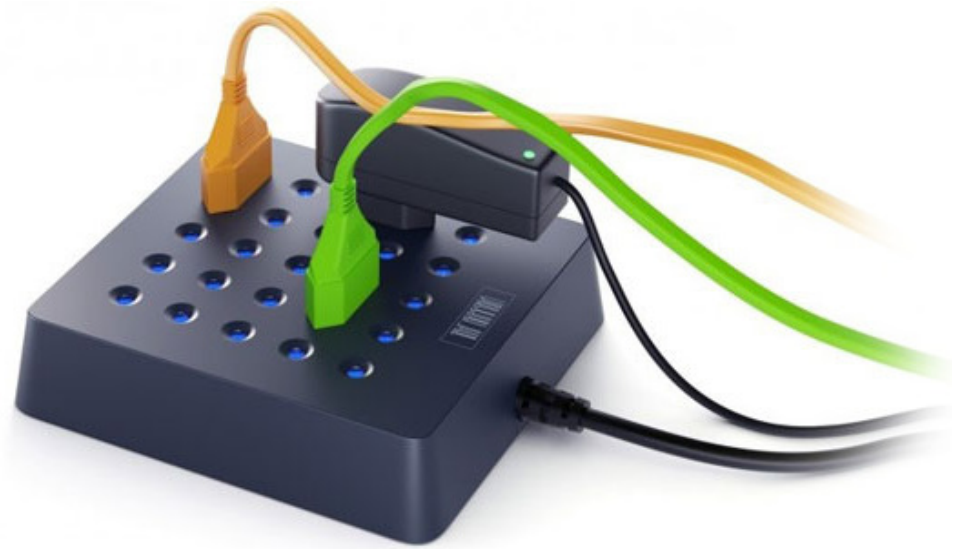
Dynamisches Eisenbahn System Modell
Modèle dynamique d'un système ferroviaire
Dynamic model of a railway system

Bern, 06.02.2013

DESM Middleware



Spezifikation v0.12

Autor & Referenzperson
Sebastian Straube

sebastian.straube@desm.ch
+41 (0) 79 4452 127
http://www.desm.ch

1 Inhaltsverzeichnis

2 Kontaktliste

Name	Verankerung	Kontakt	Aufgaben / Hintergrund DESM
Jürg Suter	Bundesamt für Verkehr	+41 31 322 5765 (G) +41 31 931 3662 (P)	Projektleiter DESM
Sebastian Straube	Lufthansa Systems	+41 79 445 2127	Standardisierung Middleware für verknüpfte Systemkomponenten
Fabian Riesen	Cisco Systems	+41 79 448 4700	Dispatcher Implementierung Re 4/4
Dr. Hansjürg Rohrer	Fachhochschule Biel	+41 032 321 63 73	Projektleiter Simulation LOCSIM

(P) Privat
(G) Geschäft

3 Dokumentenhistorie

Version	Datum	Name	Änderung
1.12	06.02.2012	Sebastian Straube	Hinzufügen Kapitel 2 und 3 Entfernen von Kapitel „Kontaktpersonen“ **Fehler! Verweisquelle konnte nicht gefunden werden.** „define_trainposition“ Beschreibung aktualisiert

4 Ausgangslage

Ausgangsprojekt
Innerhalb der Promotionsarbeit von Jürg Suter wurde ein Forschungslabor aufgebaut, welches zu einem offiziellen Verein mit dem Namen DESM institutionalisiert wird. Das Forschungslabor besteht momentan aus zwei Fahrsimulatoren der Loktypen Re 4/4 und Re 460. Der DESM Simulator des Loktyps Re 460 ist gegenwärtig in der Schweiz der einzige Vollsimulator dieser Art.. Für weitere Details zu der Promotionsarbeit verweise ich auf die Vereinsseite: http://www.desm.ch .

Unterprojekte
Alle Untersuchungen der Forschungsarbeit beziehen sich auf qualitative und quantitative Analysen und der Erarbeitung von neuen Methoden, die im Forschungslabor durchgeführt werden. Dafür ist es nötig ein System aufzubauen, in dem diese Methoden erarbeitet und die Ergebnisse wissenschaftlich analysiert werden können. Sie finden weitere Details auf die Vereinsseite: http://www.desm.ch.

Systemkomponenten Re 4/4
Der Simulator der Re 4/4 beinhaltet verschiedene Systemkomponenten, um dem Lokführer ein möglichst realitätsgetreues Interface und „feeling“ zu bieten. Die Führerstandkabine enthält alle Bedienelemente der Lok vom Typ Re 4/4. Ausserdem wird das Bremssystem mit Druckluft betrieben, so dass realitätsnahe Geräusche und mechanische Bewegungen zu hören sind. Für die Darstellung der Umwelt wird das Simulationsprogramm LOCSIM von der Berner Fachhochschule Biel eingesetzt. Diese Simulationssoftware verfolgt einen videobasierten Ansatz, um die Umwelt für den Lokführer realitätsgetreu abzubilden. Dabei wird für die Simulation eine bearbeitete Videoaufnahme abgespielt. Des Weiteren werden alle benötigten Simulationsvariablen durch die Simulationssoftware berechnet. Für weitere Details zu dem LOCSIM Simulator verweise ich auf die LOCSIM: http://www.locsim.ch

Systemansatz
Für die Forschungsarbeit wird es nötig sein, die Kommunikation zwischen bestimmten Systemelementen zu ermöglichen. Das heisst, es soll nicht nur der Lokführer in die Simulation eingebunden werden, sondern auch der Zugverkehrsleiter in der Betriebszentrale sowie die dazugehörigen Stellwerke auf der simulierten Strecke.

5 Ziele

Das Projekt Middleware verfolgt das langfristige Ziel, verschiedene Systemkomponenten einer Simulation über eine Kommunikationsarchitektur miteinander zu verbinden und somit die Integration einer Betriebszentrale in Bezug auf den Schienenverkehr zu ermöglichen. Dieses Vorhaben wird innerhalb des Vereins DESM umgesetzt. Die einzelnen Komponenten der Architektur sollen möglichst modular aufgebaut sein, um die Anwendung und Integration verschiedener Komponenten kurzfristig zu ermöglichen. Die Kommunikation soll anhand eines anerkannten Industriestandards umgesetzt werden.

6 ToDo

- ✓ Analyse des bisherigen Simulators Re 420 (Fabian Riesen, Thomas Schneider)
- ✓ Integration LokSim 3D (Fabian Riesen)
- ✓ Integration Locsim FHS Biel (FHS Biel, Fabian Riesen)
- ✓ Aufbereitung der Infrastruktur in der Umgebung von Obermatt auf der Strecke Bern - Luzern (FHS Biel, Jürg Suter)
- ✓ Modellierung der Aussenanlagen in vergangene Epochen (Jürg Suter)
- ✓ Schnittstelle zu Aussenanlagen (Signale) definieren (DLL, TCP, IP)
- Zugriff auf Signale im Locsim-Gelände
- Steuerung und Schnittstelle zu Stellwerken (Sebastian Straube, Jürg Suter)
- Zusammenführung Loksimulation und Stellwerksteuerungen (DESM)

7 Locsim - Interlocking Middleware

Die Middleware Software ist so modular aufgebaut, dass dort verschiedenste Systemkomponenten angeschlossen werden können. Allerdings wird für jede Komponente eine bestimmte Konfiguration für den Datenverkehr benötigt. In diesem Fall umfasst das System die Kommunikationsverbindung von folgenden Systemkomponenten:

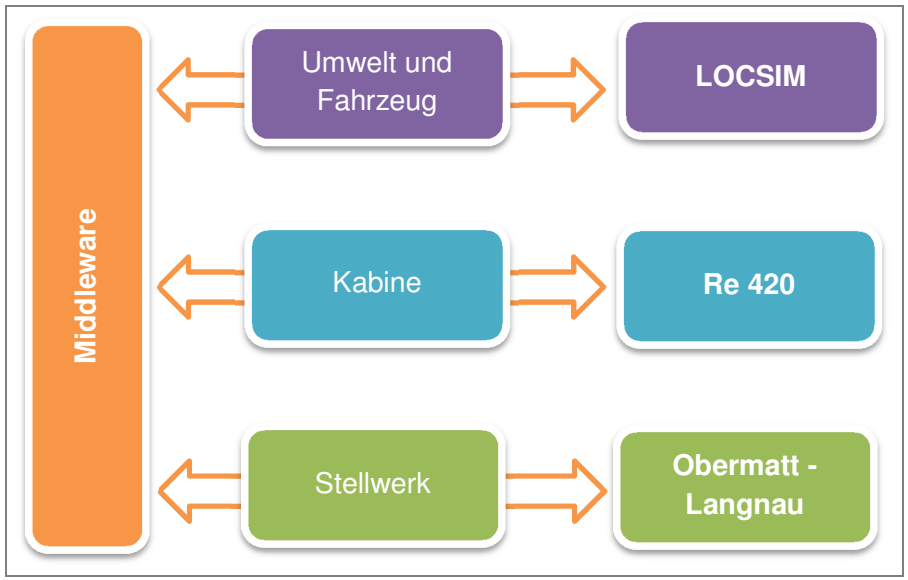


Abb. 1: Wichtigste Elemente der Zusammenführung von Loksimulatoren und Stellwerklogik am Beispiel des Simulators der Re 4/4 und des Stellwerks Obermatt.

7.1 ToDo Ressourcen

- ✓ PC und Projektion für den Simulator 420
- ✓ Software für Loksim 3D
- ✓ Server für Locsim der FHS Biel
- ✓ Software mit Lizenz für Locsim der FHS Biel und Gelände der Strecke Bern – Luzern

7.2 ToDo Schnittstelle

- ✓ DLL Interface von Middleware nach Locsim spezifizieren
- ✓ Interface in unmanaged C++ implementieren
- ✓ Umsetzung erster Server Tests für den Datenaustauch der Middleware
- ✓ Testumgebung für Collaboration einrichten (GIT)
- ✓ Server und Client in Testumgebung einrichten
- ✓ Allgemeines Interface Format für Datenaustausch finden: {XML, OWL, railML}.
- ✓ Locsim verantwortlichen Herrn Dr. Rohrer informieren und DLL übergeben
- Threadsafe Implementierung des Server und des Clients in DLL
- Synchronisationsstruktur für Asynchronen Datenaustausch aufbauen
- XML Dateistruktur für die Infrastrukturdaten in dem Anwendungsfall Signal-Obermatt Langnau aufbauen
- XML Reader und Parser in die DLL implementieren
- Integration Stellwerklogik anhand einer RuleEngine
- Test und finale Zusammenführung mit den Komponenten des LOCSIM

7.3 Systemkommunikation (TCP/IP Protokoll)

Die Verbindung zwischen der Middleware und dem Locsim wird per Ethernet hergestellt. Aus den gegebenen Anforderungen wird eine modulare Lösung und damit die Wiederverwendbarkeit der DLL angestrebt. Daher wird in der DLL ein Server und ein Proxy für die Datenübertragung bereitgestellt. Dadurch wird es ermöglicht, auf beiden Systemen mit der gleichen DLL eine Verbindung über das Netzwerk herzustellen.

7.4 Kommunikationsarchitektur

Es stehen verschiedene Techniken zur Verfügung, die Daten über eine Ethernet Verbindung auszutauschen.

Es wurden folgende Techniken evaluiert.

Protokollname	Beschreibung	Vorteil	Nachteil
Winsocket (TCP)	Eine „Winsocket“ Client-Server Übertragung. Das Übertragungsprotokoll soll lediglich ByteStreams empfangen und senden können, weil nur XML Daten transferiert werden sollen und keine RAW Datentypen. Dafür muss ein Magic Packet (Initialisierung) definiert werden.	• schnell • Abstraktion durch XML Struktur	• Integration kompliziert • eigenes Übertragungsprotokoll nötig
HTTP (high level) & Json	verschiedene libraries sind u.A. boostASIO, libHTTP, libEvent, libOV	• High Level Integration • ermöglich Steuerung über Webinterface	• Overhead gross • keine permanente Verbindung • zwei Webserver
Open Sound Protokoll		• library Unterstützung sehr umfangreich	• proprietäre Implementierung

7.5 Übertragungsprotokoll

Es ist je nach gewählter Technik für die Umsetzung der Ethernet Schnittstelle ein sehr hoher Aufwand für die Entwicklung eines Protokolls nötig. Damit der Aufwand auf jeder Ebene gering gehalten wird, gibt es ein Dateiübertragungsprotokoll. Der Vorteil dieser Lösung ist ein geringerer Synchronisationsaufwand und die Implementierung und Nutzung von später benötigten Techniken. Für die Übertragung wird eine XML strukturierte Datei genutzt. Die Struktur stützt sich auf das railML Datenformat. Dafür wird ein Reader und Writer in die DLL implementiert. Das hat den Vorteil, dass diese Funktion von der Middleware und vom LOCSIM genutzt werden können.

7.6 Synchronisation der Systemkomponenten

Die Datenübertragung von LOCSIM zur Middleware und umgekehrt findet asynchron statt. Die Daten können vom LOCSIM sowie von der Middleware zu einem beliebigen Zeitpunkt übertragen und gelesen werden. Dabei werden die Daten in der DLL zwischengelagert. Die Synchronisation wird über ein Status Flag gesteuert. Sobald Daten im DLL Cache verändert werden, wird das Status Flag auf „dirty“ gesetzt. Ausserdem wird durch eine „lock_id“ die Synchronisationsrichtung gesteuert.

7.7 XML Format (Datenformat railML)

Es wird für die Datenhaltung und Datenübertragung zwischen den Systemkomponenten ein standardisiertes gültiges Format benötigt. Dadurch wird die Anordnung von späteren Versuchsaufbauten erleichtert. Mit dieser Datenbasis werden Methoden angewandt, um verschiedene Stellwerktypen und Sicherungsanlagen standardmässig integrieren zu können. Das railML Format ist bereits in der Wirtschaft etabliert und findet allgemeine Anerkennung. Für die Modellierung der Stellwerklogik und der Infrastrukturdaten wird daher das Datenformat railML verwendet. Für weitere Informationen verweise ich auf die Seite: http://www.railml.org.

7.8 DLL

Die DLL wurde als „unmanaged“ Code in C++ geschrieben. Zum Laden der DLL wird kein MFC benötigt. Auf die Definition als COM Komponente wurde aus Vereinfachungsgründen verzichtet. Die Daten in der DLL werden in einem Cache gehalten und von dort weitergegeben oder abgeholt.

7.8.1 Funktionen in der DLL

Der Zeitpunkt des Zugriffs auf bestimmte DLL Funktionen ist durch gewisse Anwendungsstatus des Locsim gegeben.

define: locsim => stellwerk (cached in DLL)
get:stellwerk (cached in DLL) => locsim

7.8.1.1 DLL laden (initialize meta infos)

Funktionsbeschreibung	• Beim Laden der DLL library, wird der Netzwerkserver durch diese Funktion gestartet.	
seit Version	0.1	
Signatur	înt stw_onLoadLibrary()	
Attribute		
Rückgabepointer		

Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0001

Funktionsbeschreibung	• Wenn der LOCSIM die Simulation stoppt wird die DLL entladen und der der Netzwerkserver durch diese Funktion geschlossen und gestoppt. Damit wird die Verbindung zwischen Server und Client ordnungsgemäss geschlossen.	
seit Version	0.1	
Signatur	int stw_onUnloadLibrary()	
Attribute		
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0002

~~**Funktionsbeschreibung**~~	~~• Es braucht eine zusätzliche Funktion stw_reset(), ohne Parameter, welche beim Neuladen einer Strecke alle Daten in der Stellwerk-dll löscht.~~	
~~**seit Version**~~		
~~**Signatur**~~	~~int stw_reset ()~~	
~~**Attribute**~~		
~~**Rückgabepointer**~~		
~~**Rückgabewert (OK)**~~	~~0~~	
~~**Rückgabewert (ERROR)**~~	~~1~~	~~err-stw_000.000.000.000~~

7.8.1.2 DLL ist geladen

Funktionsbeschreibung	• Behandlung von Versionskonflikten der DLL.	
seit Version	0.1	
Signatur	const char* stw_getVersion()	
Attribute		
Rückgabepointer	const char*	
Rückgabewert (OK)	0.1	i.o.
Rückgabewert (ERROR)	„“	LocsimDesmMiddleware_ error_0004

7.8.1.3 Simulation Transition (Start, Aufbau und Ende)

Funktionsbeschreibung	• markiert Simulation Start • impliziert das Event „Strecke neu laden“ • es werden ab jetzt mögliche „defines“ zum Verarbeiten erwartet	
seit Version	0.1	
Signatur	int stw_init()	
Attributbeschreibung		
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0005

Funktionsbeschreibung	• markiert das Ende der Simulation • erwartet danach „gets“ Funktionen oder die „define_trainposition“ Funktion	
seit Version	0.1	
Signatur	int stw_ready()	
Attributbeschreibung		
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0006

Funktionsbeschreibung	• Definition von zu Hauptgleis parallelen Gleisabschnitten • Das Hauptgleis ist die Standardfahrstrasse der Simulation.	
seit Version	0.1	
Signatur	int define_track (int gleisId, double von, double bis, float abstand, char* name)	
Attributbeschreibung	gleisId von bis abstand	locsim-interne ID als Meterangabe als Meterangabe Abstand von Hauptgleis(Meter) - pos. Rechts, neg. Links

	name	Gleisnummer gemäss Sicherungsanlage, z.B. A1 (kann auch leer sein)
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0007

Funktionsbeschreibung	• Definition von Verbindungen zwischen parallelen Gleisabschnitten • weiche1_id, weiche2_id = ID der Weiche (eindeutige locsim-interne Nummerierung), wenn 0: keine Weiche (wenn z.B. connection2 in gleis2 übergeht) • pro Längsposition dürfen max. 20 verschiedene Gleise definiert sein (ID= 1…20)	
seit Version	0.1	
Signatur	int define_trackConnection (int gleis1, int gleis2, double von, double bis, char* name, int weiche1_id, int weiche2_id)	
Attributbeschreibung	gleis1 gleis2 von bis name weiche1_id weiche2_id	von gleis nach gleis Positionsangabe der Gleise als Meterangabe Positionsangabe der Gleise als Meterangabe Gleisnummer gemäss Sicherungsanlage, z.B. A1 (kann auch leer sein) ID der ersten Weiche ID der zweiten Weiche
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0008

Funktionsbeschreibung	• wird pro Signal aufgerufen	
seit Version	0.1	
Signatur	int define_signal (int signal_id, int gleis_id, double position, int typ, float hoehe, float distanz, char* name)	
Attributbeschreibung	signal_id gleis_id position typ	eindeutige Nummer (willkürlich) interne Gleisnummer Geoposition des Signals der Typ des Signals

	hoehe distanz name	die Höhe des Signals die Distanz des Signals zum Gleis name des Signals
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0009

Funktionsbeschreibung	• definiert eine Balise auf einem bestimmten Gleis an einer bestimmten Position	
Signatur	int define_balise (int gleis_id, double position, int balise_id)	
Attributbeschreibung	gleis_id position balise_id	eindeutige Gleis ID eindeutige Nummer (willkürlich) Wirkungsrichtung, 1=vorwärts, -1=rückwärts
Rückgabewert (Attribut)		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0010

Funktionsbeschreibung	• Ist in den locsim-Streckendaten bis jetzt nicht drin, könnte aber dazugefügt werden	
seit Version	0.1	
Signatur	int define_isolierstoss (int gleis, double position)	
Attributbeschreibung		
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0011

7.8.1.4 Simulation gestartet

Funktionsbeschreibung	• Funktion: Locsim fragt DLL, welche events vom Stellwerk vorhanden sind (DLL cached) • jedes Event führt zum Aufruf einer der nachstehenden Funktionen • stw_get_event(3, typeList(1,1,2), idList(63, 32 , 765))
seit Version	0.1

Signatur	int stw_getEvents(int* number, int** type_list, int** id_list)	
Attributbeschreibung	number type_list id_list	Anzahl Events (Arraygrösse): 0=nichts types = 1 (Signal), 2 (Balise), 3 (Weiche) as Array list id = id from type as Array list
Rückgabepointer	int* number	als Integer Pointer
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0012

Funktionsbeschreibung	• gibt die Stellung eines bestimmten Signals zurück	
seit Version	0.1	
Signatur	int stw_getSignal (int signal_id, int* stellung)	
Attributbeschreibung	signal_id stellung	ID des Signals Stellung gemäss help\locsimmanualsignal-d.htm
Rückgabepointer	int* stellung	als Integer Pointer
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0013

Funktionsbeschreibung	• gibt eine Stellung und das Protokoll einer bestimmten Balise zurück	
seit Version	0.1	
Signatur	int stw_getBalise (int balise_id, int* stellung, char** protokoll)	
Attributbeschreibung	balise_id	
	stellung	gemäss locsimmanualsignal-d.htm, „Zugbeeinflussung durch Signale“, „v“, ausser -3001…-7000; wenn=-9998 => protokoll
	protokoll	gemäss help\zugsicherungen.txt, wenn stellung ungleich -9998: leer
Rückgabepointer	int stellung char** protokoll	ist rückgabewert, als Integer Pointer ist rückgabewert, als char Doppelpointer (String Array)
Rückgabewert (OK)	0	
Rückgabewert (ERROR)		LocsimDesmMiddleware_ error_0014

Funktionsbeschreibung	• gibt die Stellung einer bestimmten Weiche zurück	
seit Version	0.1	
Signatur	int stw_getWeiche (int weiche_id, int* gleis_id)	
Attributbeschreibung	weiche_id gleis_id	ID gemäss define_track_connection Gleisnummer der Stellung (im stumpfen Bereich)
Rückgabewert (Attribut)	int* gleis_id	ist rückgabewert, als Integer Pointer
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0015

Funktionsbeschreibung	• Übergibt pos1 – gleis1 – pos2 – gleis2 – pos3 - … des Zuges train an stellwerk.dll wenn die Zugspitze oder der Zugschluss einen Isolierstoss überfährt • Was bedeutet in diesem Fall die Position? • Anzahl pos = Anzahl gleis +1	
seit Version	0.1	
Signatur	int define_trainposition(int train, int direction, double** positionList, int** gleisList)	
Attributbeschreibung	direction train positionList gleisList	1=vorwärts, -1=rückwärts 0=simulierter Zug, 1 und weitere=andere Züge Position „von, bis“ als Objekt von „define_track“ Gleisnummern „gleisId“ als Objekt von „define_track“
Rückgabepointer		
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0016

Funktionsbeschreibung	• Speicher Allocation and Deallocation muss in DLL und von Locsim aufgerufen werden, sonst tritt ein memory-leak auf.	
seit Version	0.1	
Signatur	int stw_deallocate(void*)	
Attributbeschreibung		

Rückgabepointer	void*	pointer auf memory
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0017

Funktionsbeschreibung	• Speicher Allocation and Deallocation muss in DLL und von Locsim aufgerufen werden, sonst tritt ein memory-leak auf.	
seit Version	0.1	
Signatur	int stw_deallocate_array(void*)	
Attributbeschreibung		
Rückgabepointer	void*	pointer auf memory
Rückgabewert (OK)	0	
Rückgabewert (ERROR)	1	LocsimDesmMiddleware_ error_0018

8 Installation Middleware

8.1 .NET Framework