Institut für Fertigungstechnik und Werkzeugmaschinen

Forschungsinstitut

Kennwort/Akronym: VDWFI-068 / umati2MTConnect2umati

Abschlussbericht

Integration von MTConnect in das umati Ökosystem

Laufzeit: 01.02.2025 bis 30.09.2025

Projektkonsortium:

- VDW - Forschungsinstitut e.V. (Herr Dr.-Ing. Alexander Broos)
- DMG MORI Digital GmbH (Herr Dennis Do-Khac)
- Yamazaki Mazak Deutschland GmbH (Herr Mathias Dehn)
- IFW Hannover (Herr Aleks Arzer)

1 Beschreibung der durchgeführten Tätigkeiten

1.1 AP 1: Konzepterstellung - Integration von MTConnect in das umati Ökosystem

In AP 1 wurde ein Konzept zur Einbindung von Daten aus einer MTConnect-Schnittstelle in eine umati OPC UA-Schnittstelle entwickelt. Hierzu wurden zunächst die relevanten Companion Specifications (CS) OPC UA for Machinery, OPC UA for Machine Tools (UA4MT) sowie aufbauende CS eingehend analysiert [VDM24, VDM25]. OPC UA Server, die die genannten CS verwenden, werden im Nachfolgenden mit dem Begriff „OPC UA4MT“ bezeichnet. Parallel erfolgte eine detaillierte Untersuchung der Variablen sowie der Modellierungsvorgaben des MTConnect-Standards (Spezifikationen Version 2.2.0 (2023) Part 2.0 und Part 3.0 [MTC23a, MTC23b]) sowie der CS „OPC 30070-1 OPC UA for MTConnect“ [MTC19].

Ergänzend zur theoretischen Analyse wurden OPC UA4MT- und MTConnect-Schnittstellen experimentell untersucht. Für OPC UA4MT stehen am IFW vier DMG Mori-Maschinen sowie der auf GitHub bereitgestellte umati Sample Server [CHR23] zur Verfügung. Für MTConnect wurden die von Mazak veröffentlichten Beispielserver für Version 1.3 bis 2.0 analysiert [MAZ22]. In Abstimmung mit Mazak wurde die weitere Arbeit auf Version 1.3 fokussiert, da nur diese Version in den in der freien Wirtschaft verfügbaren Maschinen implementiert ist. Zusätzlich stellte Mazak den VPN-Zugang zu einer realen MTConnect-Maschine bereit. Ebenfalls wurden die MTConnect-Implementierungen der DMG-Maschinen am IFW untersucht. Die verfügbaren Variablen auf beiden Schnittstellen können nach Bild 1 entsprechend Ihrer Verfügbarkeit sowie ihrer Modellierungsvorgabe (*mandatory*, *optional*) eingeteilt werden.

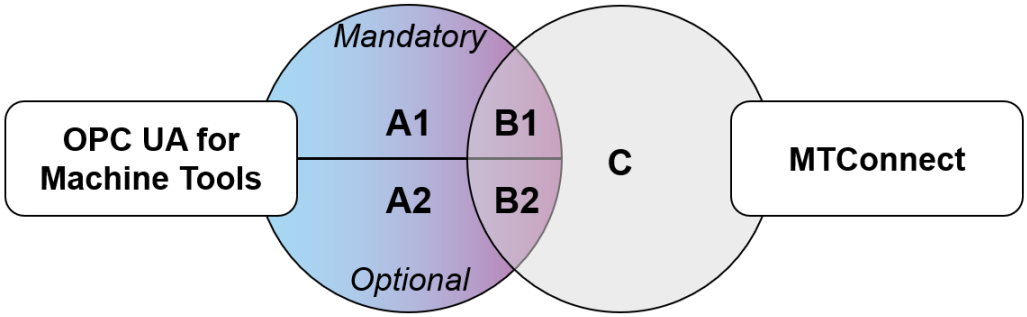


Bild 1: Aufteilung der verfügbaren Variablen

Auf Basis dieser Analysen wurde geprüft, welche Variablen mit beiden Schnittstellen verfügbar sind bzw. fehlen und wie MTConnect-Daten in die umati-Informationsmodelle integriert werden können. Hierfür wurde eine Mapping-Tabelle erstellt, die für jede Variable Variablenname, Datentyp, Datenpfad und Modellierungsvorgabe enthält. Fehlende Variablen wurden gekennzeichnet. Für die MTConnect-Implementierungen von DMG und Mazak wurde jeweils ein Informationsmodell mit *UaModeller* auf Grundlage der *OPC UA for Machine Tools* CS erstellt. MTConnect-Variablen ohne direkte Entsprechung (C) wurden nach fachlicher Bewertung durch VDW und IFW in die Struktur aufgenommen. Zusätzlich wurden diese in der Mapping-Tabelle auf einer Skala von 1 bis 5 hinsichtlich ihres Potenzials für künftige CS-Versionen bewertet. Mit Abschluss von AP 1 liegen zwei *UaModeller*-Projektdateien, die zugehörigen *NodeSet2.xml*-Informationsmodelle sowie eine vollständige Mapping-Tabelle zwischen *OPC UA for Machine Tools* und MTConnect vor.

1.2 AP 2: Softwareentwicklung: Adapter MTConnect nach OPC UA4MT

In AP 2 wurde eine prototypische Software zur Konvertierung von Daten aus MTConnect-Schnittstellen in OPC UA4MT-Schnittstellen entwickelt (*mtc2umati*). Die Implementierung erfolgte als .NET 9.0-Anwendung unter Verwendung der *UA-.NETStandard*-Bibliothek (OPCFoundation.NetStandard.Opc.Ua, Version 1.5.375.457), da diese den Import des aktuellsten *OPC UA for Machine Tools* Informationsmodells (Stand 18.08.2025) unterstützt. Die technische Umsetzung ist in Anhang 4.1 und 4.2 dargestellt.

Zunächst erfolgte die Einarbeitung in die *UA-.NETStandard*-Bibliothek, um die Grundlagen für die Entwicklung zentraler Adapterfunktionen zu schaffen. Dabei wurden Softwarefunktionen zum Import der in AP 1 erstellten Informationsmodelle sowie der zugehörigen Excel-Mappingtabelle (mapping.xlsx) entwickelt. Das Setzen erforderlicher Parameter (z. B. IP-Adresse und Port des MTConnect Quellservers sowie Pfad der Mappingtabelle) wurde über eine Konfigurationsdatei (config.json) umgesetzt.

Bei Ausführung der Adapter-Software wird zunächst auf Basis der importierten Informationsmodelle ein OPC UA-Server aufgebaut und anschließend eine Verbindung zum MTConnect-Server hergestellt. Der Maschinenname ist nicht als reguläre Variable auf der MTConnect-Schnittstelle verfügbar. Stattdessen konnte gezeigt werden, dass er automatisiert aus dem *DeviceStream*-Namen der MTConnect-XML extrahiert werden kann. Hierfür wurde eine Funktion entwickelt, die den entsprechenden Maschinenordner im OPC UA-Server automatisch umbenennt. Die Variablen werden gemäß der Mappingtabelle ausgelesen und asynchron auf den OPC UA-Server geschrieben, wobei ausschließlich geänderte Werte übertragen werden. Für ausgewählte Variablen wurden spezifische Konvertierungsregeln konzipiert und implementiert.

Zur Ergänzung fehlender *mandatory*-Daten (A1) wurde eine Funktion zum Einlesen statischer Werte entwickelt. Diese Werte können direkt in der Mappingtabelle definiert werden und werden in gleicher Weise auf den OPC UA-Server geschrieben. Für den Adapter stehen drei Betriebsmodi zur Verfügung:

1. Übertragung ausschließlich standardkonformer A- und B-Variablen,
2. Integration zusätzlicher C-Variablen in die OPC UA-Struktur,
3. Zusammenführung der C-Variablen in einem separaten Ordner „MTConnect“.

Der Adapter ist in der Lage, Ordner und Knoten im OPC UA-Server dynamisch anzulegen. Aufgrund der statischen MTConnect-Implementierungen ist diese Funktion jedoch in der Praxis nicht erforderlich. Die Praxistauglichkeit wurde durch die Anbindung der Referenzmaschinen an den umati-Showcase des VDW über das umatiGateway (Stand 26.06.2025) nachgewiesen (Anhang 4.3). Zusätzlich liegt eine *Docker-Compose*-Implementierung bei, die eine automatisierte Verbindung zu einem Mazak-MTConnect-Referenzserver herstellt und sich mit einem einzigen Befehl ausführen lässt.

1.3 AP 3: Softwareentwicklung: Adapter OPC UA4MT nach MTConnect

In AP 3 wurde eine Software zur Konvertierung von Daten aus einer OPC UA4MT-Schnittstelle in das MTConnect-Format entwickelt (umati2mtc). Die Implementierung erfolgte als Python-Anwendung (3.11) in Kombination mit dem offiziellen C++ Agent des MTConnect Institute (Version 2.5.0.11). Die technische Umsetzung ist in Anhang 4.4 und 4.5 dargestellt.

Zunächst wurde die vom MTConnect Institute bereitgestellte technische Dokumentation zur vorgesehenen Umsetzung von MTConnect-Schnittstellen analysiert [MTC25]. Diese sieht die Umsetzung einer Adapter-Agent-Architektur unter Verwendung des offiziellen C++ Agenten vor. Der Datenaustausch erfolgt über das SHDR-Protokoll (Simple Hierarchical Data Representation) und erfordert den Aufbau eines SHDR-Servers im Adapter. Das Informationsmodell des Agenten definiert die Struktur der XML-Dateien, bestehend aus *Device*- und *Component*-Streams sowie den Datentypen *Samples*, *Events* und *Conditions*. Jede Variable wird durch einen Namen (*SpecName*) eindeutig gekennzeichnet. Zusätzlich werden Parameter des Agenten (z. B. IP-Adresse des SHDR-Servers) in der Konfigurationsdatei *agent.dock* gesetzt.

Mit dieser Ausgangslage wurde der Adapter entwickelt und implementiert. Um eine hohe Wartbarkeit durch die Verwendung von Standardkomponenten zu erzielen, wurde in Abstimmung mit dem VDW das *umatiGateway* als Datenquelle genutzt. Dieses stellt bereits Funktionen bereit, um sich mit OPC UA4MT-Servern zu verbinden und die erfassten Daten gebündelt im JSON-Format über MQTT an einen Broker zu übertragen. Der entwickelte Adapter baut einen MQTT-Client auf, verbindet sich mit einem konfigurierten Broker und liest die Nachrichten an einem definierten „Topic“ zyklisch aus. Auf Grundlage der Mappingtabelle werden die Werte in den JSON-Nachrichten ausgelesen, bedarfsgerecht konvertiert und im Adapter zwischengespeichert. Parallel dazu wird ein SHDR-Server aufgebaut, der als Schnittstelle zwischen Adapter und C++ Agent fungiert.

Es wurde untersucht, wie die Daten möglichst effizient über SHDR übertragen werden können. Dabei wurde festgestellt, dass das Protokoll die gleichzeitige Übertragung beliebig vieler Variablen unterstützt. Eine Funktion wurde entwickelt, die aus allen verfügbaren Daten eine einzelne Nachricht im SHDR-Format erstellt und diese an den SHDR-Server überträgt.

SHDR-Nachricht = {Zeitstempel} | *SpecName1* | *Wert1* | *SpecName2* | *Wert2* | …

Der C++ Agent greift zyklisch auf diesen Server zu, integriert die Daten in die XML-Struktur und stellt sie im eigenen Dashboard live dar. Damit konnte der korrekte Datenaustausch verifiziert werden. Zusätzlich wurde die Funktionalität des Adapters mit der von Mazak bereitgestellten MTConnect-Dashboard-Software *Smooth Monitor AX* validiert (Anhang 4.6). Hierfür war es erforderlich, die C-Variable *Availability* gesondert auf „AVAILABLE“ zu setzen. Inzwischen ist Mazak intern von *Smooth Monitor Ax* auf die Dashboard-Software *iConnect* umgestiegen, welche MTConnect nicht mehr unterstützt. Für das Teilprojekt *umati2mtc* wurde zusätzlich ein Demo-Modus über eine *Docker-Compose*-Implementierung samt Simulationsumgebung umgesetzt (siehe Anhang 4.7).

1.4 Einbindung weiterer Maschinen

Mit *umatiConnect* stehen Beispielimplementierungen für DMG- und Mazak-Maschinen bereit. Neue Maschinen lassen sich mit geringem Anpassungsaufwand integrieren. Im Folgenden ist beschrieben, wie die vorhandenen Adapter für zusätzliche Maschinen genutzt und Variablen hinzugefügt oder geändert werden können. Die Schritte beziehen sich namentlich auf die im Repository abgelegten Softwaremodule und Dateien.

1.4.1 Adaption der mtc2umati-Software (Anbindung einer MT-Connect Maschine)

Um die Software optimal an die vorliegende, neue MT-Connect Maschine anzupassen, kann es erforderlich sein, die bereitgestellten *NodeSet2*-Informationsmodelle zu verändern oder ein neues Informationsmodell entsprechend der individuellen MTConnect-Implementierung zu erstellen. Für das Informationsmodell sollte dabei ein neuer und eindeutiger *Address Space* definiert werden. Zur Erstellung des Modells kann etwa *UaModeller* eingesetzt werden. Die neue Modelldatei ist anschließend im Verzeichnis *mtc2umati/Nodesets* abzulegen und über den *umatiNodeManager* zu importieren. Danach sollte die bestehende *mapping.xlsx*-Datei aktualisiert oder durch eine neue Datei ersetzt werden, um die Variablenzuordnung anzupassen. In der Konfigurationsdatei *mtc2umati/config.json* sind anschließend sowohl die neue Mapping-Datei als auch die Verbindungsdaten zum MTConnect-Server einzutragen. Falls für bestimmte Variablen besondere Umwandlungen notwendig sind, sind diese im Modul *DataConversion* zu implementieren.

1.4.2 Adaption der umati2mtc-Software (Anbindung einer OPC UA4MT-Maschine)

Die Software ist erweiterbar und kann zur Unterstützung von spezifischen MTConnect-Dashboards angepasst werden. Hierfür lässt sich das resultierende MTConnect-XML-Format mit minimalem Aufwand durch eine Anpassung des MTConnect-Informationsmodells in der Datei *umati2mtc/Agent/Devices.xml* flexibel ändern. Anschließend muss die bestehende *mapping.xlsx*-Datei entsprechend aktualisiert oder eine neue Datei erstellt werden, um die Zuordnung der Variablen festzulegen. Die neue Mapping-Datei ist in der Konfigurationsdatei *umati2mtc/config.json* einzutragen. Danach wird eine Instanz des umatiGateway eingerichtet und mit dem umati-Server der Maschine verbunden. Die MQTT-Einstellungen des Gateways sind so zu konfigurieren, dass die Daten an den bereitgestellten MQTT-Broker gesendet werden (localhost:1883). Falls für bestimmte Variablen spezielle Umwandlungen erforderlich sind, sind diese im Programmcode im Modul *data_conversion* zu implementieren.

2 Zusammenfassung und Ausblick

Durch die Analyse der OPC UA- und MTConnect-Spezifikationen sowie entsprechender Softwareumsetzungen und Bibliotheken konnten Lösungen zur Integration von MTConnect in das umati-Ökosystem konzipiert und erfolgreich realisiert werden. Als zentrale Ergebnisse liegen Mappingtabellen zur bidirektionalen Konvertierung von Daten zwischen beiden Schnittstellen sowie zwei funktionsfähige Softwareadapter für die praktische Umsetzung vor. Der dokumentierte Quellcode wurde dem VDW bereitgestellt. Damit wurden die Kernziele des Forschungsvorhabens erreicht. Durch eine ausführliche technische Dokumentation sowie durch die zugängliche Gestaltung der Softwarelösungen samt Demo-Modus wurde die einfache Weiterverwendung und Weiterentwicklung der Ergebnisse sichergestellt.

Das im Rahmen des Projekts erworbene Wissen zu *OPC UA for Machine Tools* und MTConnect wird am IFW auch künftig genutzt. Darüber hinaus werden die erzielten Ergebnisse und Projekttätigkeiten aktiv in der Öffentlichkeitsarbeit des IFW kommuniziert. Aufbauend auf der gewonnenen Expertise wird das IFW auch zukünftig einen aktiven Beitrag zur weiteren Verbreitung und Etablierung von umati in der Industrie leisten.

3 Literaturverzeichnis

[CHR23] Christian von Arnim, et al. (2023): umati/Sample-Server: Release 1.1.1, online verfügbar unter: https://github.com/umati/Sample-Server/releases/tag/v1.1.1

[MAZ22] Mazak (2022): Demo Agents, online verfügbar unter: http://mtconnect.mazak-corp.com/

[MTC19] MTConnect Institute (2019): OPC 30070-1: OPC UA for MTConnect® Part1: Device Model (Ver. 2.00.00)

[MTC23a] MTConnect Institute (2023): MTConnect Standard. Part 3.0 - Observation Information Model - Version 2.2.0

[MTC23b] MTConnect Institute (2023): MTConnect Standard. Part 2.0 - Device Information Model - Version 2.2.0

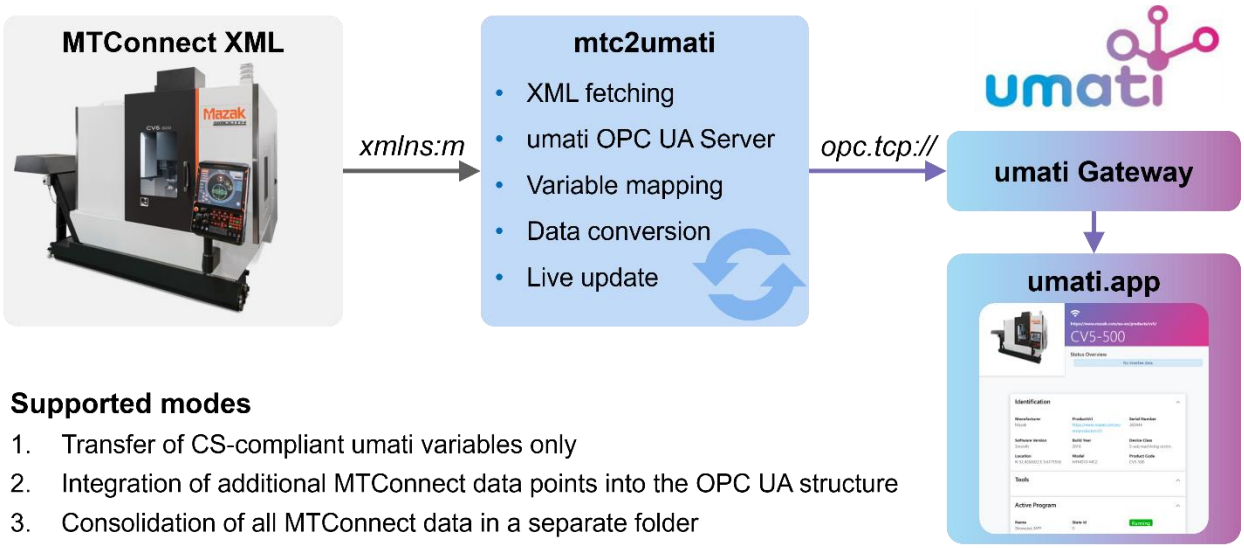
[MTC25] MTConnect Institute (2025): MTConnect Standard, online verfügbar unter: https://www.mtconnect.org/standard-download20181, zuletzt aufgerufen am 12.08.2025

[VDM24] VDMA (2024): VDMA 40501-1. OPC UA for Machine Tools - Part 1: Machine Monitoring and Job Management

[VDM25] VDMA (2025): VDMA 40001-1. OPC UA for Machinery - Part 1: Basic Building Blocks (Draft)

4 Anhang

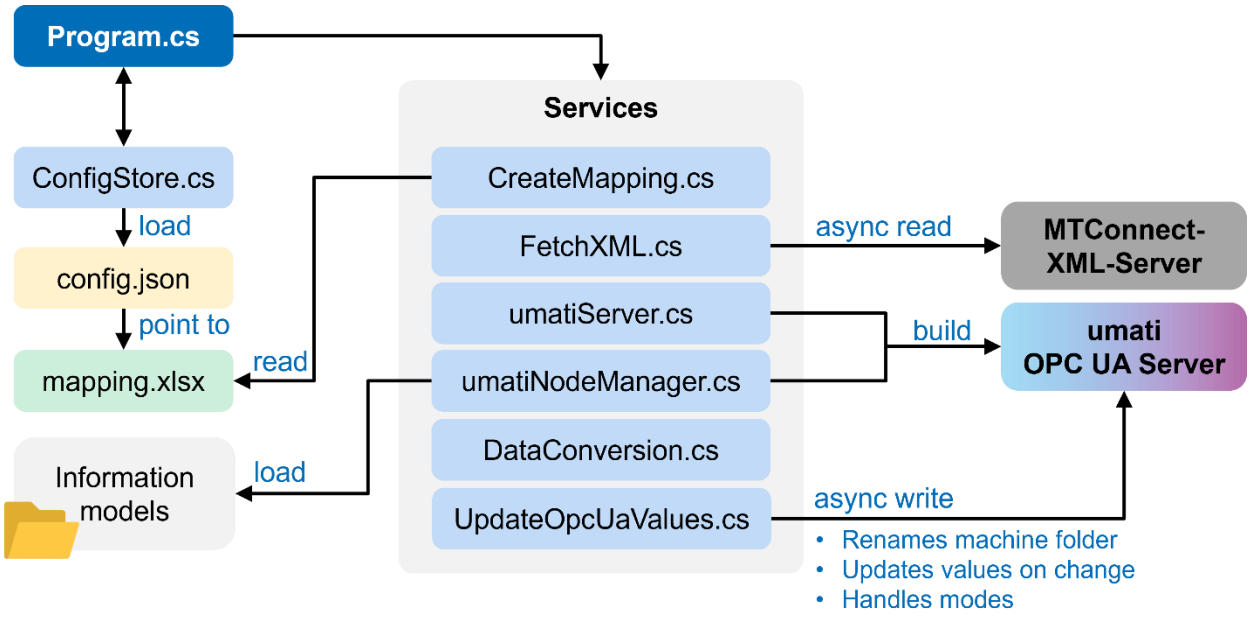
4.1 mtc2umati – Überblick



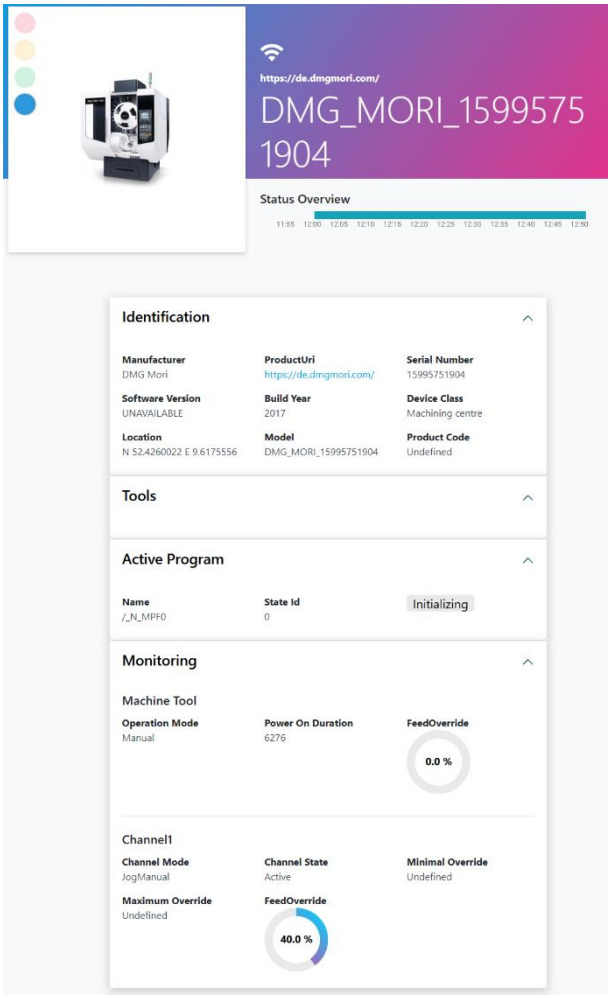
Supported modes

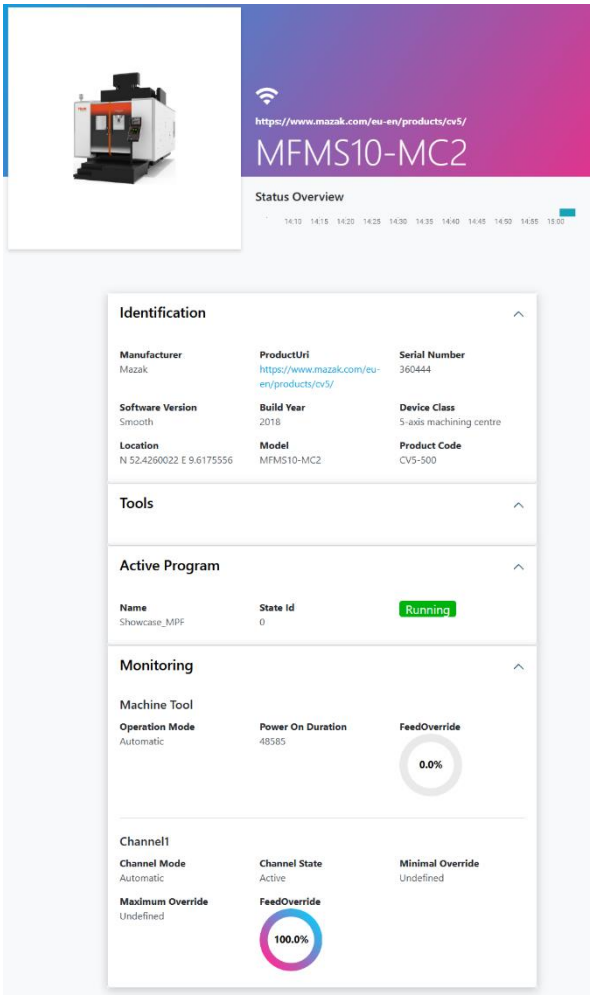
1. Transfer of CS-compliant umati variables only
2. Integration of additional MTConnect data points into the OPC UA structure
3. Consolidation of all MTConnect data in a separate folder

4.2 mtc2umati – Technische Umsetzung

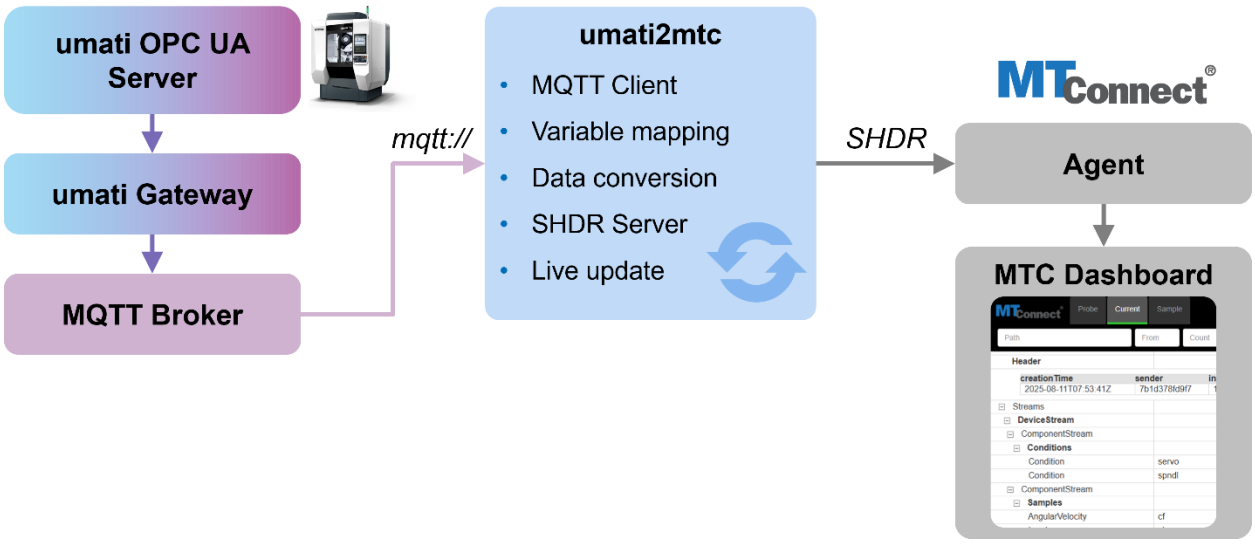


4.3 Integration von MTConnect-Referenzmaschinen auf der umati.app

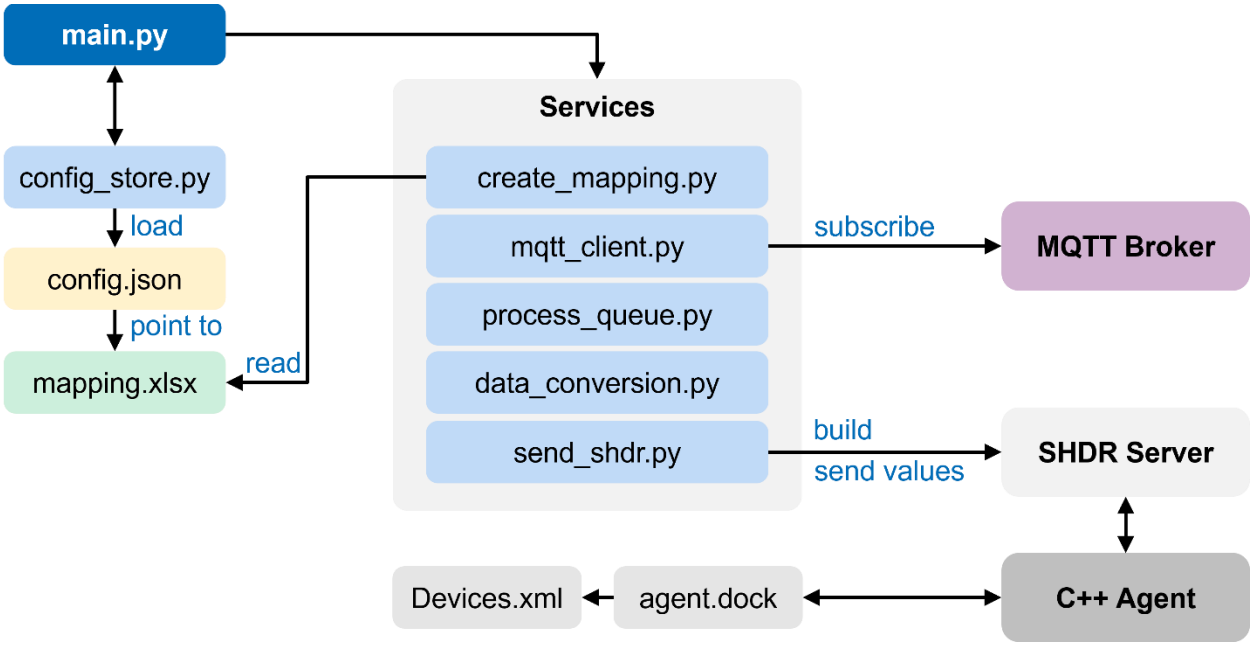




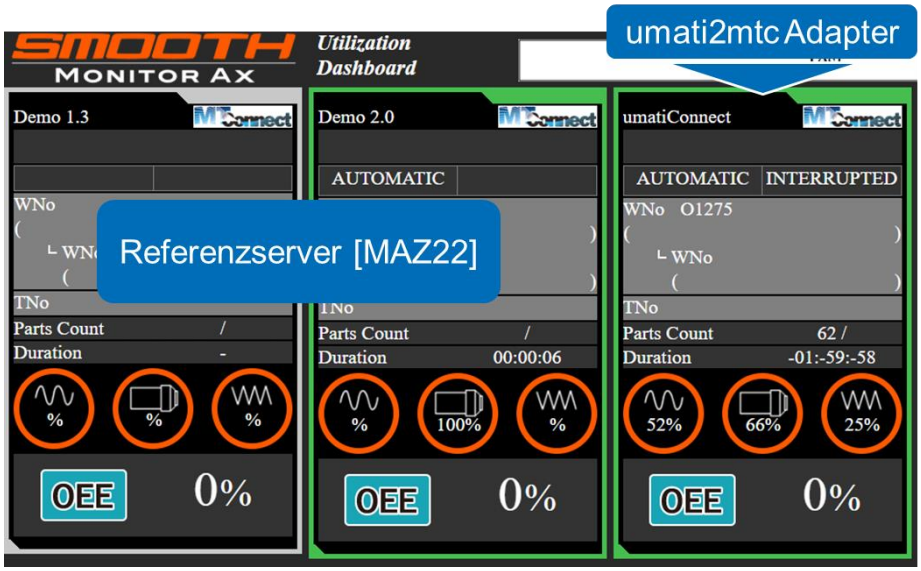
4.4 umati2mtc – Überblick



4.5 umati2mtc – Technische Umsetzung



4.6 Integration des umati2mtc Adapters in Smooth Monitor Ax



4.7 umati2mtc – Demo Modus

- **umati2mtc** provides a demo mode using **docker compose**
- **Simulator** publishes generated data for *PowerOnDuration* and *FeedOverride* in *umatiGateway* JSON-format
- MTConnect **Agent dashboard** is available under *localhost:5000*