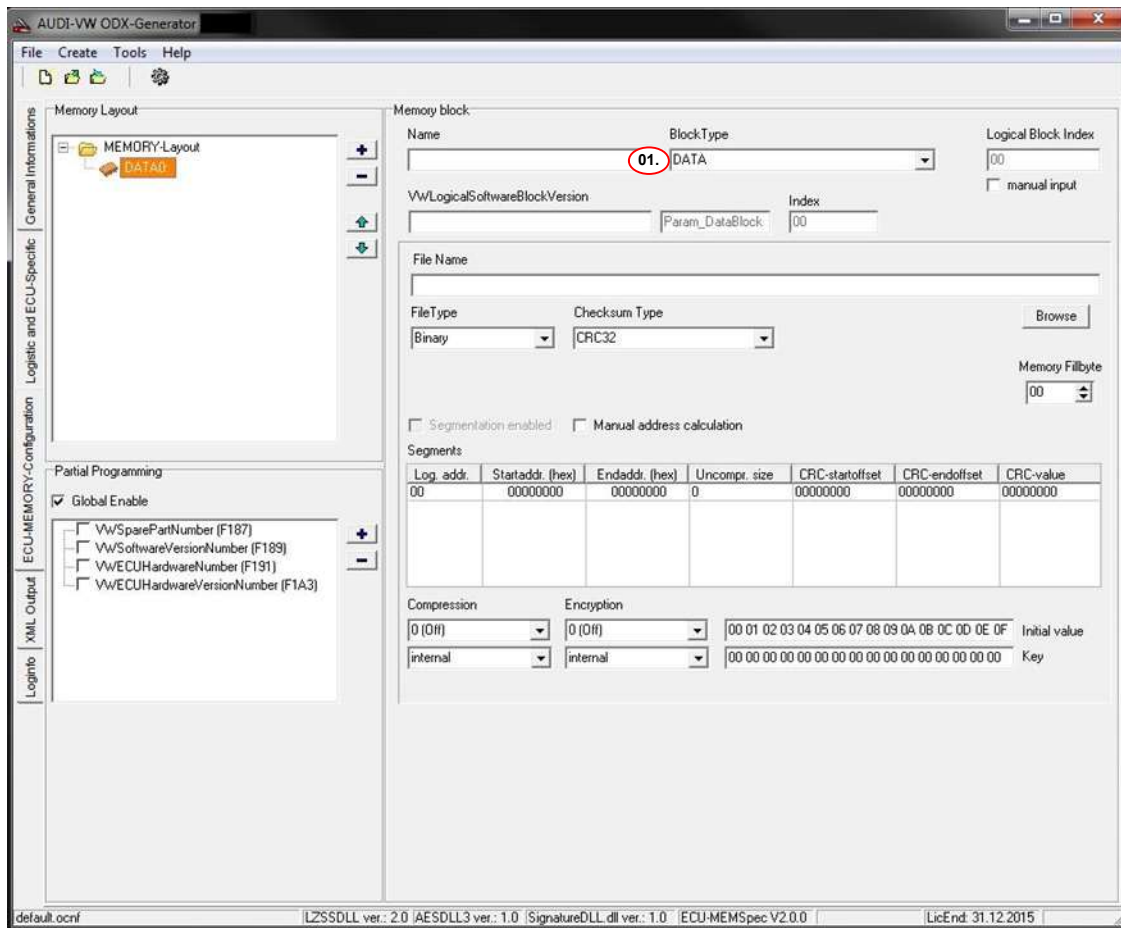


9.1 AdressAndLengthFormatIdentifier

01. „Len of MemSize“ (CheckMemory) – Der ALFID beim CheckMemory entspricht im Service 0x31 dem #5 Data Byte. Die Bits 4 bis 7 geben die Länge des Parameters für memorySize an (0x31 01 02 02 **xx xx xx**). Damit bestimmt **x** die Anzahl der xx-Pärchen im Service unmittelbar nach dem ALFID Parameter, z.B. 0x31 01 02 02 **24 xx xx xx xx xx**.
02. Len of MemAddr (CheckMemory) Vorgabe – Der ALFID beim CheckMemory entspricht im Service 0x31 dem #5 Data Byte. Die Bits 0 bis 3 geben die Anzahl der Bytes für die memoryAddress an (0x31 01 02 02 **xx xx xx...**). Damit bestimmt **x** die Anzahl der letzten xx-Pärchen im Service, z.B. 0x31 01 02 02 **24 xx xx xx xx xx xx**

10 ECU-MEMORY-Configuration für ECU-MEMSpec V2.0

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel Logistic and ECU-Specific für ECU-MEMSpec V2.0 erläutert. Die restlichen Punkte sind identisch zur V3.0



10.1 Memory block

01. „BlockType“ Vorgabe – Mit dem BlockType wird der Flashablauf im Flashjob gesteuert. Dazu gibt es 3 vorgegebene Arten von BlockType: DATA, DRIVER und ERASE

Die unterschiedlichen Typen und ihre Bedeutung:

DATA:

Stößt die Sequenz RequestDownload, TransferData und RequestTransferExit an

DRIVER:

Stößt dabei zweimal die Sequenz RequestDownload, TransferData und RequestTransferExit an. Einmal für die Erase Routine (Löschtreiber) und einmal für die Programmieroutine (Schreibtreiber).

ERASE:

Stößt die Routine für eraseMemory an.

11 Logistic and ECU-Specific für ECU-MEMSpec V3.1

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel Logistic and ECU-Specific für ECU-MEMSpec V3.0 erläutert. Die restlichen Punkte sind identisch zur V3.0

11.1 AdressAndLengthFormatIdentifier

01. „using“ Auswahl – Festlegung ob der Optionale Routine Control „Verify_partial_software_checksum“ vom Steuergerät unterstützt wird. Nur wenn die Auswahl aktiviert wurde, werden die Parameter für den Len of Checksum/Signature und Len of MemAddr auch in den Flashcontainer eingetragen.
02. „Len of Checksum/Signature“ (Verify_partial_software_checksum) Eingabe – Der ALFID beim Verify_partial_software_checksum entspricht im Service 0x31 dem #5 Data Byte mit den Bits 0 bis 3 (MSB) und im #6 Data Byte den Bits 0 bis 7 (LSB) und geben die Länge des Parameters für die checksum/signature an (0x31 01 05 44 xx xx xx). Damit bestimmt xx xx die Anzahl der xx-Pärchen am Ende des Services z.B. 0x31 01 05 44 x0 04. xx xx...xx xx xx xx).

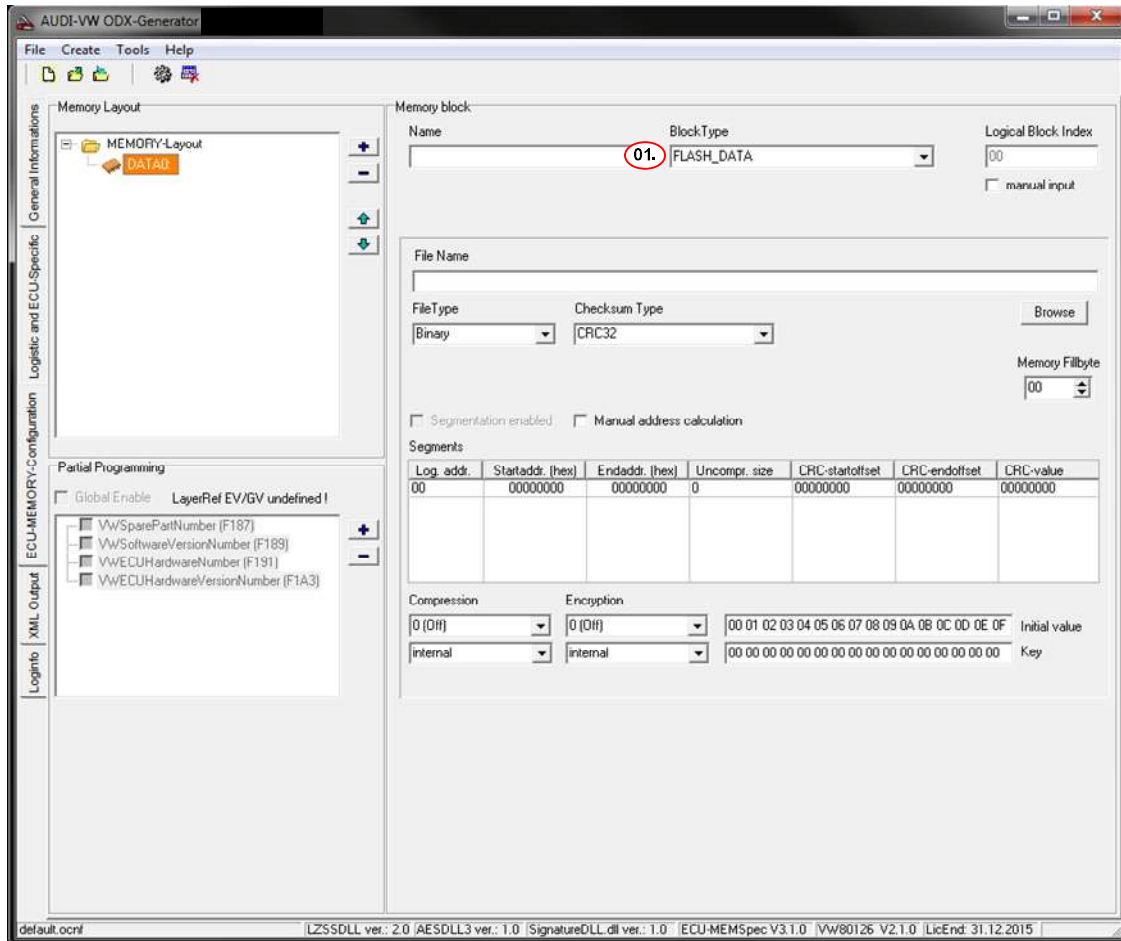
Die Eingabe der Länge für die Checksumme oder Signature ist als Dezimalwert vorzunehmen.

Eingabebereich liegt zwischen 0 und 4095 (Hex 0000 bis 01FF).

03. „Len of MemAddr“ (Verify_partial_software_checksum) Eingabe - Der ALFID beim Verify_partial_software_checksum entspricht im Service 0x31 dem #5 Data Byte. Die Bits 4 bis 7 geben die Anzahl der Bytes für die memoryAddress an (0x31 01 05 44 xx xx xx...). Damit bestimmt x die Anzahl der xx-Pärchen im Service unmittelbar nach dem ALFID Parameter,
z.B. 0x31 01 05 44 1x xx xx xx...xx

12 ECU-MEMORY-Configuration für ECU-MEMSpec V3.1

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel ECU-MEMORY-Configuration für ECU-MEMSpec V3.0 erläutert. Die restlichen Punkte sind identisch zur V3.0.



12.1 Memory block

01. „BlockType“ Vorgabe –Hinzufügen von den neuen BlockType:

ERASE_ASN1:

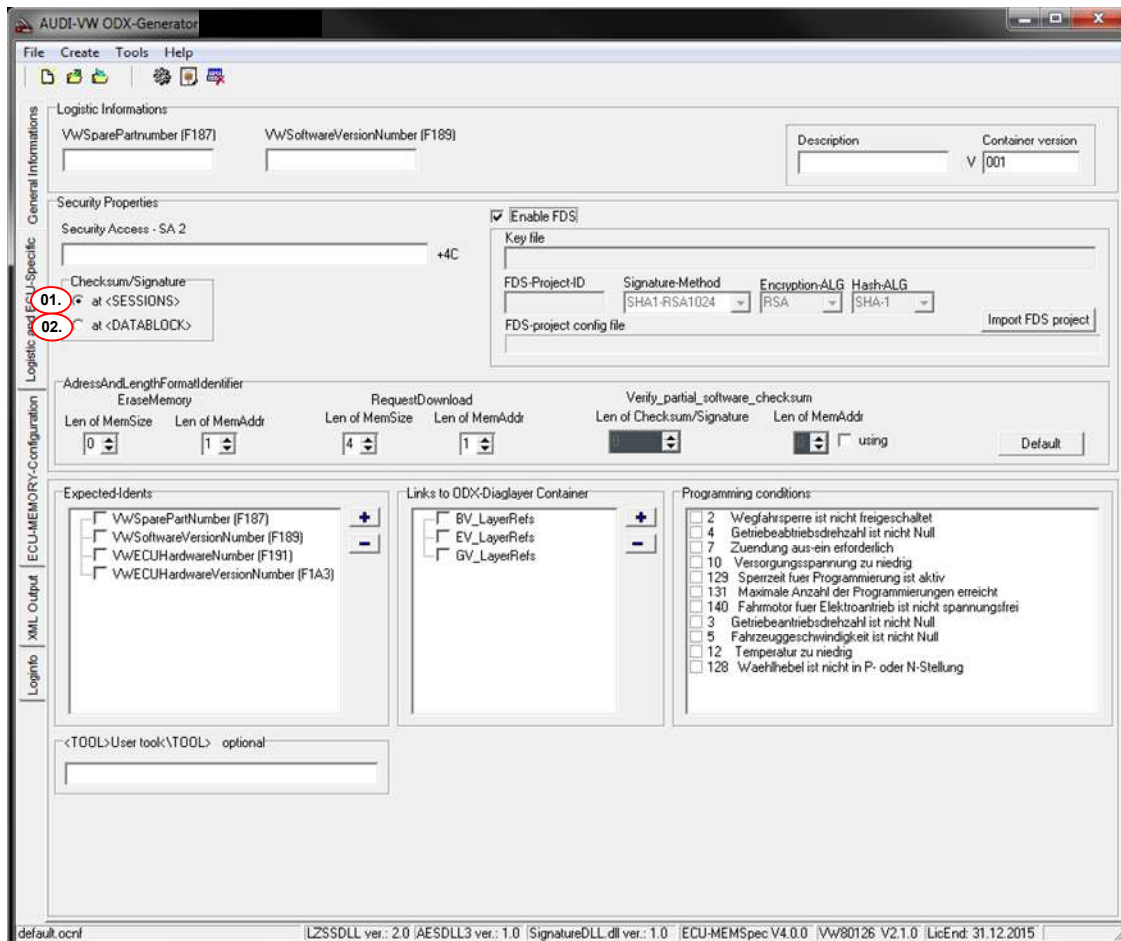
Neuer BlockType um reine Lösch-Container (ohne logische Datenblöcke) für das ASN1-Parser Zertifikat erstellen zu können.

13 Logistic and ECU-Specific für ECU-MEMSpec V4.0

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel Logistic and ECU-Specific für ECU-MEMSpec V3.1 erläutert. Die restlichen Punkte sind identisch zur V3.1. Die Besonderheit in der Einstellung ECUMEM 4.0.0 ist die Auswahl eines Flashcontainers als ODX oder PDX zu erstellen. In der aktuellen Version 2.03 des ODXCreate ist diese Umsetzung noch eine Prototypische Vorab Umsetzung (zum Zeitpunkt der Beauftragung gab es noch keine Released Version 4.0 der QLAH 80128 Teil 3, aus diesem Grund ist die Basisconfiguration ECUMEMSpec V4.0 nur für Testzwecke zu verwenden).

Zum Erstellen eines PDX-Flashcontainers siehe Kapitel „Allgemeine Elemente des ODXCreate“ und seine Unterkapitel.

Die Einstellung des Parameters „PDX Latebound datafile“ ist in Kapitel „General Informations“ in dem Unterkapitel „General“ beschrieben (Kurzerläuterung: mit dem Parameter „PDX Latebound datafile“ legt man fest, ob die Binär-Datafiles bereits im PDX abgelegt sein müssen [Einstellung = „false“] oder ob diese Binär-Datafiles auch beim Aufruf des Flashvorgangs hinzugelinkt werden können [Einstellung = „true“]. Die genaue Definition sollte in dem QLAH 80128 Teil 3 V4.0 stehen).



13.1 Security Properties

13.1.1 Checksum/Signature

Bei den ODX Flashcontainer werden die Security-Elemente (FW-SIGNATURE und FW-CHECKSUM) für einen logischen Block an die SESSIONS angehängt und über eine „VALIDITY-FOR“ Verknüpfung diesem logischen Block zugewiesen.

In der PDX Flashcontainer Umsetzung bei Porsche werden diese Security-Elemente aber nicht an der SESSION angehängt, sondern direkt an den DATABLOCK selbst.

Bis zur endgültigen Klärung, an welcher Stelle die Security-Elemente im neuen VW-PDX stehen sollen, gibt es daher die Option im ODXCreate für die PDX-Erstellung dies zu konfigurieren (bei der Erstellung eines ODX-Container spielt diese Einstellung keine Rolle).

01. „at <SESSIONS>“ Auswahl – Security-Elemente werden wie bei ODX-Flashcontainer an die SESSIONS angehängt.
02. „at <DATABLOCK>“ Auswahl – Security-Elemente werden wie beim Porsche PDX-Flashcontainern an den DATABLOCK angehängt.

14 Logistic and ECU-Specific für Bootloader-Datensätze nach V3.1

Im Reiter „Logistic and ECU-Specific“ für Bootloader-Datensätze (ab MLBevo) werden die logistischen Steuerungselemente bedatet. Die Bootloader-Datensätze der Version 3.1 werden nach den Vorgaben der QLAH 80128 Teil 3 V3.0 (ECUMEM 3.0.0) aufgebaut.

The screenshot shows the 'AUDI-VW ODX-Generator' application window. The 'Logistic and ECU-Specific' tab is active. The interface includes the following sections and highlighted elements:

- Logistic Informations:**
 - 01. DiagAddr [DDD]: 000
 - 02. Address [AAAA]: 71 00
 - 03. ProductId [PPP]: 000
 - 04. Version [VVV]: 0001
 - 05. ModulVariant [MM] / DatasetName [NNNNNNNNNNNN] [Text]: 00
 - 06. (Empty text field)
- Security Properties:**
 - 07. Security Access - SA 2
 - 08. ☒ Enable FDS
 - 09. ☒ External signature
 - 10. Key file: +4C
 - 11. FDS-Project-ID
 - 12. Signature-Method: DSDG2
 - 13. Encryption-ALG: RSA
 - 14. Hash-ALG: SHA-1
 - 15. FDS-project config file
 - 16. Import FDS project button
- AddressAndLengthFormatIdentifier:**
 - 17. EraseMemory: Len of MemSize: 0, Len of MemAddr: 2
 - RequestDownload: Len of MemSize: 4, Len of MemAddr: 2
 - Default button
- Expected-Ids:**
 - 18. (Empty list box)
- Links to ODX-Diaglayer Container:**
 - 19. BV_LayerRefs
- Programming conditions:**
 - 20. List of conditions with checkboxes:
 - ☐ Wegfahrsperre ist nicht freigeschaltet
 - ☐ Getriebebetriebsdrehzahl ist nicht Null
 - ☐ 7 Zuendung aus-ein erforderlich
 - ☐ 10 Versorgungsspannung zu niedrig
 - ☐ 129 Sperzeit fuer Programmierung ist aktiv
 - ☐ 131 Maximale Anzahl der Programmierungen erreicht
 - ☐ 140 Fahrmotor fuer Elektroantrieb ist nicht spannungsfrei
 - ☐ 3 Getriebeantriebsdrehzahl ist nicht Null
 - ☐ 5 Fahrzeuggeschwindigkeit ist nicht Null
 - ☐ 12 Temperatur zu niedrig
 - ☐ 128 Waehlhebel ist nicht in P- oder N-Stellung
- Bottom Section:**
 - 21. <TOOL>User tool<TOOL> optional

The status bar at the bottom displays: default.ocnf | LZSSDLL ver.: 2.0 | AESDLL3 ver.: 1.0 | SignatureDLL.dll ver.: 1.0 | ECU-MEMSpec V3.0.0 | Vw80126 V2.0.0 | LicEnd: 31.12.2015

14.1 Logistics Informations

In den Logistics Informations werden die nach der Dateinamenskonvention für den Datensatzdownload Generation 2 Version 3.1 vorgegebenen Namensfelder bedatet.

01. „DiagAddress[DDD]“ Eingabe – Eingabe der Tester Diagnoseadresse mit 3 Stellen.
02. „Address[AAAA]“ Eingabe – Eingabe der logischen Blockadresse. Da für den Bootloader-Datensatz der Bereich von 0x7100 bis 0x71FF vorgegeben ist, können auch nur noch die letzten beiden Stellen der Adresse eingegeben werden.
03. „ProductId[PPP]“ Eingabe – Eingabe der Produkt-Identifikation bei dem erst

einsetzenden Fahrzeugprojekt (3 Stellige Hauptgruppe aus der VW-AUDITeilenummer).

04. „Version[VVVV]“ Eingabe – Eingabe der 4 stelligen Datensatzversion, die der VWLogicalSoftwareBlockVersion im Bootloader entspricht.
05. „ModulVariante/Datasename[MM/NNNNNNNNNNNN]“ Eingabe – Eingabe der 2 stelligen Modulvariante (ist z.B. gedacht zur Unterscheidung zwischen High, Low etc. Ausstattungen) oder des 12 stelligen Datensatznamen.
06. „[Text]“ Eingabe – Hier kann optional der Datensatzname um einen Freitext zur besseren Unterscheidung erweitert werden.

14.2 Security Properties

07. „Security Access - SA 2“ – Eingabe des SA2-MiniBefehlssatzes für die Freischaltung in der Programming Session. Achtung der Ende-Befehl „+4C“ wird vom ODXCreate automatisch ergänzt.
08. „Enable FDS“ Auswahl – Festlegung, ob das Steuergerät FlashDatenSicherheit unterstützt (Wegfahrsperren-, Komponentenschutz- und Tachomanipulationschutz-Steuergeräte). Wenn die Auswahl „FDS“ gewählt wurde, dann werden die Eingabefelder für FDS wie in unter „Logistic and ECU-Specific für ECU-MEMSpec V3.0.“ dargestellt eingeblendet. Zusätzlich gibt es noch die Auswahl 09.
09. „External signature“ Auswahl – Festlegung, wenn das Steuergerät FDS relevant ist, ob die Signatur im ODX-Container (wie bei den Flashcontainer) abgelegt wird, oder ob die Signatur als zusätzliches externes XML-Signaturfile erstellt wird.
10. „Keyfile“ Anzeige – Zeigt an, auf welchen Signaturschlüssel das FDS-Project-XML File verweist.
11. „FDS-Project-ID“ Anzeige – Interpretiert aus dem FDS-Project-XML File, welcher FDS-Project-ID diesem Schlüssel zugeordnet ist.
12. „Signature-Methode“ Auswahl – Bei den Bootloader-Datensätzen gibt es zwei neue unterschiedliche Signatur-Methoden:
 - a. Zulieferer-Signatur „SUP“ – Diese Signatur ist für einen nicht bekannten Zulieferer Signaturschlüssel vorgegeben. Der ODXCreate erstellt dann als Vorbereitung im ODX-Container ein leeres FW-SIGNATURE Feld, das im Nachgang vom Zulieferer mit der berechneten Signatur für den jeweiligen logischen Block ergänzt werden muss. Bei der Signature-Methode „SUP“ wird die Signatur im Bootloader-Datensatz erwartet (13. bis 16 wird damit ausgegraut).
 - b. Krypto42-Signature „DSDG2“ – Abweichend zu den Flashcontainern wurde für die Bootloader-Datensätze eine neue Signature-Method „DSDG2“ eingeführt. Dabei kann die Signatur während der Entwicklungsphase im Bootloader-Datensatz enthalten sein, während für den Serienprozess eine „Externe“ Signature verwendet wird, die beim Einstellen in system42 neu berechnet wird (es werden beim Einlagern „Interne“ Signaturen aus dem Bootloader-Datensatz entfernt und externe Signaturen erstellt).
13. „Encryption-ALG“ Anzeige - Interpretiert aus dem FDS-Project-XML File, welcher Verschlüsselungs-Algorithmus verwendet wird.
14. „Hash-ALG“ Anzeige - Interpretiert aus dem FDS-Project-XML File, welcher Hash-Algorithmus verwendet wird.
15. „FDS-project config file“ Anzeige – Zeigt das aktuell geladene FDS-Project-XML File
16. „Import FDS project“ Button – Startet den Browser, um das FDS-Project-XML File laden zu können. Das FDS-Project-XML File muss aus system42 heruntergeladen werden und enthält unter anderen den Entwicklungs-Signaturschlüssel.

Dadurch werden auch die notwendigen Erweiterungen im Container vorbereitet, damit dieser von krypto42 beim Einstellen interpretiert werden kann.

14.3 AdressAndLengthFormatIdentifier

Siehe „Logistic and ECU-Specific für ECU-MEMSpec V3.0“. Dort werden die gleichen Felder verwendet und in der Anleitung auch beschrieben.

14.4 Expected-Idents

Da es beim Bootloader-Datensatz keine Expected-Idents gibt, ist dieses Feld ohne Funktion.

14.5 Links to ODX-Diaglayer Container

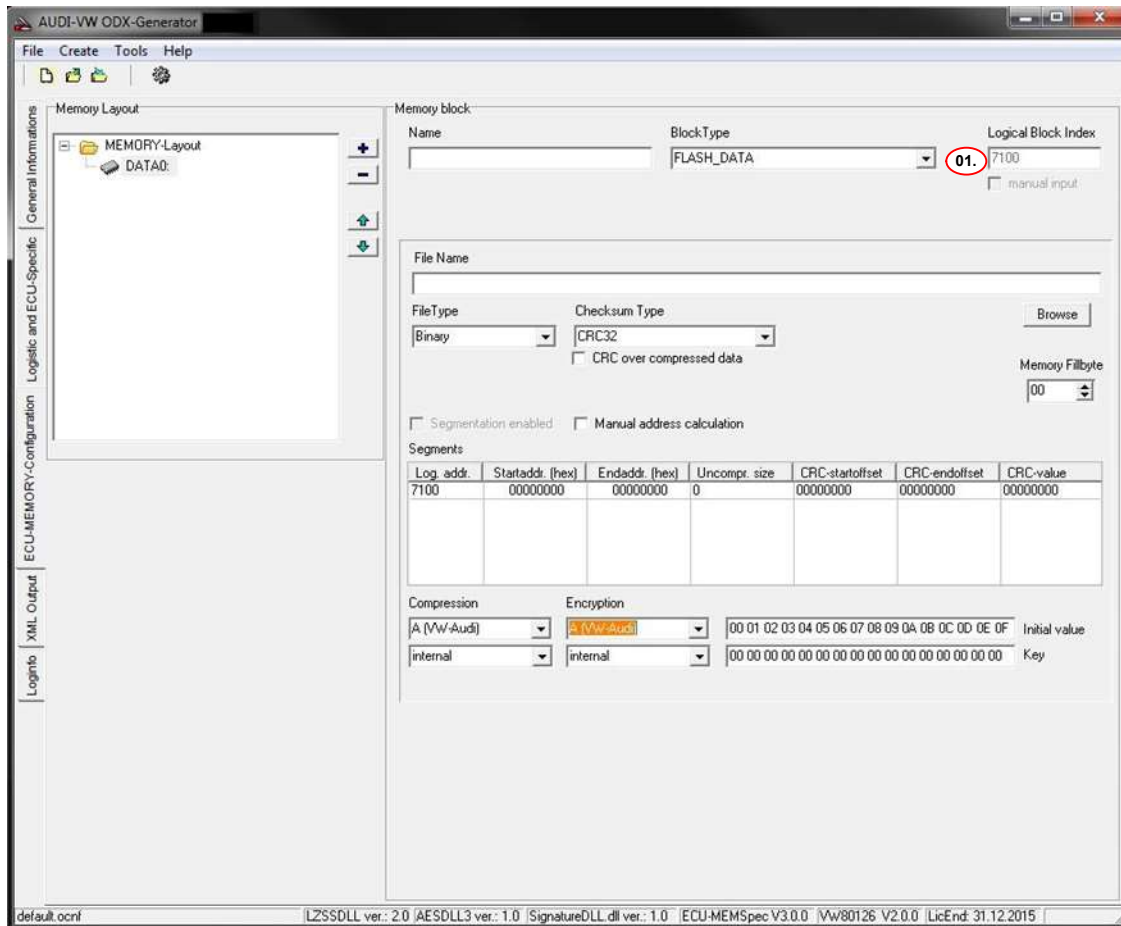
Siehe „Logistic and ECU-Specific für ECU-MEMSpec V3.0“. Dort werden die gleichen Felder verwendet und in der Anleitung auch beschrieben. Allerdings sind bei Bootloader-Datensätzen nur die BV_LayerRefs erlaubt und die EV_ und GV_LayerRefs entfallen.

14.6 Programming conditions

Siehe „Logistic and ECU-Specific für ECU-MEMSpec V3.0“, dort werden die gleichen Felder verwendet und in der Anleitung auch beschrieben.

15 ECU-MEMORY-Configuration für Bootloader-Datensätze nach V3.1

Siehe „ECU-MEMORY-Configuration für ECU-MEMSpec V3.0“, dort werden die gleichen Felder verwendet und in der Anleitung auch beschrieben.



Bei den Bootloader-Datensätzen gibt es folgende Einschränkungen:

- Es muss genau einen logischen Block vom Typ „FLASH_DATA“ geben
- Der Logical Block Index für den logischen Block vom Typ „FLASH_DATA“ entspricht dabei der Address[AAAA] wie in Logistic and ECU-Specific für Bootloader-Datensätze festgelegt und kann daher im Feld 01. „Logical Block Index“ nicht verändert werden.
- Von den restlichen BlockTypes darf es optional, jeweils einen weiteren logische Block geben.
- Es darf aber jeder BlockType nur maximal einmal pro Bootloader-Datensatz vorkommen.
- Da es beim Bootloader-Datensatz nur einen logischen Datenblock gibt, entfällt das Eingabefenster für das partielle Flashen.

16 Logistic and ECU-Specific für Bootloader-Datensätze nach V3.2

Die Bootloader-Datensätze der Version 3.2 werden nach den Vorgaben der QLAH 80128 Teil 3 V3.1 (ECUMEM 3.1.0) aufgebaut.

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel „Logistic and ECU-Specific für Bootloader-Datensätzen nach V3.1“ beschrieben.

16.1 Logistics Informations

In den Logistics Informations werden die nach der Dateinamenskonvention für den Datensatzdownload Generation 2 Version 3.2 vorgegebenen Namensfelder bedatet. Im Gegensatz zu der Version 3.1 gibt bei folgenden Eingabefeldern eine Änderung:

01. „DiagAddress[DDDD]“ Eingabe – Eingabe der Tester Diagnoseadresse mit 4 Stellen.
02. „ModulVariante/Datasetname[NNNNNNNNNNNNNN]“ Eingabe – Eingabe des 12 stelligen Datensatznamen.

16.2 AdressAndLengthFormatIdentifier

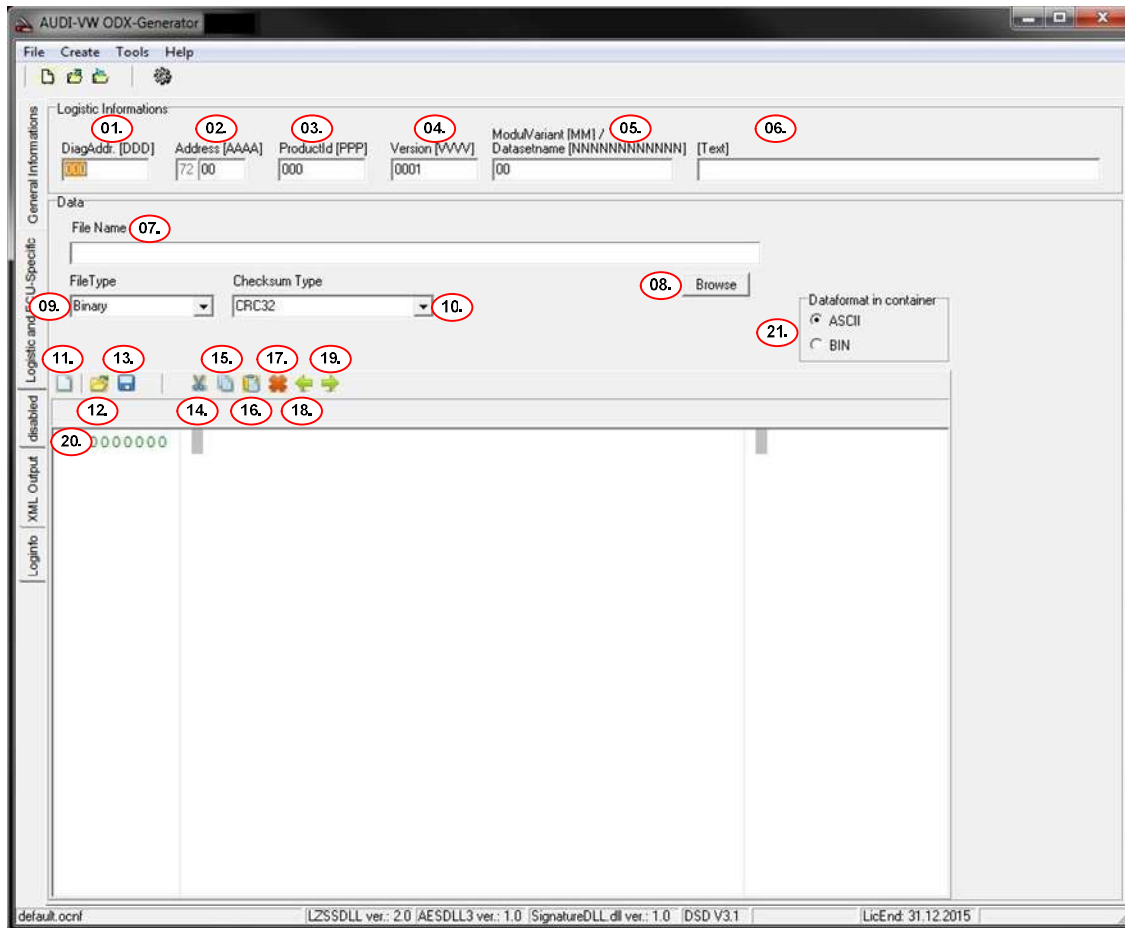
Siehe „Logistic and ECU-Specific für ECU-MEMSpec V3.1“.

17 ECU-MEMORY-Configuration für Bootloader-Datensätze nach V3.2

Es gelten die gleichen Einstellungen und Vorgaben wie im Kapitel ECU-MEMORY-Configuration für Bootloader-Datensätze nach V3.1

18 Logistic and ECU-Specific für Applikations-Datensätze nach V3.1

Im Reiter „Logistic and ECU-Specific“ für Applikations-Datensätze (ab MLBevo) werden die logistischen Steuerungselemente bedatet.



18.1 Logistics Informations

In den Logistics Informations werden die nach der Dateinamenskennung für den Datensatzdownload Generation 2 Version 3.1 vorgegebenen Namensfelder bedatet.

- 01. „DiagAddress[DDD]“ Eingabe – Eingabe der Tester Diagnoseadresse mit 3 Stellen.
- 02. „Address[AAAA]“ Eingabe – Eingabe des Identifiers. Da für den Applikations-Datensatz der Bereich von 0x7200 bis 0x72FF vorgegeben ist können auch nur noch die letzten beiden Stellen des Identifiers eingegeben werden.

- 03. „ProductId[PPP]“ Eingabe – Eingabe der Produkt-Identifikation bei dem zuerst einsetzenden Fahrzeugprojekt (3 Stellige Hauptgruppe aus der VW/AUDI Teilenummer).
- 04. „Version[VVVV]“ Eingabe – Eingabe der 4 stelligen Datensatzversion.
- 05. „ModulVariante/Datasename[MM/NNNNNNNNNNNN]“ Eingabe – Eingabe der 2 stelligen Modulvariante (ist z.B. gedacht zur Unterscheidung zwischen High, Low etc. Ausstattungen) oder 12 stelligen Datensatznamen.
- 06. „[Text]“ Eingabe – Hier kann optional der Datensatzname um einen Freitext (zur besseren Unterscheidung) erweitert werden.

18.2 Data

- 07. „File Name“ Ausgabefeld – zeigt einen Datenfile-Namen an, wenn für diesen Applikations-Datensatz die Daten von einer Datei geladen wurden.
- 08. Browse“ Button – Öffnet den Explorer, um ein Datenfile laden zu können, das den Binärdatenanteil des Applikations-Datensatzes enthält.
- 09. „File Type“ Auswahl – Einstellung, von welchem Typ das Datenfile ist. Dabei wird die Voreinstellung auch bei der „Browse“ Datenfile-Suche berücksichtigt, so dass nur noch Datenfiles dieses Typs im Browser angezeigt werden.
- 10. „Checksum Type“ Auswahl – Hier können verschiedene Checksummen Berechnungsalgorithmen ausgewählt werden. Dabei berechnet ODXCreate über das hochgeladene Datenfile die Checksumme nach dem ausgewählten Checksum Type und hängt die Checksumme beim Erzeugen des Applikations-Datensatzes am Ende der Binärdaten an.

18.3 Editor-Funktion

Wenn die Binärdaten nicht über ein externes Datenfile geladen werden sollen, so können die Binärdaten auch über den Editor eingegeben oder das geladene Datenfile im Editor nachbearbeitet werden. Dazu bietet der Editor folgende Funktionen:

- 11. „New“ – Es wird ein temporäres, neues und leeres Datenfile erzeugt.
- 12. „Browse“ - Öffnet den Explorer um ein Datenfile laden zu können, das den Binärdatenanteil des Applikations-Datensatzes enthält.
- 13. „Speichern“ – Speichert den Inhalt des Editors als Datenfile. ODXCreate kann nur einen Applikations-Datensatz erstellen, wenn zum Erstellungszeitpunkt ein gespeichertes Datenfile vorliegt, das geladen werden kann. Daher muss vor dem Erstellen des Applikations-Datensatzes ein neu erzeugtes oder manipuliertes Datenfile, das sich „nur“ im Editor befindet, abgespeichert werden.
- 14. „Cut“ – Mit dieser Funktionen können Teile des Datenfiles im Editor gelöscht werden, wenn diese zuvor im Editor markiert wurden.
- 15. „Copy“ – Dient zum Kopieren von markierten Bereichen im Editorfenster.
- 16. „Paste“ – Dient zum Einfügen des Inhalts aus dem Zwischenspeichers an die markierte Cursorstelle.
- 17. „Clear selection“ – Löscht den markierten Bereich im Editor.
- 18. „Undo“ – Nimmt die letzte Änderung zurück.
- 19. „Redo“ – Rückgängig von Undo.
- 20. „Editorfenster“ – Bearbeitungsfenster für die Binärdaten.
- 21. „Dataformat in container ASCII/BIN“ – Formatkonvertierung der Binärdaten in den Applikations-Datensatz Container.

Standard ist die Darstellung von ASCII-Zeichen, die als Hexwert (Beispiel 0x34 = „4“) übertragen werden. Damit steht im Applikations-Datensatz als zu übertragende Daten 0x34.

Durch die Einstellung „BIN“ wird der Hexwert wieder zu einem ASCII-Zeichen gewandelt und als Daten übertragen. In diesen Beispiel steht damit im fertigen Flashcontainer als Daten „4“ und nicht „34“.

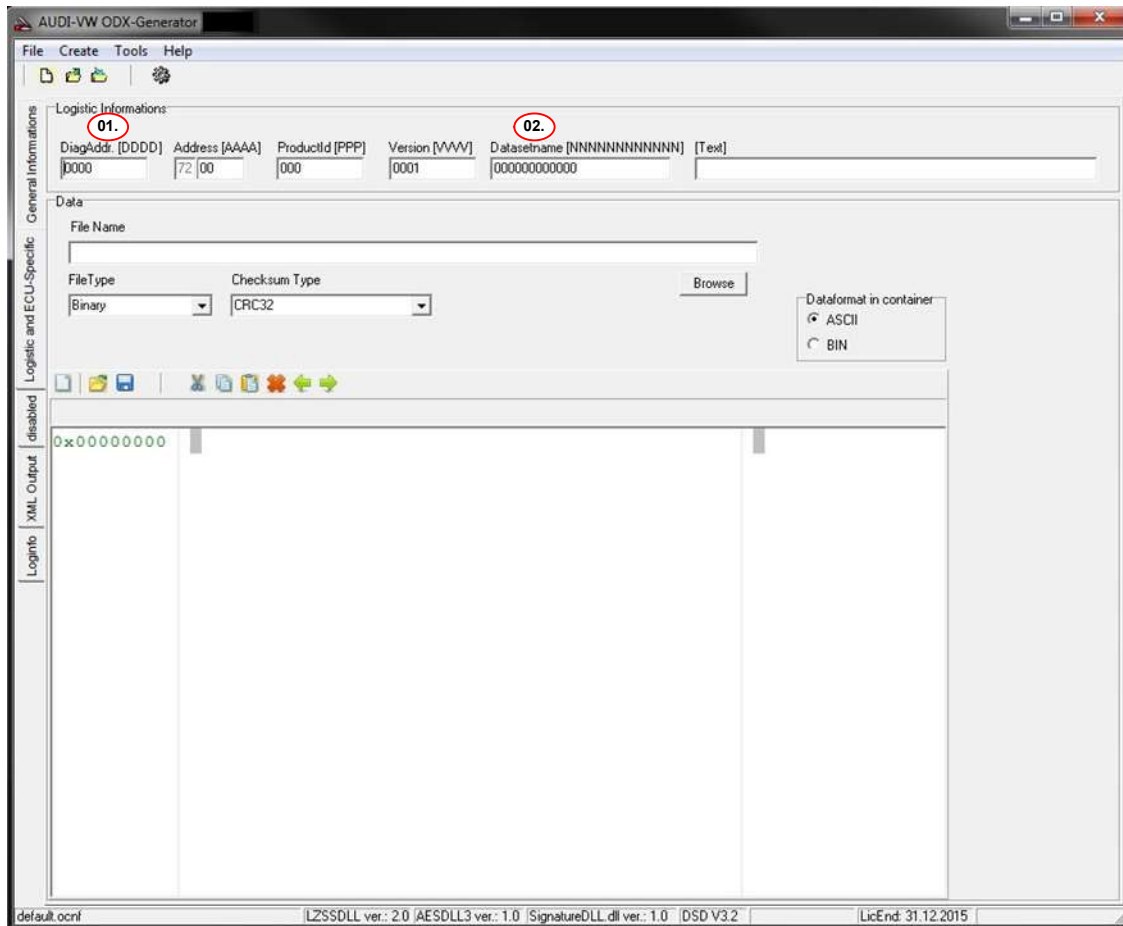
Dieses Feature wurde nur für spezielle Steuergeräte eingebaut, die dadurch eine Verkleinerung der zu übertragenden Datenmenge erreichen. Unabhängig von der Einstellung ASCII oder BIN werden die 4 Bytes der Datensatzversion im ASCII-Format abgelegt.

Es muss aber beachtet werden, dass Daten die über die Editor-Funktion eingegeben oder bearbeitet wurden, erst gespeichert werden müssen, bevor ein Applikations-Datensatz Container erstellt werden kann.

Der ODXCreate nutzt zur Container Erstellung nicht die aktuellen Binärdaten im Editorfenster sondern benötigt immer ein gespeichertes File, auf das im Eingabefeld 07. „File Name“ verwiesen ist. Wenn Daten im Editorfenster erstellt oder verändert wurden und dann anschließend mit 13. „Speichern“ gesichert werden, dann übernimmt ODXCreate den Pfad und Dateinamen automatisch in das Eingabefeld 07. „File Name“.

19 Logistic and ECU-Specific für Applikations-Datensätze nach V3.2

Es werden in diesem Kapitel nur noch die Abweichungen zum Kapitel „Logistic and ECU-Specific für Applikations-Datensätzen nach V3.1“ beschrieben.



19.1 Logistics Informations

In den Logistics Informations werden die nach der Dateinamenskennung für den Datensatzdownload Generation 2 Version 3.2 vorgegebenen Namensfelder bedatet. Im Gegensatz zu der Version 3.1 gibt es bei folgenden Eingabefeldern eine Änderung:

01. „DiagAddress[DDDD]“ Eingabe – Eingabe der Tester Diagnoseadresse mit 4 Stellen.
02. „ModulVariante/Datasetname[NNNNNNNNNNNN]“ Eingabe – Eingabe des 12 stelligen Datensatznamen.

20 Erstellen eines ASN1-Parser Flashcontainers und Löscontainers

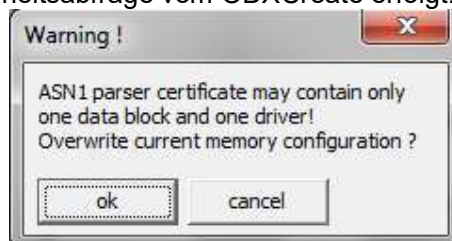
Um einen ASN1-Parser Flashcontainer (zum temporären Umschalten von Serien-Signaturschlüssel auf Entwicklungs-Signaturschlüssel) erstellen zu können, wird aus system42 das ANS1-Parser Zertifikat als Binär-Datei benötigt (muss vom Steuergeräteverantwortlichen erstellt und gedownloaded werden). Der erstellte ANS1-Parser Flashcontainer muss in system42 eingestellt werden, damit dieser eine Serien-Signatur enthält. Ansonsten kann das neue Zertifikat nicht auf das Steuergerät geflashed werden. Der ANS1-Parser Flashcontainer ist wie eine Applikations-Software über den Mechanismus „Update Programmierung“ in das Steuergerät einzubringen.

Um auf dem Steuergerät das ASN1-Parser Zertifikat wieder löschen zu können (Rückschaltung von Entwicklungs-Signaturschlüssel auf den Serien-Signaturschlüssel), ist ein spezieller Löscontainer notwendig, der ebenfalls über den Mechanismus „Update Programmierung“ auf das Steuergerät aufgespielt wird.

20.1 Erstellung des ANS1-Parser Flashcontainers mit ODXCreate

01. Basiskonfiguration für einen Flashcontainer mit ECUMEM 3.0 (oder höher) auswählen (es kann auch als Ausgangsbasis die Konfiguration für die Applikations-Software verwendet werden, es muss nur beim Speichern darauf geachtet werden, dass die Konfiguration einen neuen Namen bekommt).
02. Die Reiter General Informations, Logistic and ECU-Specific werden wie beim Erstellen einer Applikations-SW befüllt. Bei der VWSoftwareVersionNumber sollte eine Version verwendet werden, die nicht mit einer Applikations-Software verwechselt werden kann (da der ASN1-Parser auch in system42 eingestellt werden muss, ist darauf zu achten, dass die VWSoftwareVersionNumber beim einchecken eineindeutig sein muss).
03. Im Reiter ECU-MEMORY-Configuration muss ein Datenblock vom BlockType „ASN1_DATA“ angelegt werden (wenn das Steuergerät für das Löschen und/oder Schreiben Treiber benötigt, dann müssen diese ebenfalls im Memory-Layout erzeugt werden). Der logische Block vom Typ ASN1_DATA erhält bei „Logical Block Index“ die Adresse 0xFE, dabei darf sich in Abhängigkeit des ALFIDs für die Len of MemAddr nur die Anzahl der führenden 00 vor der Adresse FE geändert werden (0xFE; 0x00FE etc.).

Es ist zu beachten, dass mit der Auswahl des BlockType „ASN1_DATA“ eine Sicherheitsabfrage vom ODXCreate erfolgt.



Durch das Bestätigen mit dem „ok“ Button werden alle logischen Blöcke vom BlockType „FLASH_DATA“ gelöscht. Es verbleiben bei einer zu vor geladenen

- Konfiguration nur die logischen Blöcke die einen Treiber enthalten. Dem BlockType „ASN1_DATA“ wird automatisch der FileType „ASN1-Parser Cert.“ zugewiesen.
04. Diesem logischen Block darf kein Binärfile sondern nur das von system42 erzeugte ASN1-Parser Zertifikat (Dateiendung *.der) zugewiesen werden.
 05. Für diesen „speziellen“ Flashcontainer gelten ansonsten die gleichen Einstellungen, die auch für das Erstellen eines Applikations-Flashcontainers gelten.
 06. Konfiguration unter neuem Namen speichern und Flashcontainer erstellen.
 07. Der erzeugte Flashcontainer muss abschließend in system42 Seriensigniert werden bevor dieser in das Steuergerät geflashed werden kann.

Hinweise:

Sollte das Steuergerät das Partiellen Flashen unterstützten (Partial Programming „Global Enable“ aktive) so wird beim Erstellen des ASN1-Parser Zertifikats die Partielle Programmierung automatisch vom ODXCreate deaktiviert, das wird aber mit einer Warnung vom ODXCreate angezeigt.

Ebenfalls ist zu beachten, dass das ASN1-Parser Zertifikat nicht weiter komprimiert werden kann, so dass das komprimierte ASN1-Parser Zertifikat bei einer gewählten Kompressions-Methode größer werden würde als das unkomprimierte ASN1-Parser Zertifikat. In diesem Fall wird vom ODXCreate eine Warnung erzeugt und die Kompression wird für diesen logischen Block deaktiviert.

20.2 Erstellen eines Löscontainers, zum Löschen des ASN1-Parser Zertifikats

20.2.1 Allgemein

01. Bei Steuergeräten die zum Löschen des ASN1-Parser Zertifikats einen eigenen Löschtreiber benötigen (BlockType ERASE_ROUTINE oder ERASE_PROGRAMMING_ROUTINE), müssen die erzeugten Löscontainer mit dem Entwicklungs-Schlüssel signiert werden (Steuergeräte sind durch das ASN1-Parser Zertifikat auf den Entwicklungs-Schlüssel umgeschaltet).

20.2.2 Löschen des ASN1-Parser Zertifikats bei ECUMEM 3.0.0 Steuergerät ohne Löschtreiber

01. Es muss eine neue Basiskonfiguration für einen Flashcontainer mit ECUMEMSpec Version 2.0 erstellt werden (wichtig: der Löscontainer wird nicht nach ECUMEM 3.0 gebaut, daher muss hier eine neue Konfiguration erstellt werden).
02. Die Reiter General Informations und Logisitic and ECU-Specific sind gemäß den Vorgaben für einen Applikations-SW Container zu bedaten. Es muss nur beachtet werden eine eigene VWSoftwareVersionNumber zu wählen, die nicht mit einem regulären Applikations-SW Stand verwechselt werden kann.
03. Bei Steuergeräten, die keinen Löschtreiber benötigen, ist zu beachten, dass der zusätzliche CheckMemory „Len of Memory Size“ und „Len of MemAddr“ keine Rolle spielt (Default-Werte beibehalten), da beim Löschen des Zertifikats nur die Löschroutine benutzt wird.
04. Im Reiter ECU-MEMORY-Configuration wird ein logischer Block vom Typ „ERASE“ mit dem „Logical Block Index“ FE (dazu muss „manual input“ aktiviert sein) erstellt.

Wenn die Len of MemAddr größer 1 ist, so muss der „Logical Block Index“ um führende 0 ergänzt werden (z.B. Len of MemAddr = 2 -> Logical Block Index = 00FE)


05. Konfiguration speichern und Löscontainer erstellen.

20.2.3 Löschen des ASN1-Parser Zertifikats bei ECUMEM 3.0.0 Steuergerät mit Löschtreiber

01. Konfiguration für die Applikations-SW verwenden und als neue Konfiguration abspeichern.
02. In der abgespeicherten Konfiguration in dem Reiter „ECU-MEMORY-Configuration“ in dem Memory Layout alle logische Blöcke vom BlockType „FLASH_DATA“ löschen.
03. Den ersten (verbleibenden) logischen Block mit dem Block type „ERASE_ROUTINE“ b.z.w. „ERASE_PROGRAMMING_ROUTINE“ behalten.
04. Die restlichen logischen Blöcke müssen ebenfalls gelöscht werden.
05. Es muss ein neuer logischer Block mit dem „Logical Block Index“ 0xFE erstellt werden (bei der Länge der Adresse die Len of MemAddr beachten und Gegebenenfalls mit führenden „00“ auffüllen). Für die Eingabe der logischen Adresse muss der Hacken bei „manual input“ gesetzt werden.
06. Dieser logische Block muss nun mit einer EraseMemory Sequenz beginnen (Block type „PROGRAMMING_ROUTINE“ oder „FLASH_DATA verwenden“).
07. Bevor nun ein Löscontainer erstellt werden kann muss noch beachtet werden, eine eigene VWSoftwareVersionNumber zu wählen, die nicht mit einem regulären Applikations-SW Stand verwechselt werden kann.
08. Konfiguration speichern und Löscontainer erstellen.

20.2.4 Löschen des ASN1-Parser Zertifikats ab ECUMEM 3.1.0 Steuergerät

Ab der ECUMEM 3.1.0 wurde ein neuer BlockType „ERASE_ASN1“ eingeführt, der ausschließlich das ASN1-Parser Zertifikat (an der logischen Block Adresse 0xFE) löscht. Das besondere an diesen BlockType ist, dass damit eine EraseMemory Sequenz angestoßen wird, ohne dass anschließend noch Daten an das Steuergerät übertragen werden. Dadurch gibt es für die meisten Anwendungsfälle eine Automatisierung bei der Erstellung eines ASN1-Parser Löscontainer.

01. Konfiguration für die Applikations-SW verwenden und als neue Konfiguration abspeichern.
02. Eine eigene VWSoftwareVersionNumber wählen, die nicht mit einem regulären Applikations-SW Stand verwechselt werden kann.
03. Konfiguration speichern
04. Entweder in der Menüleiste „Create“ -> „ERASE-ASN1“ auswählen oder das Symbol 
05. Der Löscontainer wurde erzeugt

Durch die Automatisierung werden alle logischen Blöcke vom Block Type „FLASH_DATA“ und „PROGRAMMING_ROUTINE“ entfernt. Wenn die Konfiguration einen Lös- oder Lös-Schreib-Treiber enthält wird dieser logische Block beibehalten und die restlichen logischen Blöcke werden gelöscht. Anschließend wird ein logischer Block von dem Block Type „ERASE-ASN1“ erstellt. Dieser neue logische Block mit dem BlockType „ERASE-ASN1“ er-

hält dabei den Logical Block Index FE und ist für den Flashjob die Anweisung das ASN1-Parser Zertifikat zu löschen.

Wenn in dem Flash MEMORY-Layout mehrere logischen Blöcke enthalten sind, die einen Löscho- oder Löscho/Schreib-Treiber haben, dann kann der ODXCreate keinen Container erstellen. Für diesen Fall muss der ASN1-Parser Löschocontainer manuell erstellt werden.

Dazu kann im Memory-Layout der BlockType „ERASE_ASN1“ ausgewählt werden.

21 Reengineering von ODXCreate Konfigurationen aus Datencontainern

Für das Reengineering von ODXCreate Konfigurationen gibt es einige Einschränkungen:

01. Es dürfen nur Datencontainer verwendet werden die mit dem ODXCreate erzeugt und nicht manipuliert wurden (Container die nicht mit dem ODXCreate erzeugt wurden, wurden nicht getestet).
02. Es können keine Flashcontainer reengineert werden die noch nach der ECUMEM 2.0.0 oder älter erstellt wurden.
03. Der ODXCreate reengineiert keine Binärdaten und deren Einstellungen (Es kann z.B. kein FileType erkannt werden oder die Einstellung bei „Manual address calculation“ wieder gewonnen werden).
04. Der ODXCreate kann keine Signatur oder Verschlüsselungs-Key aus einem Datencontainer auslesen.

21.1 Anleitung für das Reengineering

01. Nach dem Start des ODXCreate in der Menüleiste „Tools“ mit der linken Maustaste markieren.
02. Aus dem Drop-Down Menü die Einstellung „Import config from“ auswählen
03. Das Format des Datencontainers bestimmen:
 - Applikations-Datensätze: -> XML
 - Bootloader-Datensätze: -> ODX
 - Flashcontainer: -> ODX oder ODX-F (muss aus dem PDX extrahiert werden)
04. Es öffnet sich das Browser-Fenster und es muss der gewünschte Datencontainer geöffnet werden.
05. Es öffnet sich erneut das Browser-Fenster um die reengineierte Konfiguration abzuspeichern.

22 Konvertierung von Basiskonfigurationen

Der ODXCreate arbeitet zum Erstellen von Datencontainer immer auf einer Basiskonfiguration (Flashcontainer ECUMEMSpec Vx.x.0). In der Vergangenheit waren diese Basiskonfigurationen auch nicht voneinander ableitbar (ECUMEMSpec V2.0.0 kann nicht nach ECUMEMSpec V3.0.0 umgewandelt werden oder umgekehrt, ebenso wenig kann ein Applikations-Datensatz in einen Bootloader-Datensatz konvertiert werden oder ein Bootloader-Datensatz in einen Flashcontainer). Da es nun zahlreiche neue Versionen gibt, die man (mit Einschränkungen) umwandeln kann (z.B. ECUMEMSpec 3.0.0 nach ECUMEMSpec 3.1.0 oder sogar ECUMEM Spec 4.0.0), ist es hilfreich eine bereits vorhandene Konfiguration von einer Basiskonfiguration in eine andere Basiskonfiguration zu konvertieren.

Für das Konvertieren von ODXCreate Konfigurationen gibt es einige Einschränkungen:

01. Es können keine ECUMEMSpec 2.0.0 Basiskonfigurationen konvertiert werden.
02. Es können nur Konfigurationen der gleichen Basiskonfigurations-„Familie“ (Flashcontainer, Bootloader-Datensätze und Applikations-Datensätze) konvertiert werden.

22.1 Anleitung für das Konvertieren

01. Laden der Konfiguration die konvertiert werden soll.
02. In der Menüleiste „Tools“ mit der linken Maustaste markieren.
03. Aus dem Drop-Down Menü die Einstellung „Convert config to“ auswählen
04. Der ODXCreate bestimmt nun welche Konvertierungen möglich sind und bietet diese zur Auswahl an.
05. Aus der Auswahl nun die gewünschte neue Basiskonfiguration mit der linken Maustaste bestätigen.
06. Der ODXCreate öffnet ein Browser-Fenster um die konvertierte Konfiguration abspeichern zu können.
07. Speichern der neuen Konfiguration.

23 Umwandlung von ODX-Flashcontainer in PDX-Flashcontainer und umgekehrt

23.1 Anleitung für die Umwandlung von Flashcontainer

01. Nach dem Start des ODXCreate in der Menüleiste „Tools“ mit der linken Maustaste markieren.
02. Aus dem Drop-Down Menü die Einstellung „Convert container“ auswählen
03. Die gewünschte Formatumwandlung:
 - PDX -> ODX
 - ODX -> PDXMit der linken Maustaste bestimmen.
04. Es öffnet sich das Browser-Fenster und es muss der gewünschte Flashcontainer geöffnet werden.
05. Der gewählte Flashcontainer wird als Kopie in dem gewandelten Flashcontainer Format im gleichen Verzeichnis wie der Original Flashcontainer abgelegt.

Hinweis: PDX Flashcontainer die durch eine Umwandlung aus einem ODX-Flashcontainer erzeugt wurden, haben die Security-Elemente immer am DATABLOCK (siehe dazu Kapitel „Logistic and ECU-Specific für ECU-MEM Spec V4.0“ das Unterkapitel „Checksum/Signature“).

24 Die ocnf-Datei

Die ocnf-Datei ist die Konfigurationsdatei für den ODXCreate. In der ocnf werden alle benötigten Daten, die für das Erstellen eines Flashcontainers, Bootloader- oder Applikations-Datensatzes benötigt werden, gespeichert. Da mit jedem neuen Feature, das eine Auswahl oder Eingabe im ODXCreate erfordert, auch ein neuer Parameter im ocnf gespeichert werden muss, ist die Liste der OCNF-Parameter dynamisch und daher in dem beigefügten Excelsheet „OCNF_Doku_[Datum].xls“ beschrieben. Dabei wird zwischen kompatiblen und inkompatiblen Erweiterungen unterschieden. Bei kompatiblen Erweiterungen gibt es Default-Werte, die zum Beispiel dem Verhalten vor dem Einführen des neuen Parameters Entsprechen. Eine ältere „OCNF“ kann daher immer noch zum Erzeugen genutzt werden. Bei inkompatiblen Änderungen müssen vom Anwender die neuen Parameter befüllt oder zumindest bestätigt werden, wenn es einen Default-Wert als Vorschlag gibt.

25 Die Kommandozeilensteuerung

ODXCreate kann auch über die Kommandozeile bedient werden. Dazu muss zunächst eine gültige ocnf über die GUI erstellt und abgespeichert werden (es ist nicht möglich über die Kommandozeile eine neue ocnf zu erstellen). Über die Kommandozeile können mit einer gültigen ocnf, Container erstellt werden. Dazu gibt es auch eine Reihe von Parametern, die das Erstellen im Batch-Modus erleichtern sollen. Die Auswahl an möglichen Parametern kann entweder in der GUI über die Menüleiste unter „Help“ angezeigt werden oder ist auch in dem Dokument „ODXCreateKommandozeile_Vxxx.pdf“ abgelegt.

Ein Beispiel für den Aufruf einer Batchdatei die 3 Container erzeugt.

```
D:\ODXCreator\ODXCreate_1_0_60>ODXCreate.exe -s 0001 D:\ODXCreator\ODXCreate_1_0_60\RefTest\AppIDSTest_044_7201.ocnf  
  
D:\ODXCreator\ODXCreate_1_0_60>ODXCreate.exe -s 0021 D:\ODXCreator\ODXCreate_1_0_60\RefTest\Bootloader_DS_007_7100.ocnf  
  
D:\ODXCreator\ODXCreate_1_0_60>ODXCreate.exe -s 0989 D:\ODXCreator\ODXCreate_1_0_60\RefTest\ECUMEM3CRCAfterCOM_MK3B7980924.ocnf
```

26 Änderungshistorie

26.1 Änderungen zur Version 2

01. Erweiterung Datensatzdownload Generation 2 Version 3.2
02. Erweiterung QLAH 80128 Teil 3 V3.1 (ECUMEM 3.1.0)
03. Erweiterung QLAH 80128 Teil 3 V4.0 (ECUMEM 4.0.0)
04. Konvertierung von Basiskonfigurationen
05. Reengineering von Konfigurationen aus Datencontainer (Eingeschränkt)
06. Unterstützung von großen Flashcontainern (Architektur Änderung im ODXCreate)
07. Unterstützung zum Erzeugen von ASN1-Parser Zertifikaten
08. Unterstützung zum Erzeugen von Löschcontainer für das ASN1-Parser Zertifikat
09. Neue Prüfungen und Plausibilisierung bei der Datencontainer Erstellung eingebaut mit Warnungen, die dem Ersteller Hinweise auf mögliche Fehler geben soll.
10. Integration (optional) von Zuliefer-Tool-Namen in den Flashcontainer
11. Bugfixing bei Bootloader-Datensätze werden keine OWN-IDENT Elemente mehr erzeugt.
12. Bei den ALFID für die Bootloader-DS gibt es harte Vorgabe für den Len of MemAddr
13. Intel- und Motorola-Hexdateiverarbeitung optimiert (Unterstützung zusätzlicher Tags)
14. Neues Funktionalität zur Anzeige der Struktur der Hex-Daten „Structure“
15. Unterstützung von RSA3072 und ECDSA
16. Unterstützung von QLAH 80128 Teil 3 V4.1 (ECUMEM 4.1.0)
17. Unterstützung von QLAH 80128 Teil 3 V4.2 (ECUMEM 4.2.0)
18. Unterstützung von QLAH 80128 Teil 3 V4.3 (ECUMEM 4.3.0)
19. Unterstützung von Q-LAH 020_204_LAH.000.036.H_Update filebasierter Systeme_V1.2
20. Unterstützung Generierung automatischer Rückflashschutzblock gemäß Q-LAH 80126

27 Weitere Dokumente, Beschreibungen und Ergänzungen

- ODXCreateExterneTools_Vx.xx.pdf
- ODXCreateKommandozeile_Vxxx.pdf
- OCNF_Doku_[Datum].xls
- Interner Link für die verschiedenen Versionen des ODXCreate:
\\AUDIINTEPro.in.audi.vwg\ProjectS\EE_Elektrik_Elektronik\Flashen\1000_Projekte\1900_Flashkreis\19D0_Tools\ODXCreate
- Zugriff für Zulieferer (dort werden nicht die neuesten Beta-Stände abgelegt):
<https://iproject.vw.vwg/iproject/vw/DiagPortalV2.Flashen.html>

28 Lizenzinformation über ODXCreate-Komponenten

ODXCreate verwendet verschiedene Komponenten. Version, Quelle und Lizenzbedingungen sind in den Unterkapiteln aufgezählt. Die zugehörigen Lizenzvereinbarungen sind im ODXCreate-Unterverzeichnis „Lizenzdateien“ einsortiert.

28.1 Kcontrols

Produktbezeichnung: Kcontrols

Verwendete Komponente: TKHexEditor

Verwendete Version: 1.7

Quelle der Komponente: <http://www.tkweb.eu/en/delphicomp/>

Art der Lizenz: Freie Nutzung (Lizenz für die freie Nutzung unveränderter Inhalte)

Lizenzdatei 1: .Lizenzdateien\kcontrol_1_7\kcontrols_kcontrolsaz.lkp.license.txt

Lizenzdatei 2: .Lizenzdateien\kcontrol_1_7\kcontrols_readme.txt

28.2 NativeXML

Produktbezeichnung: NativeXML

Verwendete Komponente: NativeXML

Verwendete Version: 2.38

Quelle der Komponente: <http://www.simdesign.nl/nativexml.html>

Art der Lizenz: proprietäre Lizenz

Lizenzdatei: .Lizenzdateien\nativexml_2_38\NativeXML_2_38_LICENSE.txt

28.3 Jedi

Produktbezeichnung: Jedi API 2.3 and JEDI WSCL 0.9.3

Verwendete Komponente: JwaBCrypt, JwaNCrypt

Verwendete Version: 2.3

Quelle der Komponente: <https://sourceforge.net/projects/jedi-apilib/files/>

Art der Lizenz: <https://www.mozilla.org/en-US/MPL/1.1/>

Lizenzdatei: .Lizenzdateien\JEDI_API_2.3_and_JEDI_WSCL_0.9.3\MPL_1.1.txt

28.4 TurboPower Abbrevia

Produktbezeichnung: TurboPower Abbrevia

Verwendete Komponente: TurboPower Abbrevia

Verwendete Version: 5.2

Quelle der Komponente: <https://sourceforge.net/projects/tpabbrevia/>

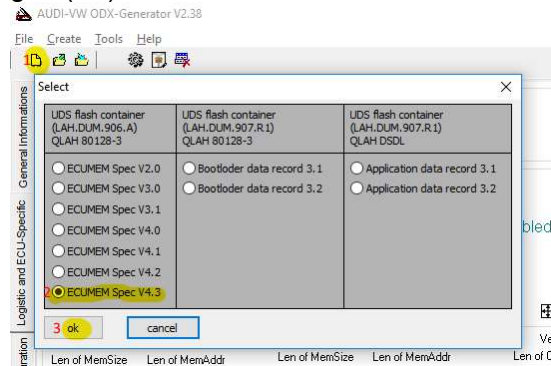
Art der Lizenz: <https://www.mozilla.org/en-US/MPL/1.1/>

Lizenzdatei: .Lizenzdateien\TurboPower_Abbrevia_5.2\MPL_1.1.txt

28.5 Erläuterte Beispielkonfiguration mit Rückflashschutzblock (ECUMEM-4.3)

28.5.1 Erstellen eines neuen Flashcontainers

Zum Erstellen eines neuen Flashcontainers zunächst das Icon „1“ auswählen. Danach öffnet sich ein Menü zur Auswahl des Flashcontainers. Dort je nach geforderter Q-LAH Version die Auswahl treffen – im Beispiel 2 ist es „2“ ECUMEM Spec V4.3. Anschließend „OK“ bestätigen („3“).



28.5.2 Auswahl der allgemeinen Informationen des Flashcontainers

Folgende Beispielhafter Konfigurationsablauf wird für 80126 / 80128-3 V4.3 –Flashcontainer mit Rückflashschutz und FDS empfohlen:

0. Auswahl des Reiters „Logistic and ECU-Specific“
1. Eingabe der Spare-Partnummer
2. Eingabe der Softwareversion, welche sich nach dem Flashen im SG befindet.
3. Eingabe eines kurzen erklärenden Textes im Feld Description
4. Eingabe der Container-Version
5. Auswahl der Security-Method für 80126-Flashen ist dies „SA2“
6. Eingabe des Security-Access Algorithmus gemäß 80126 (Hinweis dieses Beispiel darf nicht in produktiven Versionen der Steuergeräte verwendet werden)
7. Aktivieren der Flashdatensicherheit zur Erstellung von entwicklersignierten Flashcontnern
8. Aktivieren des Rückflashschutzes
9. Auswahl des XML-FDS-Projekts (Export aus Version42; nur p8-basierte Schlüssel können von ODXCreate verarbeitet werden)
10. Konfiguration EraseMemory/Len of MemAddr mit dem Wert 2 um konforme Längen der logischen Blockadressen zu unterstützen
11. Konfiguration Request Download/Len of MemSize mit dem Wert 4
12. Konfiguration Request Download/Len of MemAddr mit dem Wert 2 um konforme Längen der logischen Blockadressen zu unterstützen (passend zu Schritt 10)
13. Aktivierung der Angaben zu „Verify_partial_software_checksum“
14. Konfiguration Verify_partial_software_checksum/Len of MemSize mit dem Wert 4 (passend zu Schritt 11)
15. Konfiguration Verify_partial_software_checksum/Len of MemAddr mit dem Wert 2 (passend zu Schritt 10 und Schritt 12)

16. Auswahl der BasisVariante (zunächst „BV_LayerRefs“ selektieren, dann „+“ klicken und dann Basisvariante ohne Major/Minor-Version gemäß ODX-SG-Bedatung eingeben (Wert muss manuell ermittelt werden, und kann nicht automatisch selektiert werden.)
17. Analog Schritt 16 können noch ECU-Varianten eingegeben werden, hierbei muss die Major-Version mit angegeben werden.
18. Siehe Schritt 17 – zusätzliche ECU-Varianten
19. Aktivieren der Check-Box „ORU participant“

AUDI-VW ODX-Generator V2.38

File Create Tools Help

Logistic Informations

VwSparePartNumber (F187) 1 AAAAAAAAAA 2 X123 VwSoftwareVersionNumber (F189)

Description 3 BEISPIELRFS430 Container version 4 V4001

Security Properties

Security Access - SA 2 6 6805814A05870A22128349 +4C

Security Method 7 SA 2 8 Enable FDS 8 Enable downgrade protection

Key file D:\ODX\Creator\Testordner\FDS_XML\keys\FDSProject_1989_E.p8

FDS-Project-ID 1989 Signature-Method SHA256-RSA3072 9 Import FDS project

FDS-project config file D:\ODX\Creator\Testordner\FDS_XML\config_FDSProject_1989_E.xml

AddressAndLengthFormatIdentifier

EraseMemory Len of MemSize Len of MemAddr 10 2 11 4 12 2

RequestDownload Len of MemSize Len of MemAddr 14 4 15 2 using 13

Verify_partial_software_checksum Len of Checksum/Signature Len of MemAddr

Expected Idents

VwSparePartNumber (F187) +

VwSoftwareVersionNumber (F189) -

VwECUHardwareNumber (F191)

VwECUHardwareVersionNumber (F1A3)

Links to ODX-Diaglayer Container

BV_LayerRefs +

BV_ECU1_16

EV_LayerRefs

EV_ECU1_001 17

EV_ECU1_002 18

GV_LayerRefs

Programming conditions

1 Motordrehzahl ist nicht Null

2 Wegfahrsperre ist nicht freigeschaltet

3 Getriebeantriebsdrehzahl ist nicht Null

4 Getriebeantriebsdrehzahl ist nicht Null

5 Fahrzeuggeschwindigkeit ist nicht Null

6 Regelung aktiv

7 Zuendung aus-ein erforderlich

8 Keine Programmierspannung

9 Zuendung (Klemme 15) ist nicht eingeschaltet

10 Versorgungsspannung zu niedrig

11 Temperatur zu hoch

12 Temperatur zu niedrig

128 Waehlebel ist nicht in P- oder N-Stellung

129 Sauerzeit fuer Programmierung ist aktiv

ECU Diagnostic Class

DK4

DK3

DK2F

19 ORU participant

UFS enabled

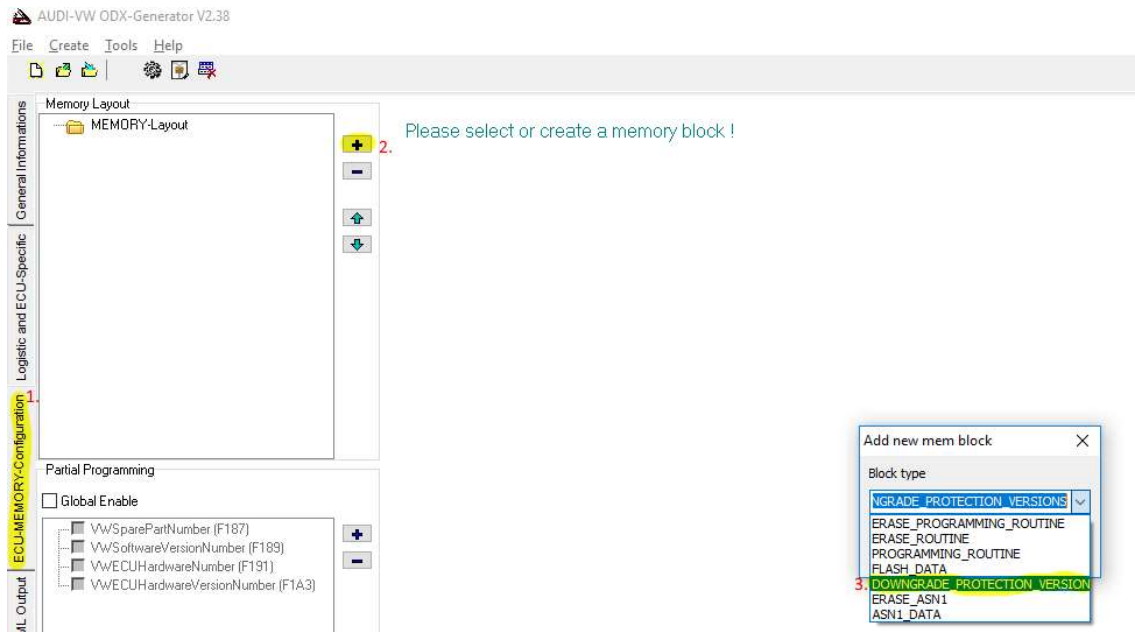
Server-Type Embedded Reset-Type HardReset Execution Timeout [ms] 1000

Update-Execution NoExecution Reset-Duration [ms] 1000 Check Timeout [ms] 1000

Update Coordination

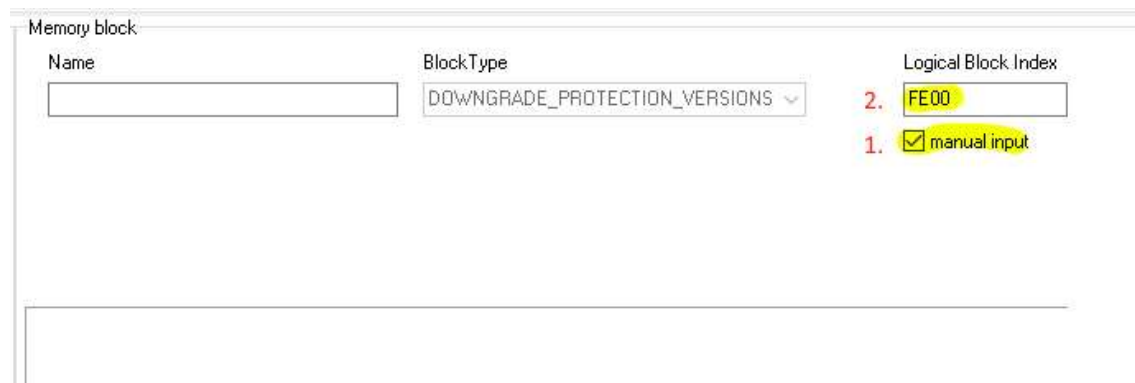
28.5.3 Anlegen eines Rückflashschutzblocks

1. Zunächst muss in den Reiter „ECU-MEMORY-Configuration“ gewechselt werden
2. Mittels „+“ können neue Blocktypen erstellt werden – es öffnet sich ein PopUp
3. Durch Auswahl von „DOWNGRADE_PROTECTION_VERSIONS“ und Bestätigung mit OK wird ein Rückflashschutz-Block erstellt



28.5.4 Konfiguration des Rückflashschutztreiberblocks

1. Mittels Aktivieren der Checkbox „manual input“ wird das manuelle Eingeben des logischen Blockindex ermöglicht
2. Hier kann zum Beispiel FE00 ausgewählt werden

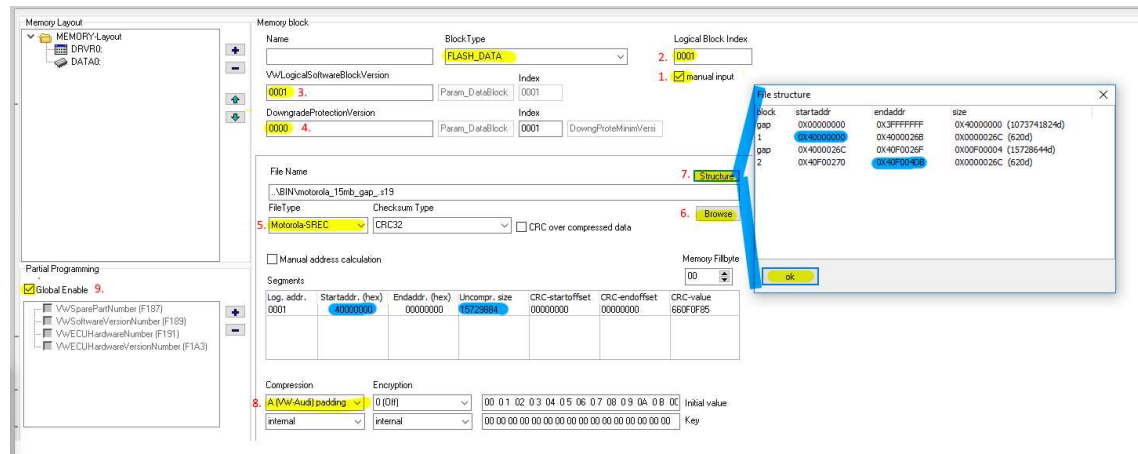


Kompression und Verschlüsselung kann je nach Bedarf zusätzlich eingestellt werden. Im Nachgang wird bei der Containererstellung auf Basis der RFS-Informationen/Blockversionen, die je logischen Block (TYP: FLASH_DATA) eingegeben wurden, automatisch ein RFS-Block berechnet und im Flashcontainer gespeichert.

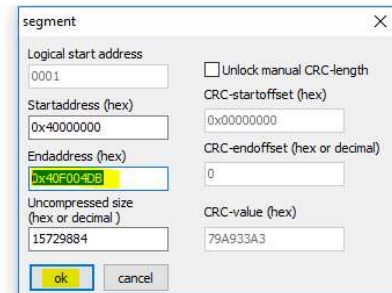
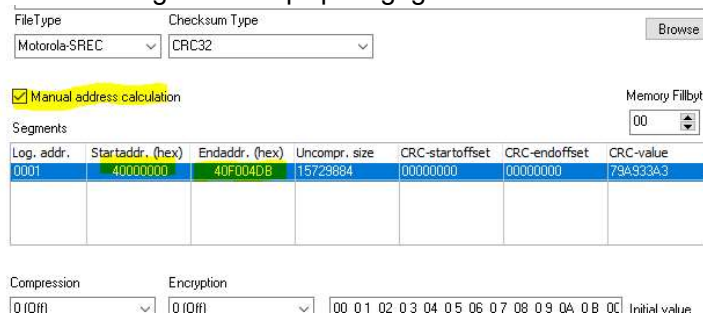
Eine Alternative manuelle Auswahl eines extern berechneten RFS-Blocks ist für zukünftige ODXCreate-Version vorgesehen.

28.5.5 Konfiguration des ersten Flashdatenblocks

Es wird eine Hexdatei mit GAPs als Flashblock ausgewählt. Erstellung eines Blocks vom Typ „FLASH-DATA“ wird übersprungen, der Ablauf ist analog zur Erstellung des RFS-Blocks.



1. Mittels Aktivieren der Checkbox „manual input“ wird das manuelle Eingeben des logischen Blockindex ermöglicht
2. Eingabe des logischen Blockindex mit dem Hex-Wert 0001
3. Auswahl der Block-Version – diese ist sowohl für partielle Programmierung, Rückflashschutz (A: 80126-A1261) und UNECE-SUMS benötigt.
4. Die Rückflashschutzversion ist in der frühen Entwicklungsphase mit 0000 zu Bedaten und ändert sich prinzipiell erst im Feldeinsatz, wenn eine schwerwiegende Sicherheitslücke entdeckt wurde.
5. Abhängig von den zu verwendenden Flashdaten muss der Dateityp ausgewählt werden – in diesem Beispiel Motorola-SREC
6. Danach kann über „Browse“ die entsprechende Datei im Dateisystem ausgewählt werden.
7. Mittels „Structure“ kann bei Hex-Dateien (Intel oder Motorola) die interne Struktur mit Freiräumen (GAPS) angezeigt werden. ODXCreate versucht automatisch die richtige Daten auszuwählen. Sollten andere Datenbereich gewünscht sein können diese über „Manual address calculation“ und Doppelklick auf z.B. die Startadresse des Blocks im folgenden PopUp eingegeben werden:

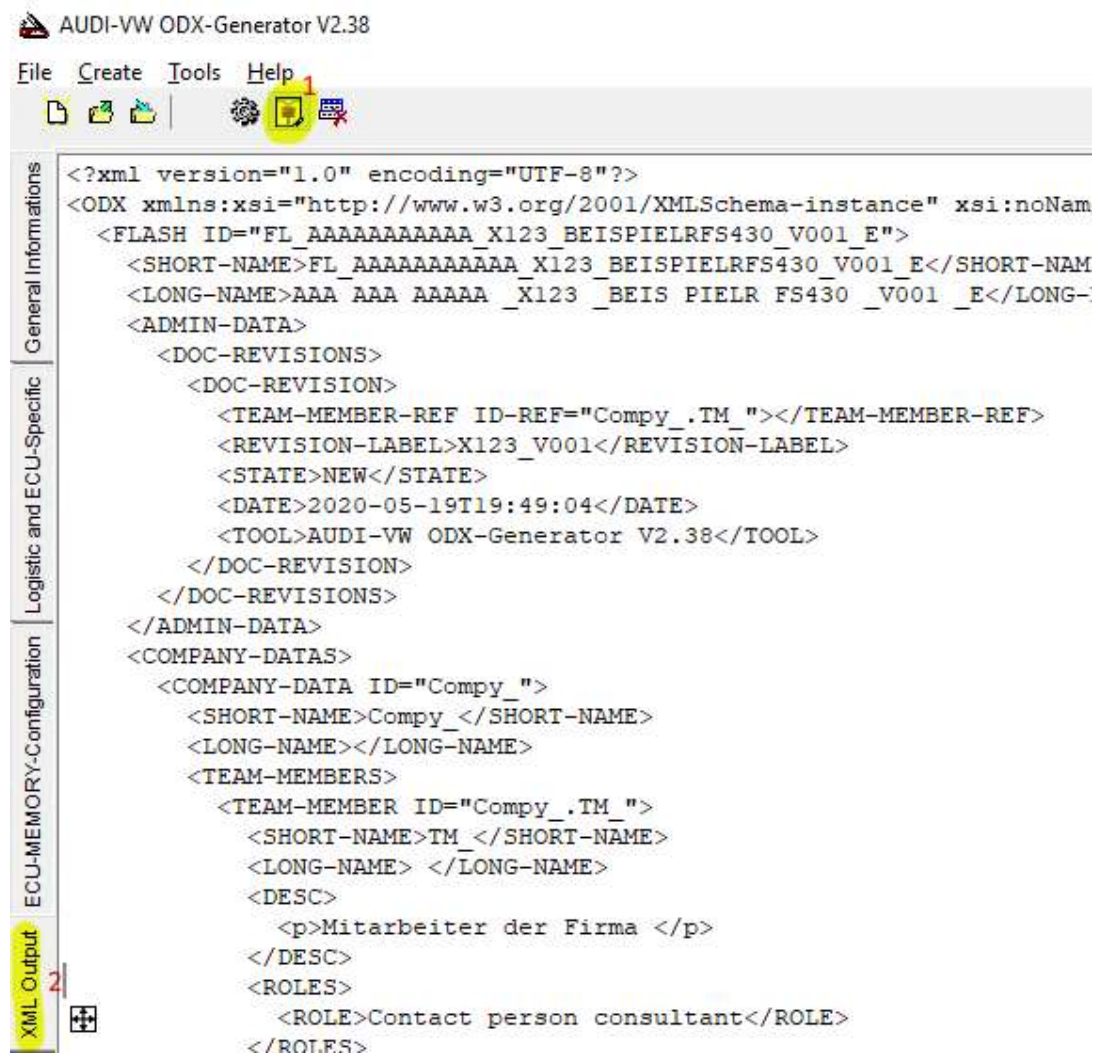


8. Typischerweise sind die Flashdaten zu komprimieren, das Standardverfahren ist hierbei LZSS welches mit „A *” ausgewählt werden kann.
9. Die Checkbox „Global Enable“ muss aktiviert werden

28.5.6 Erstellen des Flashcontainers – PDX

1. Mit Klick auf das „Packen“-Symbol wird ein PDX-Flashcontainer erstellt
2. Das Tool wechselt automatisch auf den Reiter „XML-Output“ dort werden die zentralen Daten des Flashcontainers zur Verifikation angezeigt.

Durch Wechsel auf den Reiter „Loginfo“ werden Details zur Flashcontainererstellung sowie der Pfad zum erstellten Flashcontainer angezeigt.



28.5.7 Inhalt des Flashcontainers

Die „.pdx“-Datei ist eine ZIP-Datei unter Verwendung der Dateiendung „.pdx“ – mit jedem Entzip-Tool kann das Archiv entpackt werden (ggf. muss zunächst nach .ZIP umbenannt werden)

20.347 FL_AAAAAAAAAAA_X123_BEISPIELRFS430_V001_E.pdx – Flashcontainer

⇒ Inhalt nach dem Entpacken:

530.720	2391A41F4D554C2B960E1286534DB0D567A78396A7B774DC7C37479280996AA800.bin	- Flashdaten
17	8620CE9B75C0D6C57462D0B85A0A9B6D2CD0183C63971BB7478BFCC80B74F8F600.bin	- Rückflashschutzblock
14.328	FL_AAAAAAAAAAA_X123_BEISPIELRFS430_V001_E.odx-f	- Flashcontainer-Metadaten
2.513	index.xml	- Beschreibung des PDX-Inhalts

Inhalt des RFS-Blocks:

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 | 0 07 52 46 53 56 31 2E 30 00 01 00 01 30 30 30 0.RFSV1.0....000
00000010 30                                     0
```

[A: 80126-A1261]

Der Pseudoflashtreiber muss folgende Felder enthalten:

Feld	Beschreibung	Länge [Bytes]	Wertebereich
Pseudoflashtreiber Identifikation (PTI)			
PTI _L	Länge der Pseudoflashtreiber Identifikation in Bytes	2	1 ... 65535
PTI _b	Inhalt der Pseudoflashtreiber Identifikation	PTI _L (1 ... 65535)	ASCII 32 ... 126 pro Byte
Blockversionstabelle (BVT)			
BVT _A	Anzahl der Logischen Blöcke im Flashcontainer	2	1 ... 65535
BID _i	ID des Logischen Blocks (i = 0, ..., BVT _A -1)	ALFID.BIDLEN (1 ... 15) (Bit 0...3 von ALFID)	0 ... 2 ^{(8*BIDLEN)-1}
RfsV _{PFT(i)}	Rückflashschutz-Versionsnummer des Logischen Blocks BID _i (i = 0, ..., BVT _A -1) im Flashcontainer	4	ASCII 0x30...0x39 pro Byte

Tabelle 89: Felder des Pseudoflashtreibers

Das Beispiel ist zur Veranschaulichung in der Toolauslieferung mit folgenden Dateien hinterlegt:

- Flashdaten:
BIN\motorola_15mb_gap_.s19
- Mit ODXCreate erstellte Konfiguration:
\Config\RFS_Beispielcontainer.ocnf
- Erstellter Flashcontainer:
\Container\FL_AAAAAAAAAAA_X123_BEISPIELRFS430_V001_E.pdx