BaSys überProd – BaSys für die
unternehmensübergreifende Produktion

Architektur für die Integration eines Manufacturing Service Bus mit der Verwaltungsschale (BaSyx MSB Integration)

Teilvorhaben des Fraunhofer IPA

Autoren: Daniel Schel, Matthias Stöhr, Lukas Rauh
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA
Nobelstraße 12, 70569 Stuttgart

Dokument-Version	3.0
Datum	21.08.2023
Verbreitungsgrad	Öffentlich
Projekt	BaSys überProd
Förderkennzeichen	01|S20094C
Laufzeit	1.1.2021 – 31.3.2023

Das diesem Dokument zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen01IS20094C gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der Autorin/beim Autor.“ (Nr. 5.2.2 NKBF 2017)

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Inhaltsverzeichnis

Versionen

Version	Datum	Inhalt
1	10.06.2022	Beschreibung der C4-Architektur für das MSB BaSyx Interface
2	22.03.2023	Erweiterte Konzeptbeschreibung
3	21.08.2023	Ergänzung des AAS Proxy Service

1 Einführung

Das „Verbundprojekt BaSys überProd“ baut auf den Ergebnissen und Technologien der Projekte „BaSys 4.0“ und „BaSys 4.2“ auf, um diese in konkreten Anwendungskontexten als wiederverwendbare Lösungen für die Industrie 4.0 und die unternehmensübergreifende Produktion zu realisieren. Unter Verwendung der von der Plattform Industrie 4.0 konzipierten „Verwaltungsschale“ als Basistechnologie und der Open-Source Middleware „Eclipse BaSyx“ als Basisinfrastruktur, liegt der Forschungsfokus im Verbundprojekt auf den zur Operationalisierung von Industrie 4.0 Anwendungen notwendigen Schnittstellen, Modellen und Daten-strukturen. Aufgabe des Teilvorhabens des Fraunhofer IPA ist dabei die Erweiterung der Verwaltungsschaleninfrastruktur durch Schnittstellen zu und Interoperabilität mit einem Manufacturing Service Bus, deren Bereitstellung in einem Edge-Rechenzentrum, die Integration von Anwendungen mit Verwaltungsschalenteilmodellen, sowie die dynamische Bereitstellung und der Betrieb von Industrie 4.0 Diensten auf Edge-Devices mithilfe der Verwaltungsschale und Metadaten.

Zur Realisierung der Integration eines Manufacturing Service Bus (MSB) mit BaSyx wurde eine C4-Architektur erstellt, die als Referenzarchitektur für die Integration weiterer Bus Systeme verwendet werden kann. Auf Basis der Architektur wurde die Integration mit BaSyx für die Middleware „Virtual Fort Knox Research – Manufacturing Service Bus“ (MSB) [1] umgesetzt. Nachfolgende Beschreibungen beziehen sich auf diesen MSB, können aber auch auf einen anderen Service Bus übertragen werden.

Zum besseren Verständnis der BaSyx Komponenten wird für dieses Dokument Vorwissen auf Basis der BaSyx Dokumentation voraussgesetzt [2].

2 Problemstellung und Zielsetzung

Sowohl das BaSyx Ökosystem als auch das MSB-Ökosystem bieten eine etablierte Lösung für die Integration von Diensten (digitalen Assets, Edge Devices, Smarten Objekten, Applikationen) und deren Interoperabilität. Die beiden Lösungen unterscheiden sich dabei in der Dienst-Repräsentation und den verwendeten Kommunikationsmustern und Datenmodellen. Damit sind die Dienste beider Ökosysteme nicht kompatibel und können nicht ohne hohen Aufwand in Rahmen von implementierten Einzellösungen miteinander verbunden werden.

Durch die Integration des MSB mit BaSyx kann eine Erweiterung des BaSyx Ökosystems um eine Event-basierte, asynchrone Kommunikationsfähigkeit und eine konfigurierbare Service Orchestrierung erreicht werden. Hierzu soll mit dem MSB BaSyx Interface eine neue Schnittstelle geschaffen werden, um das BaSyx Directory mit dem MSB Connected Service Management zu synchronisieren und Aufrufe aus dem jeweiligen Ökosystem in das Format und das Kommunikationsmuster des anderen Ökosystems zu transformieren.

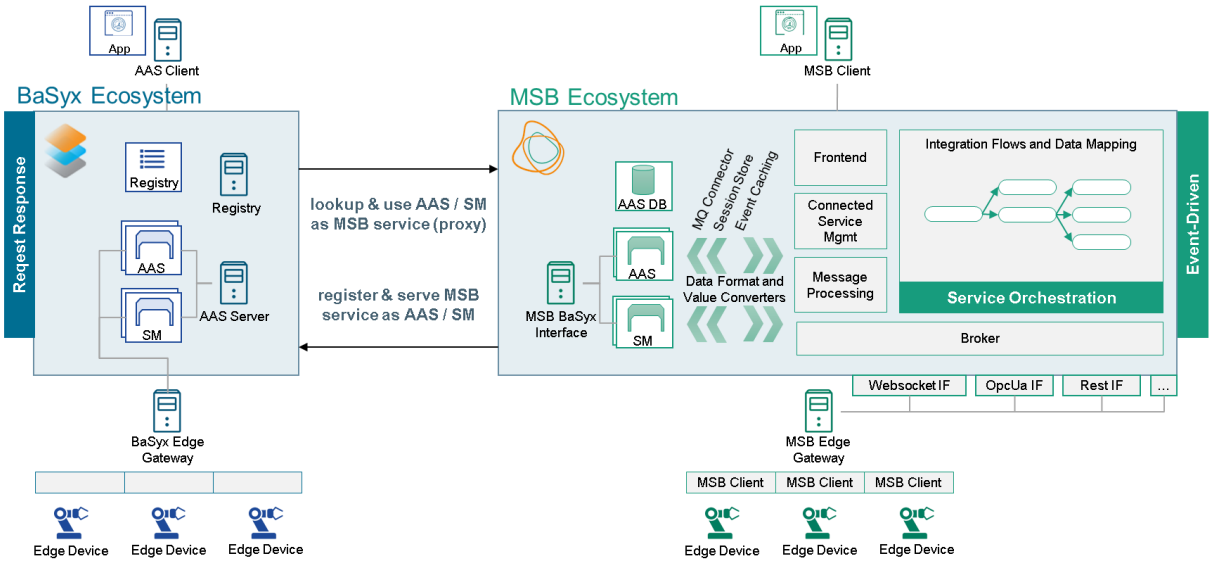

Abbildung 1: Konzeptbild zur "BaSyx MSB Integration"

Das im BaSyx Ökosystem verwendete Kommunikationsmuster basiert auf „Request-Response“, während das Kommunikationsmuster des MSB „Event-basiert“ ist. Das MSB BaSyx Interface soll hier zwischen beiden Kommunikationsmustern vermitteln. Die Beschreibung eines Assets und seiner Daten und Dienste wird in BaSyx als Verwaltungsschale und entsprechende Teilmodelle realisiert, entsprechend der Spezifikation der Plattform Industrie 4.0 [9]. Assets (Smarte Objekte) im Kontext des MSB erhalten eine MSB-spezifische Selbstbeschreibung. Dabei unterscheidet sich auch das Schema für die Daten und Datenformate. Entsprechend werden für die Übermittlung von Daten bidirektionale Konverter und Generatoren benötigt. Die angebundenen Dienste sollen unabhängig von der ursprünglichen Anbindung durch Synchronisation zwischen der BaSyx Registry und dem MSB Connected Service Management beiden Ökosystemen zur Verfügung stehen.

3 Architektur

Zur Beschreibung der Architektur der BaSyx MSB Integration wurde das C4 Modell [3] ausgewählt. Das C4 Modell beschreibt das zu entwickelnde System als Teil einer übergreifenden Systemlandschaft. Die einzelnen Systeme werden dabei über weitere Ebenen detailliert.

3.1 Anforderungen

Basis für die Architektur ist ein Anforderungskatalog bestehend aus Anforderungen aus der Szenarien-Erprobung und Anforderungen aus Workshops in den Arbeitspaketen und Anwendungsfällen des Projekts.

3.1.1 Hauptanwendungsfälle

Es kann zwischen zwei Hauptanwendungsfällen unterschieden werden:

(1) Epic 1: Integration von MSB-Diensten in das BaSyx Ökosystem:
Dienste (Smarte Objekte resp. Assets) des MSB erhalten über das MSB BaSyx Interface eine aktive Verwaltungsschale mit AAS API und Submodel API, inklusive Operations, und können so im BaSyx Ökosystem mittels AAS-Client angesprochen und in BaSyx Anwendungen integriert werden.

(2) Epic 2: Integration von BaSyx Diensten in das MSB-Ökosystem:
Aktive Verwaltungsschalen aus BaSyx erhalten eine Repräsentation im MSB als MSB-Dienst und können über die Service Orchestrierung des MSB mit anderen Verwaltungsschalen oder MSB-Diensten dynamisch verbunden werden.

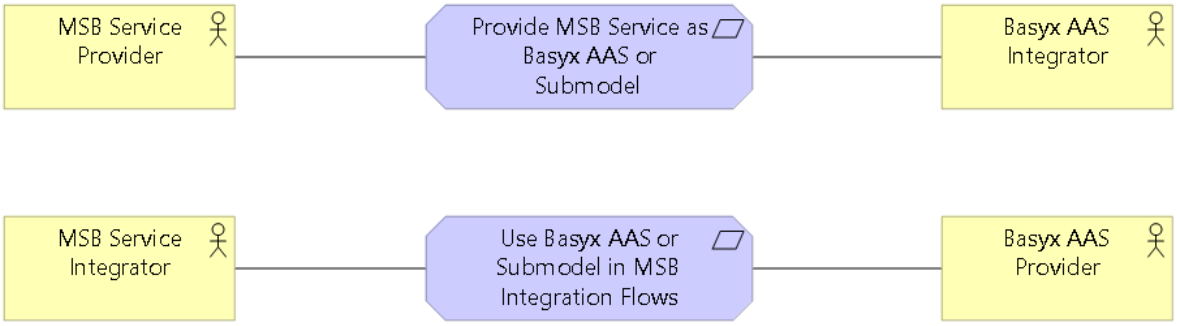

Abbildung 2: Hauptanwendungsfälle

3.1.2 Funktionale Anforderungen

Um die bidirektionale Integration (Epic 1 und 2) realisieren zu können, müssen funktionale übergreifende Anforderungen erfüllt werden. Es werden bidirektionale Konverter benötigt, um

die Daten-Schemas und –Formate für das jeweilige Zielsystem bereitstellen zu können. Zusätzlich werden Caching-Funktionen benötigt, um die Request-Response-Pattern aus BaSyx auf die Event-basierte Kommunikation in Service Bus anwenden zu können. Dabei sollen sowohl synchrone als auch asynchrone Operationsaufrufe möglich sein. Zudem sollen zur Laufzeit geänderte Endpunkt-Informationen erkannt werden.

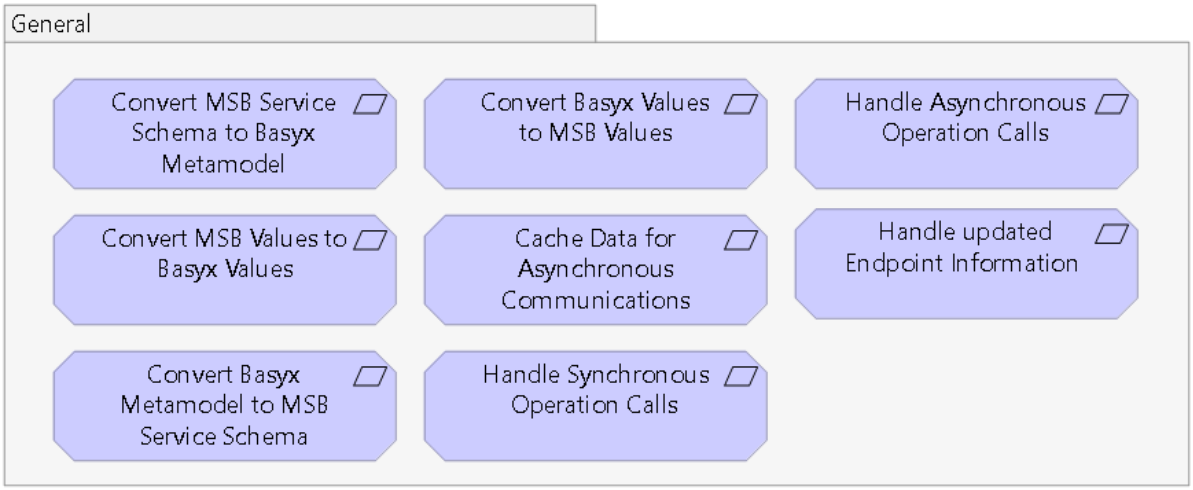

Abbildung 3: Funktionale Anforderungen (allgemein)

Neben den allgemeinen Anforderungen gibt es auch Epic-spezifische Anforderungen. Die spezifischen Anforderungen für Epic 1 sind in Abbildung 4: Funktionale Anforderungen (Epic 1)Abbildung 4 dargestellt. Der Service Bus soll über das BaSyx Interface eine Verwaltungsschalen-API (inkl. API für die Teilmodelle) anbieten. Eine Speicherung bereits generierter Verwaltungsschalen soll dabei effiziente Abfragen ermöglichen. Die entsprechenden Verwaltungsschalen sollen dabei über die BaSyx Registry zugänglich gemacht werden. Das BaSyx Interface soll synchrone und asynchrone Kommunikation unterstützen und entsprechend Aufrufe aus BaSyx in Events und zugehörige Sessions transformieren. Antworten (Reponse Events) sollen dabei in der Verwaltungsschale abgelegt und über die Session zurückgespielt werden.

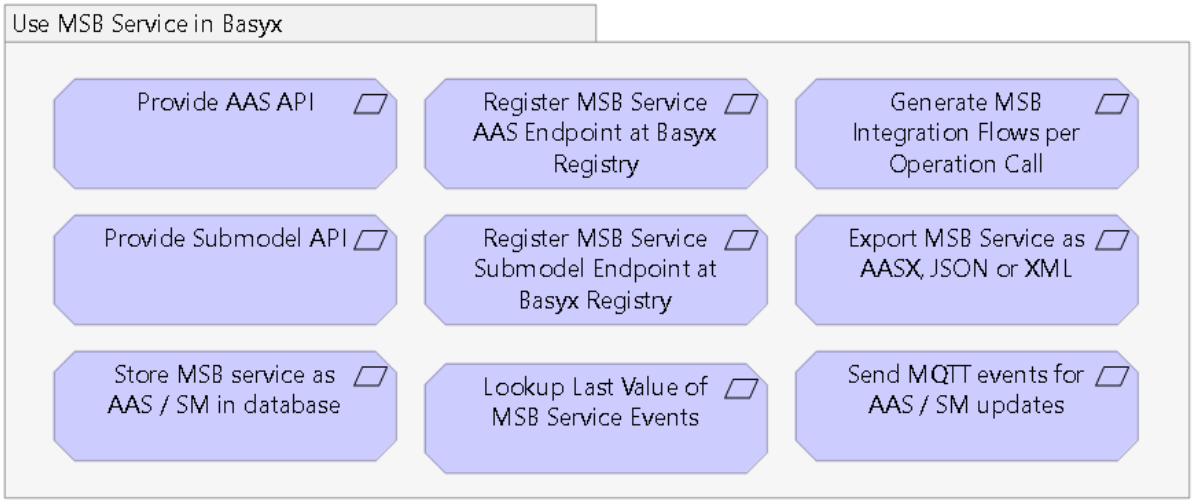

Abbildung 4: Funktionale Anforderungen (Epic 1)

Die spezifischen Anforderungen für Epic 2 sind in Abbildung 5Abbildung 4: Funktionale Anforderungen (Epic 1) dargestellt. Das BaSyx Interface soll Verwaltungsschalen aus dem BaSyx Ökosystem konsumieren und in eigenen Workflows (Message Processing) anwenden können. Die Verwaltungsschalen sollen dabei im Directory des Service Bus (Connected Service Management) eine Repräsentation erhalten.

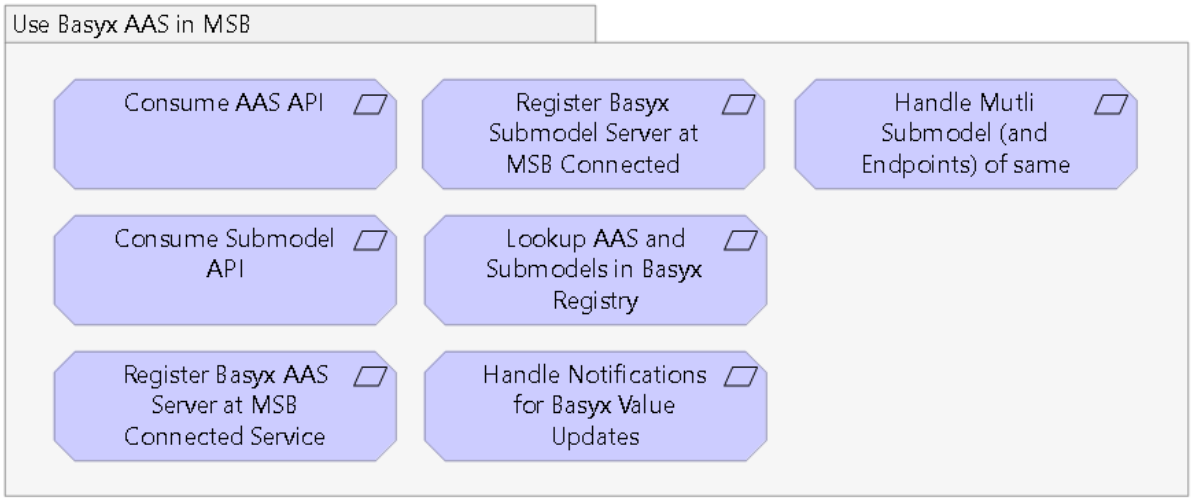

Abbildung 5: Funktionale Anforderungen (Epic 2)

3.2 Architektur-Ebenen

Abbildung 6 zeigt die ersten drei Ebenen der erstellten C4 Architektur für das MSB BaSyx Interface:

(1) In Ebene 1 „Systemlandschaft“ ist die Einordnung des BaSyx Systems und des MSB-Systems in den Gesamtsystemkontext dargestellt.
(2) In Ebene 2 „Container“ werden beide Systeme mit Ihren Hauptkomponenten (separate Container) und deren Zusammenhänge dargestellt.
(3) In Ebene 3 „BaSyx Interface“ wird das neu zu entwickelnde MSB BaSyx Interface mit seinen Komponenten detailliert dargestellt und Zusammenhänge zwischen den Komponenten und den BaSyx Hauptkomponenten beschrieben.

Die C4 Architektur ist die Grundlage für die Entwicklung des MSB BaSyx Interface bzw. der BaSyx Schnittstelle für weitere Bus Systeme.

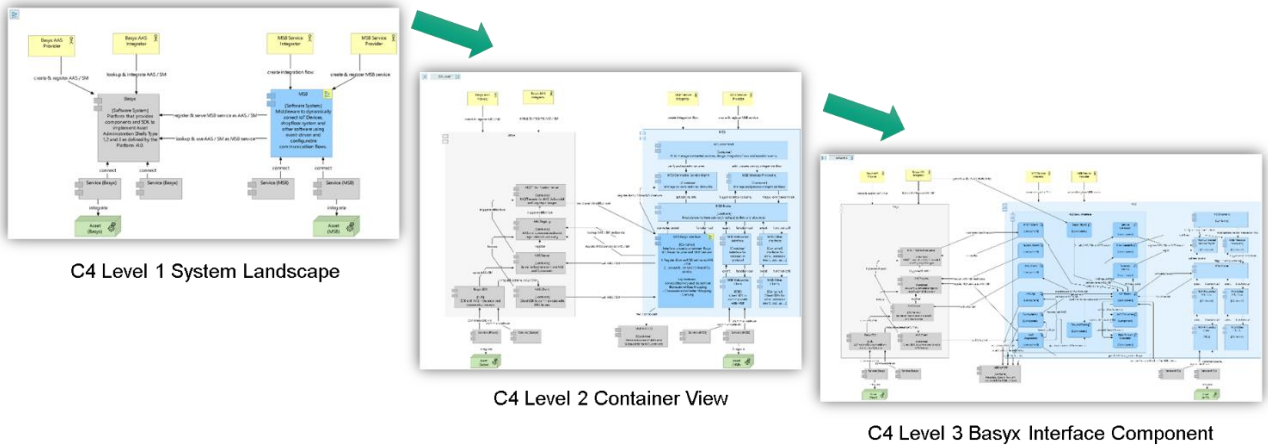

Abbildung 6: Übersicht der ersten drei Ebenen der C4 Architektur für das MSB BaSyx Interface

3.2.1 C4-Ebene 1 Systemlandschaft

In Ebene 1 „Systemlandschaft“ ist die Einordnung des BaSyx Systems und des MSB-Systems in den Gesamtsystemkontext dargestellt. Beide Systeme nutzen Dienste (Services) zur Integration der Assets. Bei BaSyx geschieht dies in Form von SDKs oder Off-the-Shelf Komponenten wie der Eclipse BaSyx Data Bridge. Die Dienste erhalten dabei einen digitalen Zwilling in Form der Verwaltungsschale.

Am MSB werden Dienste über Client- oder Server-SDKs angebunden, wobei diverse Protokolle wie OpcUa, Websocket, HTTP/Rest oder MQTT unterstützt werden. Dabei wird die Integration meist von der Dienst-Seite durch die Registrierung am Bus und Bereitstellung einer Selbstbeschreibung übernommen. Die Selbstbeschreibung bringt die heterogenen Datenstrukturen und Kommunikationsmuster der Dienste in ein einheitliches durch den Bus anwendbares Format.

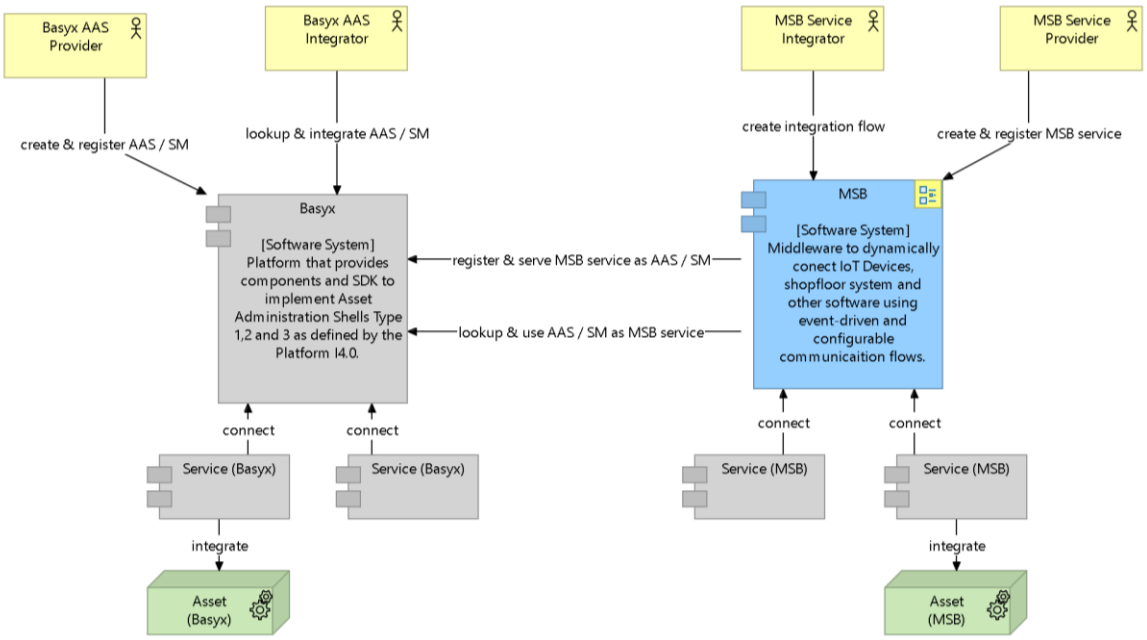

Abbildung 7: C4-Ebene 1 Systemlandschaft

Die an die Systeme angebundenen Dienste können durch Einführung des MSB BaSyx Interface nicht nur im eigenen Ökosystem zur Verfügung stehen, sondern auch übergreifend mit den Diensten des jeweils anderen Systems kommunizieren und Daten ausrauschen.

3.2.2 C4-Ebene 2 Container View

Abbildung 8 detailliert die relevantesten Container des BaSyx Ökosystems. Der **AAS-Server** übernimmt die Bereitstellung der Verwaltungsschale und verwaltet die Speicherung. Die Verwaltungsschalen werden dabei im Metamodell der Verwaltungsschale gespeichert und über die API bereitgestellt (siehe „Details of the Asset Administration Shell“ Part 1 [5] und Part 2 [8]). Der AAS-Server registriert die Endpunkte seiner Verwaltungsschalen in der **BaSyx Registry**. Damit werden die Zugriffspunkte auf die AAS für weitere Systeme bekanntgemacht.

Über das **BaSyx SDK** können so Anwendungen einfach einen **AAS-Client** implementieren, der die Veraltungsschale über den AAS-Server bereitstellt oder auch konsumiert. Erweitert wird das Gesamtsystem durch einen **MQTT-Server**, der Events von der BaSyx Registry und dem AAS-Server verbreitet. Ausgelöst werden diese Events durch Änderungen an der Verwaltungsschale oder an den registrierten Endpunkten in der Registry.

Abbildung 9 detailliert die wichtigsten Container des MSB-Ökosystems. Ein Dienst verbindet und registriert sich über das für das Protokoll passende **Interface** und das für den Dienst passende **Client-SDK**. Die dabei übermittelte Selbstbeschreibung wird in einem Directory, dem **Connected Service Management**, abgelegt. Die Selbstbeschreibung enthält neben allgemeinen Daten wie UUID und Bezeichnung auch eine Liste mit Beschreibung der Events und Funktionen inklusive Daten-Schema in dem für den Bus vorgegebenen Format (bspw. JSON-Schema [4]). Auf Basis der Events und Funktionen können so Workflows über das **Message Processing** modelliert werden. Diese Workflows verknüpfen Events und Funktionen der angebundenen Dienste und können dabei notwendige Transformationen der Daten beinhalten. Damit können über den MSB auch Dienste, die kein gemeinsames Datenmodell teilen miteinander integriert werden. Zur Laufzeit werden eingehende Events über die **Message Queue** (bspw. Rabbit-MQ) weitergeleitet. Das Message Processing öffnet dabei entsprechend der modellierten Workflows Queues für die Ziel-Systeme sowie das erforderliche Daten-Mapping, und verknüpft diese miteinander. Die ausgehenden Events werden abschließen über das dem Ziel-System zugeordneten Interface als Funktionsparameter zugestellt. Um den Prozess der Verwaltung der Dienste und der Modellierung der Workflows für Benutzer zu vereinfachen, kann eine grafische Benutzeroberfläche als **Frontend** verwendet werden.

Für die Integration des MSB mit BaSyx wird ein neues Interface, das **MSB BaSyx Interface**, benötigt. Dieses setzt das beschriebene Epic 1 (siehe Kapitel 3.1.1) und die zugehörigen Anforderungen um. Es stellt die notwendigen bidirektionalen Konverter bereit, sorgt für die Bereitstellung der Verwaltungsschalen der MSB-Dienste über das AAS-Metamodell und die AAS und Submodel API, sorgt für deren Registrierung, Speicherung und Caching, die Integration mit der Message Queue und das Mapping der unterschiedlichen Kommunikationsmuster beider Ökosysteme. Für die Anbindung von externen Verwaltungsschalen und deren Repräsentation im Connected Service Management wird zudem ein **AAS Proxy Service** benötigt (siehe Epic 2). Dieser überführt die AAS in die Selbstbeschreibung des Service Bus und sorgt für das Mapping der Daten. So können externe Verwaltungsschalen in Workflows des Message Processing integriert und mit weiteren Diensten verknüpft werden.

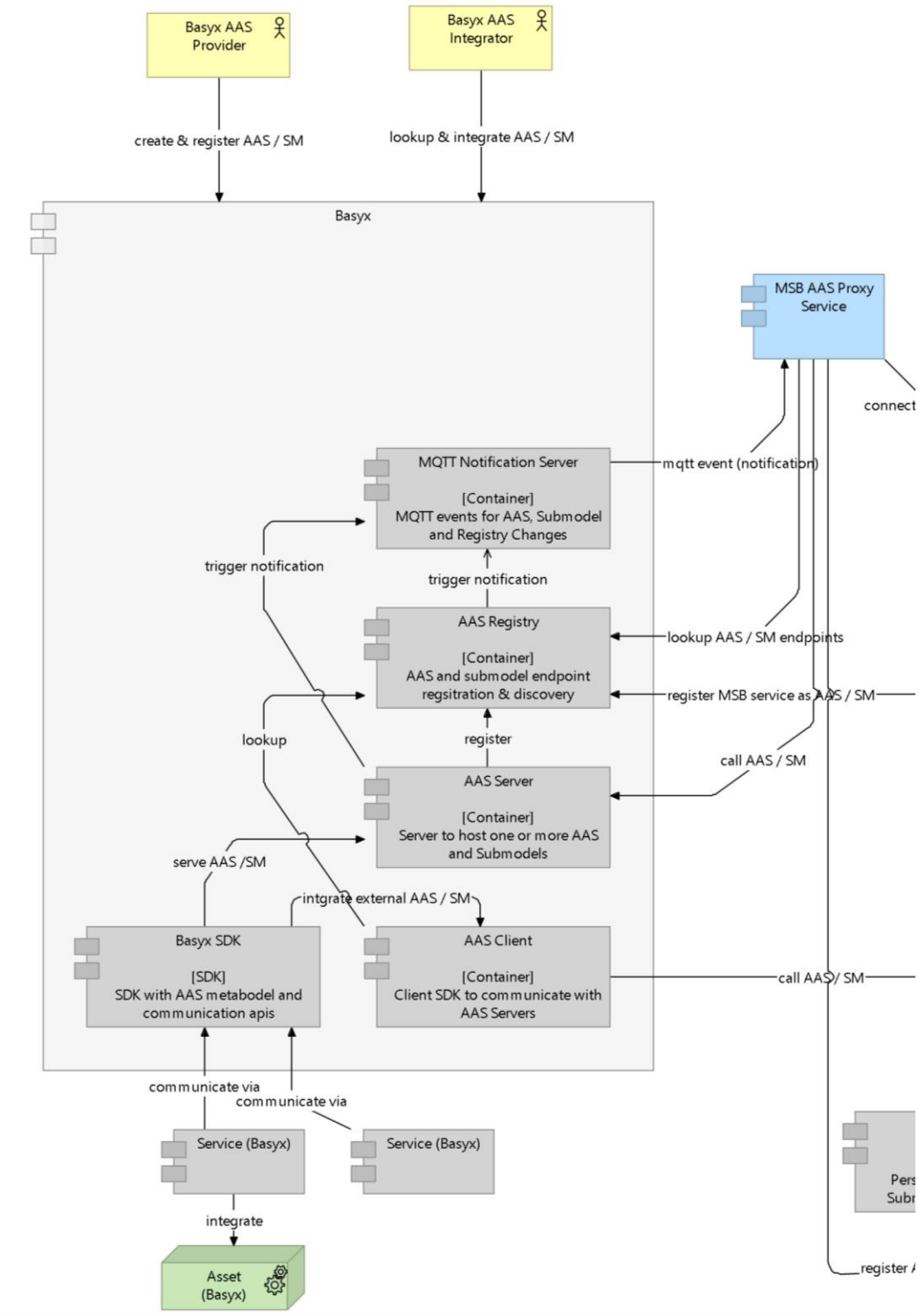

Abbildung 8: C4-Ebene 2 Container View (links)

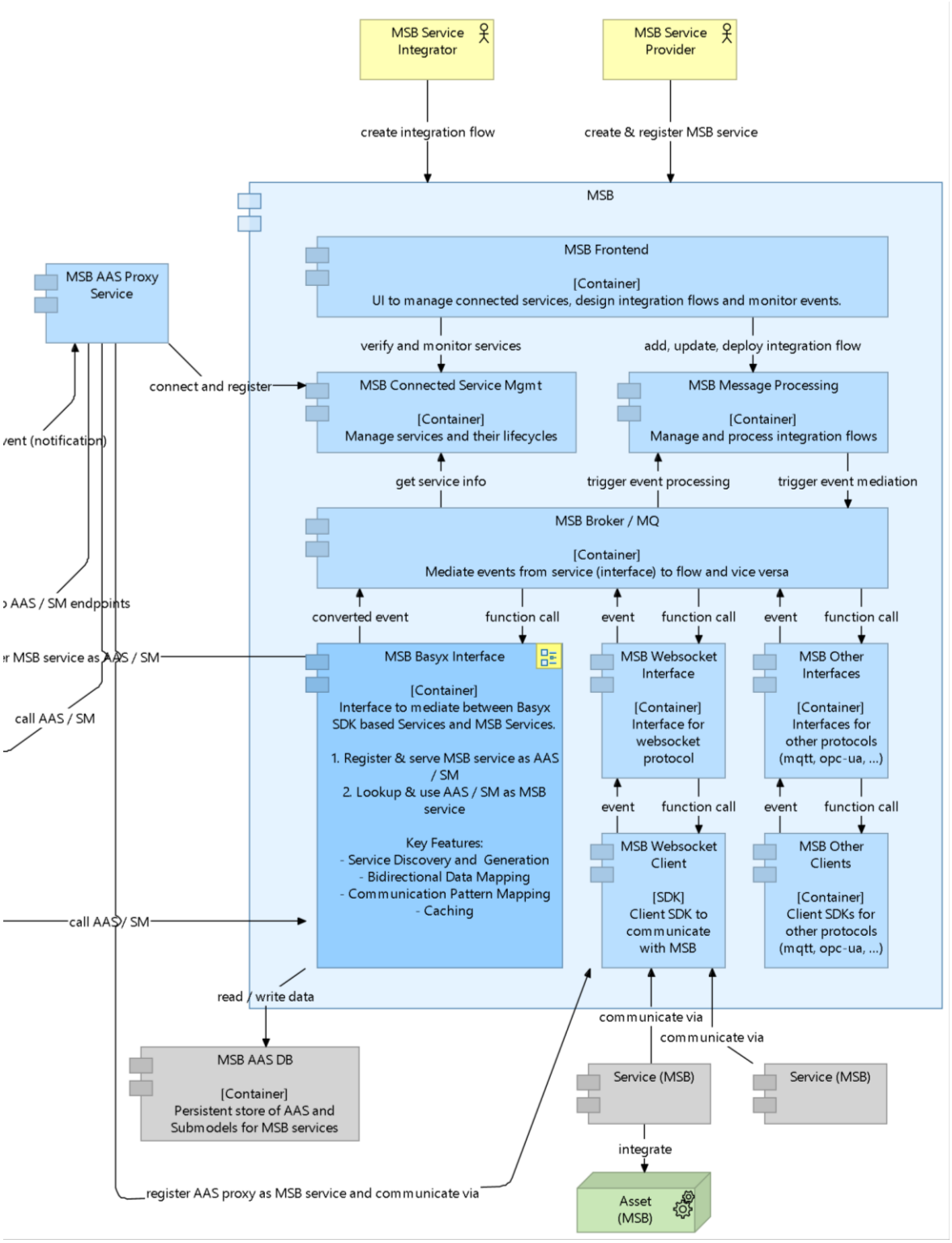

Abbildung 9: C4-Ebene 2 Container View (rechts)

3.2.3 C4-Ebene 3 Component View „MSB BaSyx Interface“

Ebene 3 detailliert das MSB BaSyx Interface mit seinen wichtigsten Komponenten. Diese Komponenten werden nachfolgend beschrieben und sind in Abbildung 11 und Abbildung 12 dargestellt.

AAS Aggregator: Der AAS Aggregator ist für das Management der Verwaltungsschalen und deren Bereitstellung zuständig. Dabei instanziiert er eine AAS API und mehrere Submodel APIs pro Verwaltungsschale und kann diese über ein übergeordnetes Servlet bereitstellen. Beim Starten des BaSyx Interface ist der AAS Aggregator zudem zuständig für das Laden der Verwaltungsschalen aus der AAS DB und deren initiale Instanziierung.

AAS API: Die AAS API implementiert die Funktionen für die Verwaltungsschalen gemäß der Spezifikation „Details oft he Asset Administration Shell“ Part 2. Eingehende Anfragen werden vom Servlet über den AAS Aggregator an die für die AAS instanziierte API weitergeleitet und entsprechend verarbeitet.

Submodel API: Die Submodel API implementiert die Funktionen für die Teilmodelle gemäß der Spezifikation „Details oft he Asset Administration Shell“ Part 2. Eingehende Anfragen werden vom Servlet über den AAS Aggregator an die für die AAS instanziierten Submodel Apis weitergeleitet und entsprechend verarbeitet.

AAS DB: Die AAS DB sorgt für die Speicherung der Verwaltungsschalen der MSB-Dienste und ihrer Teilmodelle. Die Anzahl von Konvertierungen der Daten kann so deutlich reduziert werden. Zudem werden hier für die aktiven Teilmodelle der MSB-Dienste aktuelle Events Zwischengespeichert.

AAS-Converter und Data Format Converter: Bidirektionale Konverter sorgen für die Konvertierung der Daten-Schemas und Daten-Objekte sowie für die Übersetzung der Selbstbeschreibung in Verwaltungsschalen gemäß dem Metamodell der Verwaltungsschale. Dabei werden die Daten-Schemas des Service Bus (JSON-Schema) in Teilmodelle des Metamodells der Verwaltungsschale konvertiert und die gesendeten Daten-Objekte zur Laufzeit für das jeweilige Ziel-System transformiert. Die Konverter unterstützen dabei auch komplexe rekursive Datenstrukturen.

AAS-Broker: Der AAS-Broker ist das Bindeglied zwischen den Komponenten des Service Bus und den Komponenten des BaSyx Interface. Hier sind die Funktionen für die Vermittlung und Konvertierung der Dienste aggregiert sowie übergreifende Funktionen des BaSyx Interface zusammengefasst.

Request Store: Der Request Store wird benötigt, um das Request-Response Pattern des BaSyx Ökosystems auf die Event-basierte Kommunikation des Service Bus abbilden zu können. Der Request Store verwaltet dabei die durch einen Aufruf eines AAS-Client initiierte Session und sorgt für die Zuordnung der Antwort auf Basis eingehender Events. Die Request ID eines BaSyx „Invocation Request“ wird dabei als „Korrelation ID“ in den Events des Service Bus verwendet.

MqConnector: Der MqConnector integriert das BaSyx Interface mit der Message Queue des Service Bus. Dazu öffnet er entsprechende Queues für die Dienste mit generierter Verwaltungsschale und lauscht auf eingehende Events. Diese werden dabei auf Relevanz mit den Sessions im Request Store auf Basis der „Korrelation ID“ abgeglichen und in der AAS DB zwischengespeichert.

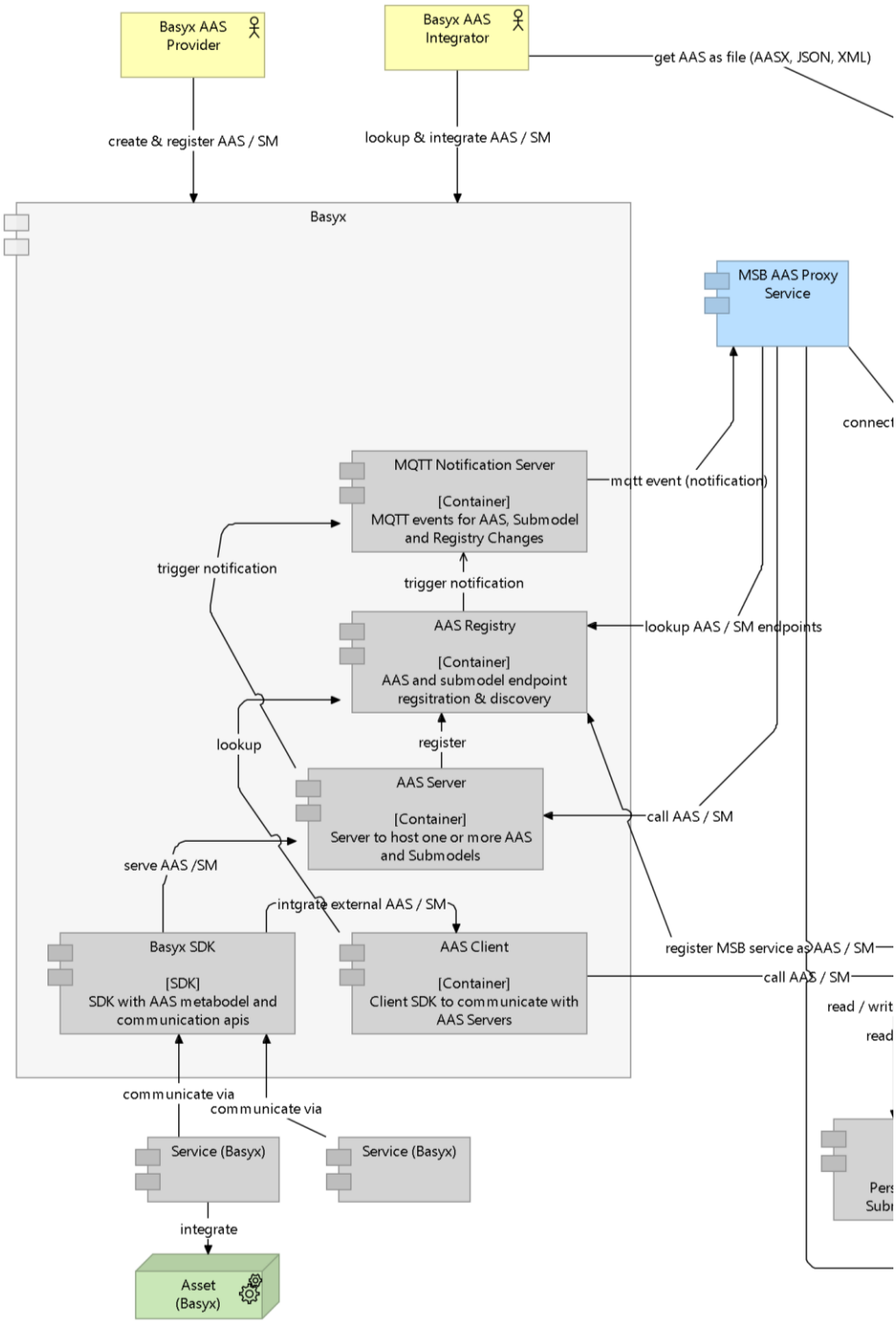

Abbildung 10: C4-Ebene 3 Component View (links)

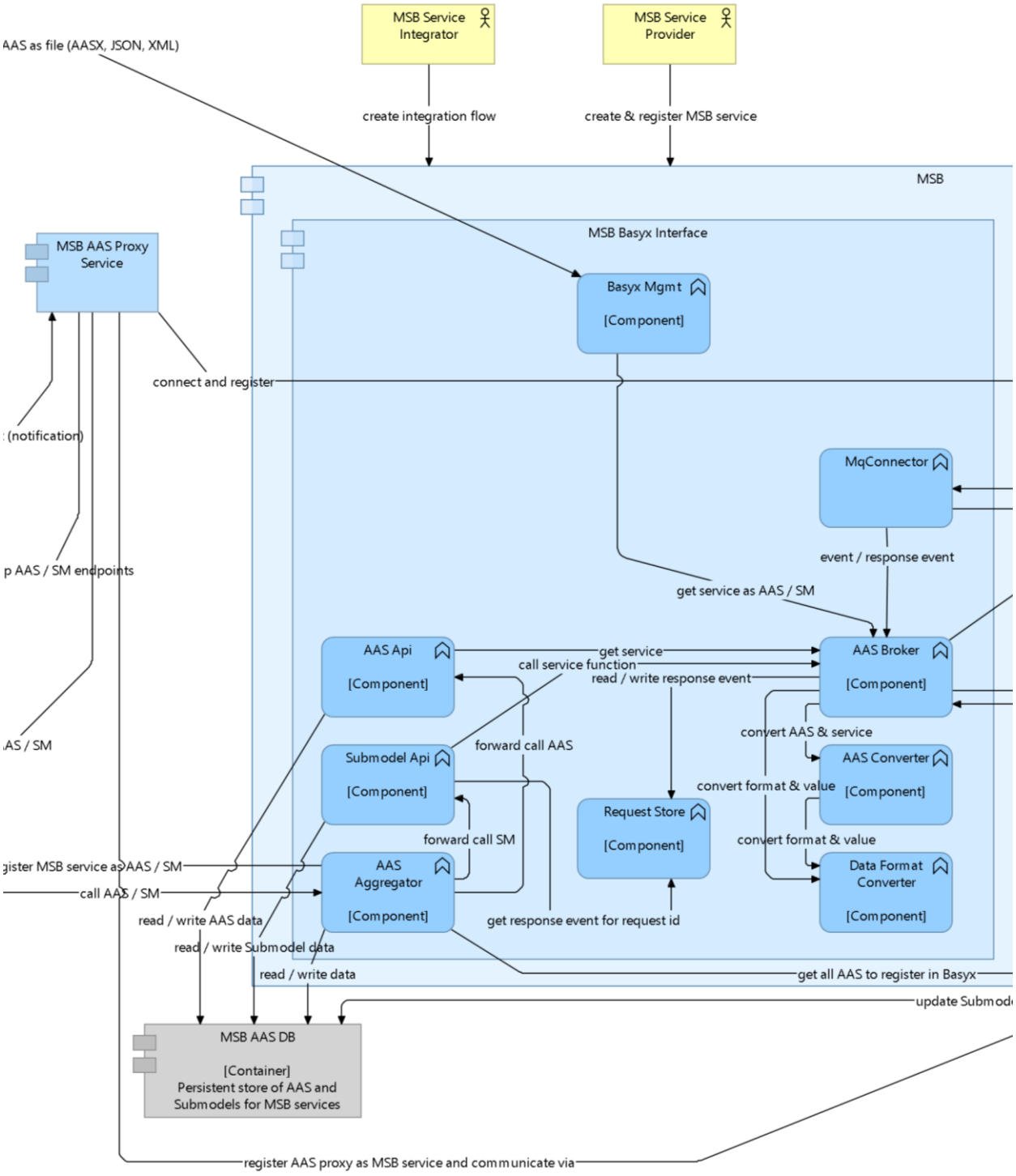

Abbildung 11: C4-Ebene 3 Component View (mitte)

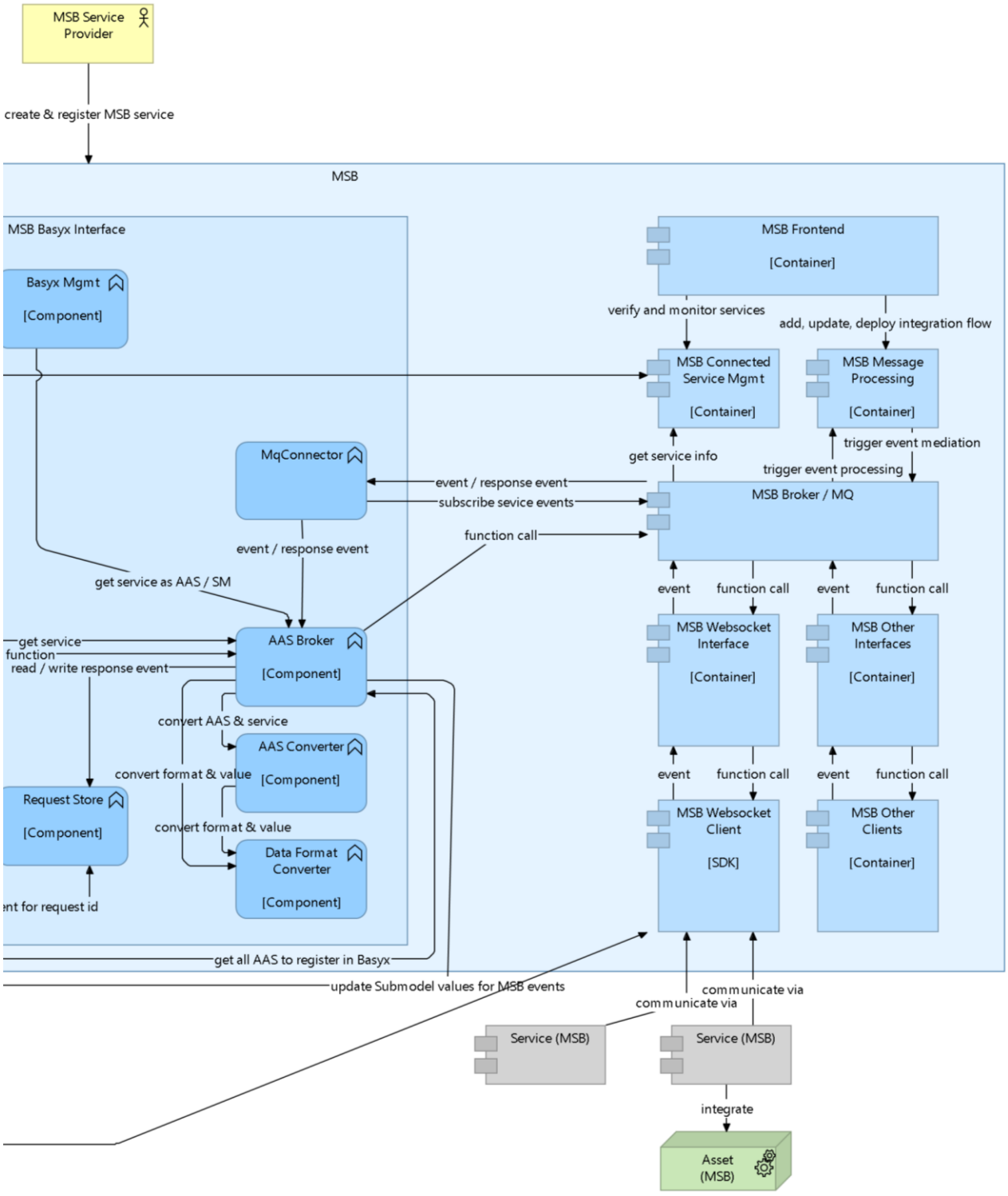

Abbildung 12: C4-Ebene 3 Component View (rechts)

3.3 Datenkonvertierung

Die Beschreibung eines mit dem Service Bus integrierten Dienstes (Anwendung, Smartes Objekt, Service, …) basiert auf einer Bus-spezifischen Selbstbeschreibung. Die Datenformate seiner Events und Funktionen werden dabei mittels Json Schema beschrieben. BaSyx und die Spezifikation der Verwaltungsschale in “Details of the Administration Shell - Part 1” werden dabei noch nicht unterstützt. Damit unterscheiden sich die Datenformate, Schnittstellen und Kommunikationsmuster beider Ökosysteme und eine Vernetzung ist ohne entsprechende Konverter und Transformationen nicht möglich.

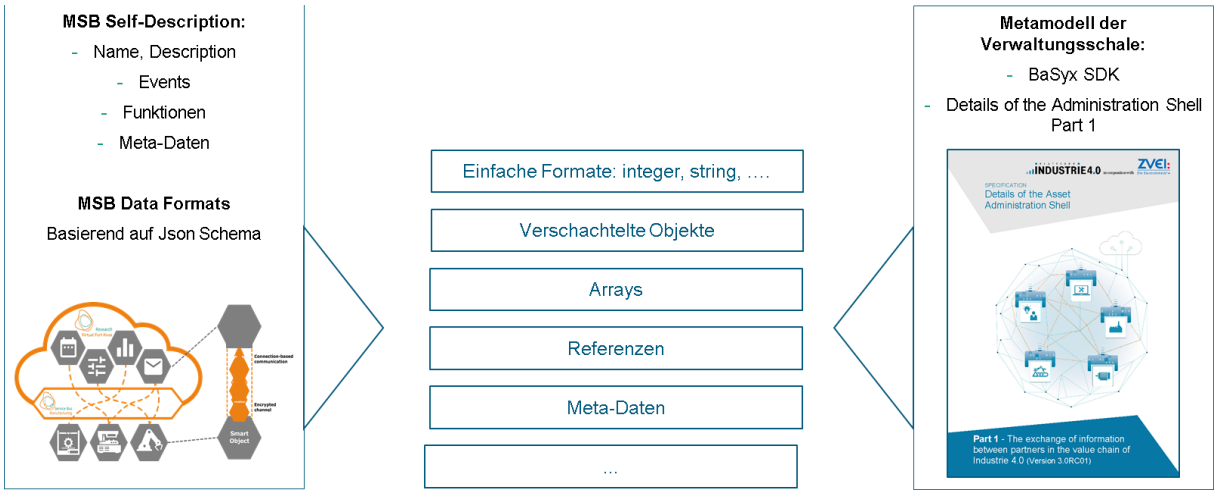

Abbildung 13: Bidirektionales Mapping zwischen BaSyx und MSB-Schemata

Für das MSB BaSyx Interface werden deshalb bidirektionale Konverter entwickelt, die zur Laufzeit die Formate in das jeweilige Zielformat konvertieren. Die Konvertierung umfasst neben den Datenformaten (Schema, Templates) auch die konkreten instanziierten Objekte mit ihren Werten (Datenobjekte). Dabei sollen alle relevanten Datentypen und Strukturen unterstützt werden:

- Einfache Datentypen wie String, Integer, Float, Double, Daetime, …
- Komplexe verschachtelte Objekte
- Arrays von Objekten und einfachen Datentypen
- Referenzen auf Objekte
- Meta-Daten zu Objekten

Die Konverter des MSB BaSyx Interface unterstützen dabei beliebige Kombinationen und Verschachtelungen. Abbildung 14 zeigt ein vereinfachtes Beispiel für die Schema-Konvertierung einer Firmware-Klasse zwischen Service Bus und BaSyx.

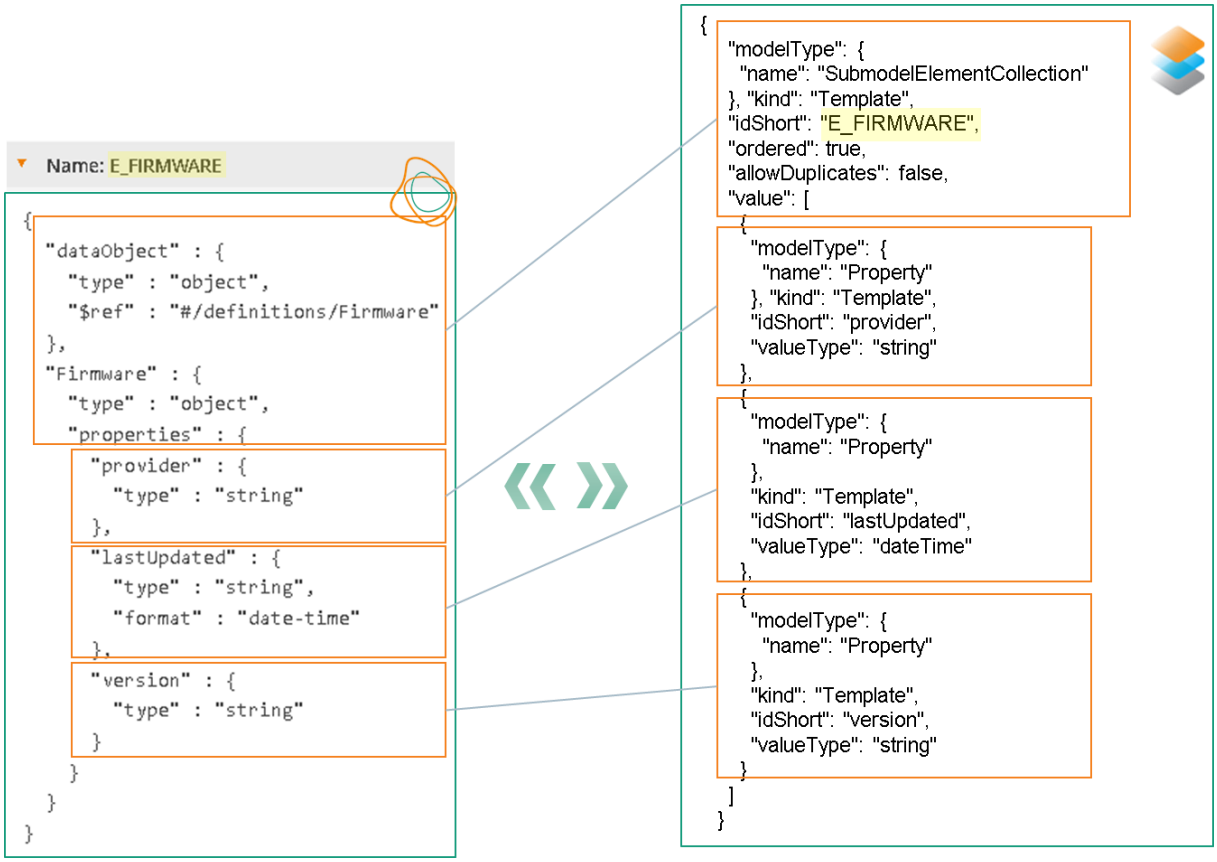

Abbildung 14: Einfaches Beispiel für das Schema-Mapping

Abbildung 15 ergänzt das Beispiel um die Konvertierung eines einfachen instanziierten Objekts des beschriebenen Schemas.

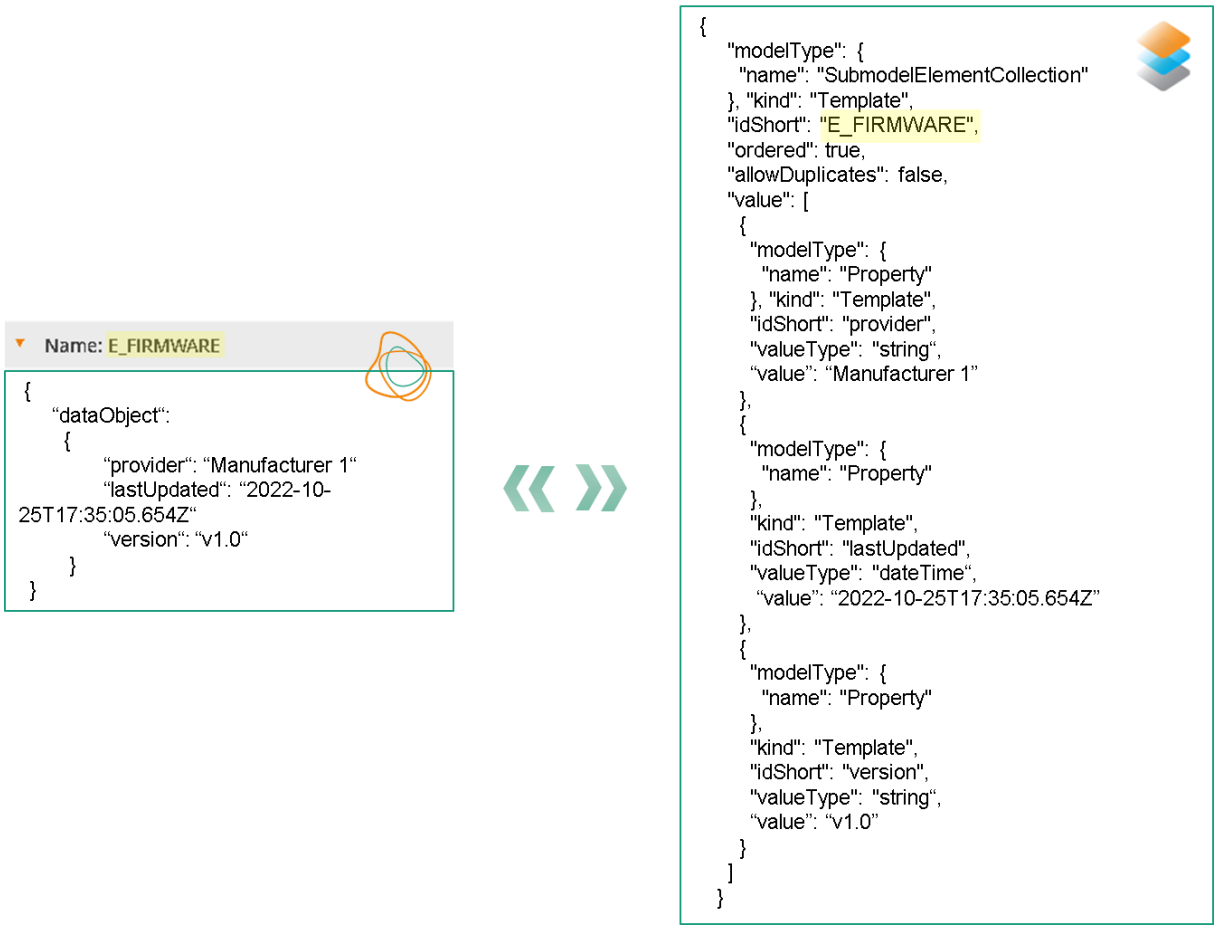

Abbildung 15: Einfaches Beispiel für das Datenobjekt-Mapping

3.4 Datenspeicherung und Caching

Die Konvertierung einer kompletten Selbstbeschreibung eines Dienstes kann je nach Komplexität der Selbstbeschreibung sehr rechenintensiv sein. Die Funktionen und Events des Dienstes können beliebig komplexe verschachtelte Datenformate haben (gemäß Json Schema). Damit nicht bei jedem Request aus BaSyx über die AAS und Submodel APIs diese Konvertierungsschritte wiederholt werden müssen, ist die Integration einer Datenbank zur Speicherung des aktuellen Zustands der Verwaltungsschalen und ihrer Teilmodelle nötig. Realisiert werden kann die AAS DB über MongoDB [6] und entsprechende Repositories im MSB BaSyx Interface. Abbildung 16 zeigt die wichtigsten Zusammenhänge und Komponenten bei der Verwendung der AAS DB.

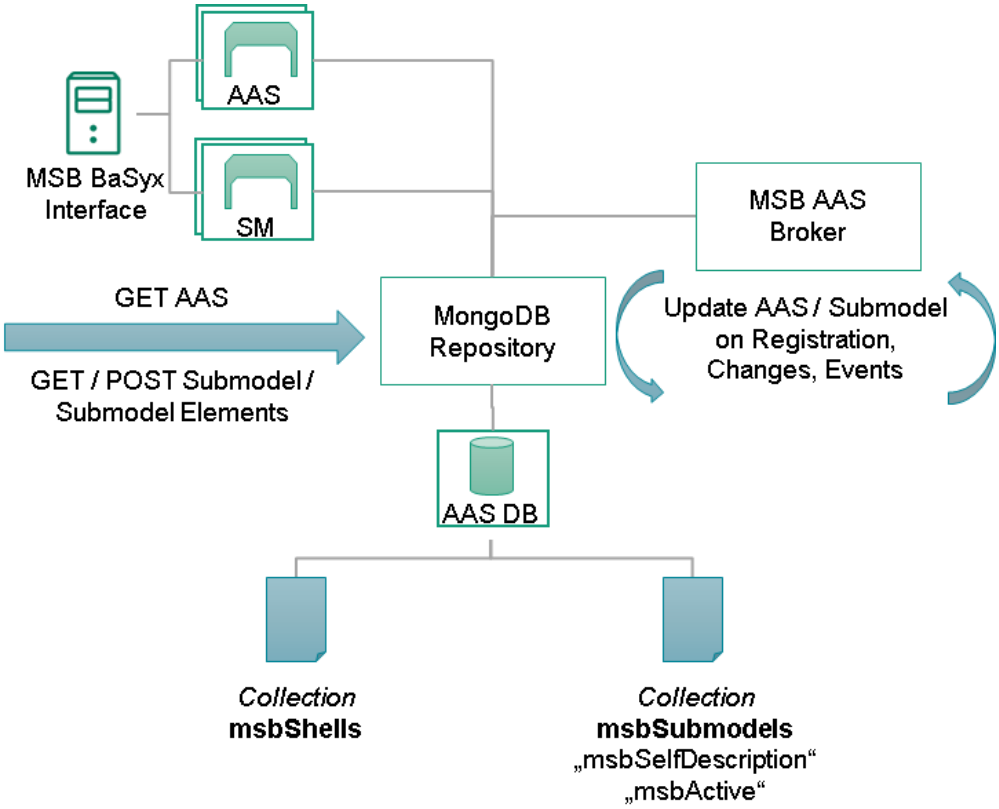

Abbildung 16: Konzeptbild für die AAS DB im MSB BaSyx Interface

Aufrufe über die AAS / Submodel API werden durch die Datenbank bedient und auch ein Schreiben von externen Diensten in die Datenbank ist bei entsprechender Berechtigung möglich. Die Verwaltungsschalen und Ihre Teilmodelle werden bei folgenden Ereignissen erstellt / aktualisiert:

1. Ein neuer Dienst registriert sich am MSB und aktiviert die Verwaltungsschalen-Option: Die Verwaltungsschale wird initial und automatisiert erstellt und gespeichert. Dabei werden das Teilmodell “msbSelfDescription” für die Selbstbeschreibung (Typ TEMPLATE) und das Teilmodell “msbActive” für aktuelle Events (Typ INSTANCE) und ausführbare Operationen bereitgestellt.
2. Ein Dienst aktualisiert seine Selbstbeschreibung: Die Verwaltungsschale und ihre Teilmodelle werden gemäß den Änderungen in der Selbstbeschreibung aktualisiert.
3. Der Dienst sendet ein Event (optional angestoßen durch einen Operations-Aufruf): Das Event (Typ INSTANCE) wird im Teilmodell “msbActive” aktualisiert. Bei einem aktiven Operations-Aufruf wird das Event zusätzlich an den Aufrufer zurückgegeben.

Im nachfolgenden Kapitel wird der übergreifende Prozess der Bereitstellung der Verwaltungsschale und der Teilmodelle weiter beschrieben.

3.5 Bereitstellung der Verwaltungsschalen

In Abbildung 17 werden die wichtigsten Schritte bei der Bereitstellung einer Verwaltungsschale über das MSB BaSyx Interface vereinfacht dargestellt. Dienste, die sich beim MSB registriert haben (1) und darüber Daten austauschen (2 und 3), erhalten ihre Verwaltungsschale durch Aktivieren im MSB Frontend (4). Das MSB Frontend löst daraufhin im MSB BaSyx Interface die Generierung der Verwaltungsschale aus. Die Selbstbeschreibung inkl. Aller Events und Funktionen wird in das Metamodell der Verwaltungsschale konvertiert (5). Die neue Verwaltungsschale wird mit passivem und aktivem Teilmodell im AAS-Store / AAS DB gespeichert und die APIs bereitgestellt (6). Zusätzlich wird eine Subskription auf die Message Queues des MSB-Dienstes erzeugt, um Live-Daten und Kommunikation über die Verwaltungsschale zu unterstützen (7).

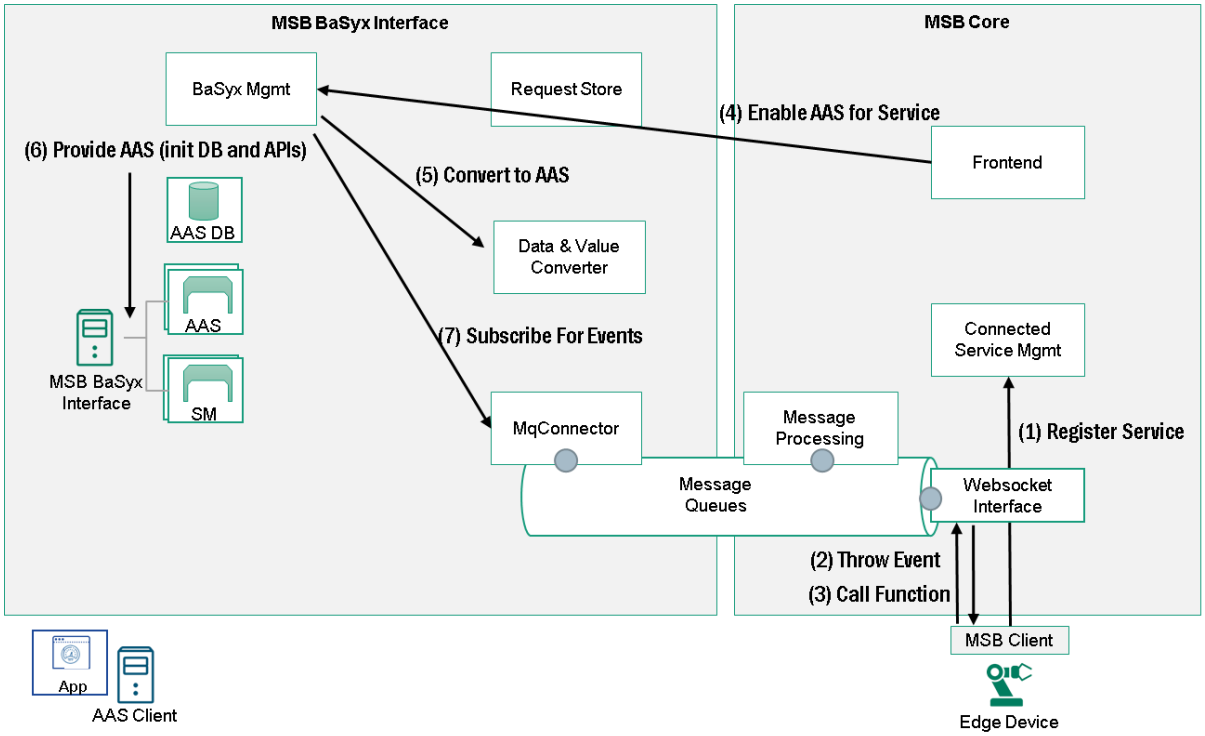

Abbildung 17: Sequenz zur Bereitstellung einer AAS über das MSB BaSyx Interface

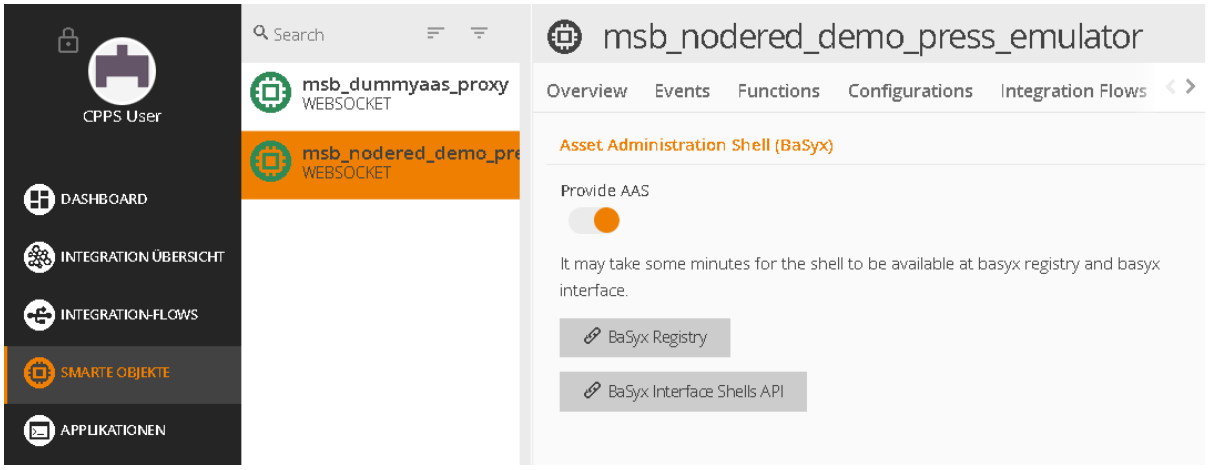

Abbildung 18: Aktivierung der Verwaltungsschale über die MSB GUI

In Abbildung 18 ist ein Ausschnitt aus der MSB GUI dargestellt. Für die „Pressanlage“ kann hier die Verwaltungsschale aktiviert oder deaktiviert werden. Abbildung 19 zeigt das Ergebnis einer über den MSB bereitgestellten Verwaltungsschale. Die beiden Teilmodelle “msbSelf-Description” und “msbActive” mit den entsprechenden Events werden gelistet und Operationen können von Systemen wie der AAS UI [7] aus dem BaSyx Ökosystem aufgerufen werden.

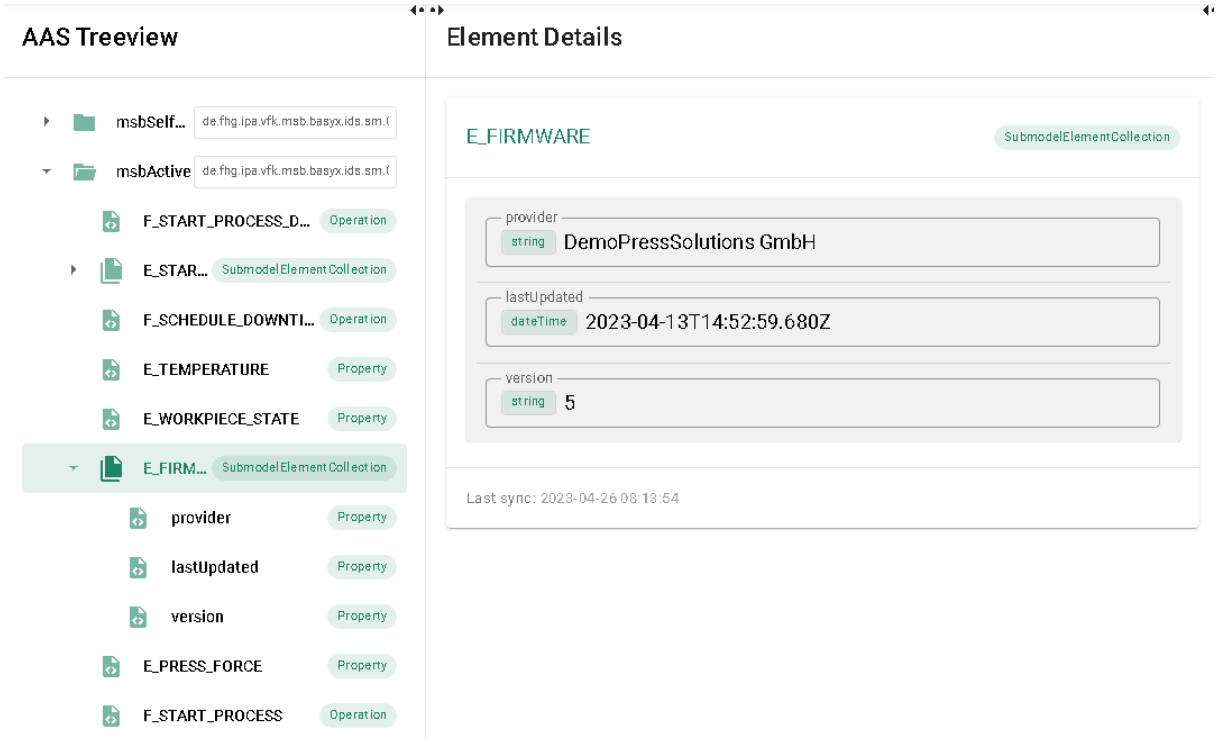

Abbildung 19: Abruf der bereitgestellten Verwaltungsschale über die AAS UI

3.6 Asynchrone Kommunikationsfähigkeit

Operationen können über das MSB BaSyx Interface sowohl synchron als auch asynchron aufgerufen werden. Dabei verbindet das MSB BaSyx Interface das Request-Response-Pattern mit der Event-basierten Kommunikation des MSB. Abbildung 20 zeigt die erste Phase eines asynchronen Aufrufs einer über den MSB bereitgestellten Operation. Initiiert wird der asynchrone Operationsaufruf über eine Applikation mit BaSyx AAS Client (1). Das MSB BaSyx Interface meldet den Start der Operation (2) und erstellt eine Session im Request Store (3). Diese Session ist verknüpft mit der Request-ID des initialen Aufrufs und hilft später bei der Zuordnung und Bereitstellung der zugehörigen Ergebnisse. Anschließend werden die Parameter des Operationsaufrufs aus dem Metamodell der Verwaltungsschale in das Datenformat und einen Funktionsaufruf des MSB konvertiert (4). Der Funktionsaufruf mit den konvertierten Parametern wird über die Message Queue an den Ziel-Dienst übertragen (5). Dabei wird die Request-ID in Form einer „Correlation-ID“ mitübertragen. Die Funktion wird gestartet (6) und die erste Phase der asynchronen Operation ist beendet.

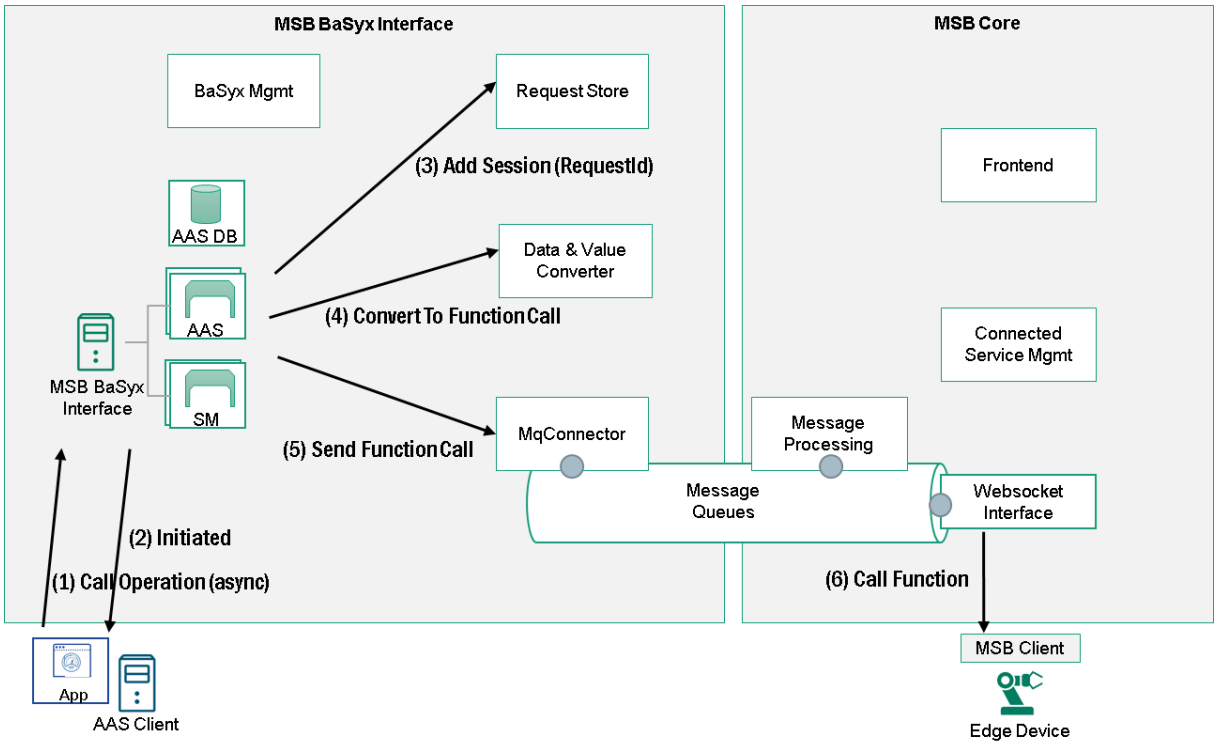

Abbildung 20: Sequenz zur Umsetzung der ersten Phase eines asynchronen Operationsaufrufs

Zu Beginn der zweiten Phase (siehe Abbildung 21) sendet der MSB-Dienst ein Response-Event mit dem Ergebnis der Operation (1). Dieses wird über die Message Queue übertragen und entsprechend der „Korrelation ID“ der passenden Session im Request Store zugeordnet (2). Hier steht das Ergebnis so lange bereit, bis es abgeholt wird oder ein definiertes Timeout abgelaufen ist. Die Applikation mit BaSyx AAS Client ruft nun zu einem späteren Zeitpunkt im Rahmen des Timeout das Ergebnis der Operation aus dem Request Store über die Request-

ID ab (3 und 4). Es erfolgt die Rückkonvertierung in das Metamodell der Verwaltungsschale (5) und die Bereitstellung des Ergebnisses als „Operation-Result“.

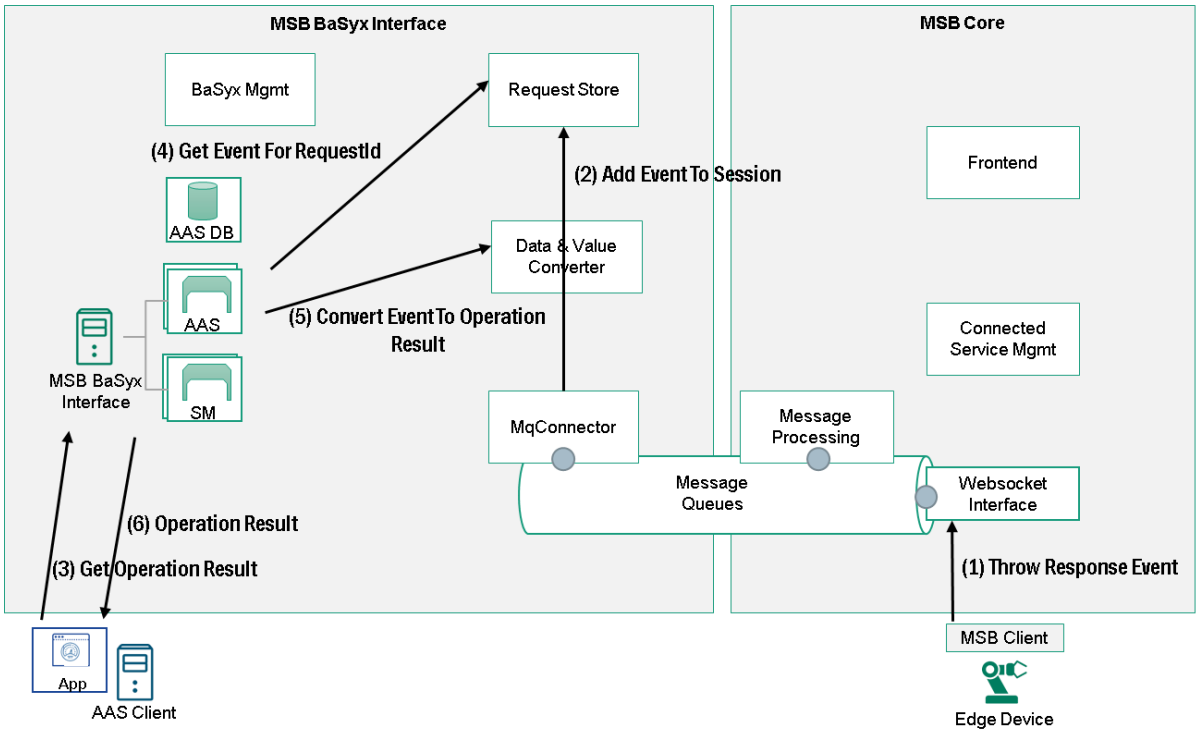

Abbildung 21: Sequenz zur Umsetzung der zweiten Phase eines asynchronen Operationsaufrufs

3.7 AAS Proxy Service

Die allermeisten Systeme haben heute noch keine Integration mit Verwaltungsschalen über eine passende Schnittstelle. Dies betrifft sowohl die Kompatibilität zum Metamodell der Verwaltungsschale als auch die zugehörige API für die Kommunikation mit der Verwaltungsschale. Das Aufsetzen durchgängiger auf Verwaltungsschalen basierender Prozesse ist dadurch mit hohem Integrationsaufwand verbunden. Ein auf Low-Code basierter konfigurierbarer Proxy Service “AAS Proxy Service” kann hier Abhilfe schaffen. Mit dem Proxy Service ist eine einfache Integration von BaSyx Verwaltungsschalen in das MSB Connected Service Management und Workflows möglich. Darüber hinaus kann der Proxy Service mit seiner Low-Code Entwicklungsumgebung aber auch für die Integration von Verwaltungsschalen mit anderen externen Systemen verwendet werden. Das generelle Konzept des Proxy Service ist in Abbildung 22 dargestellt.

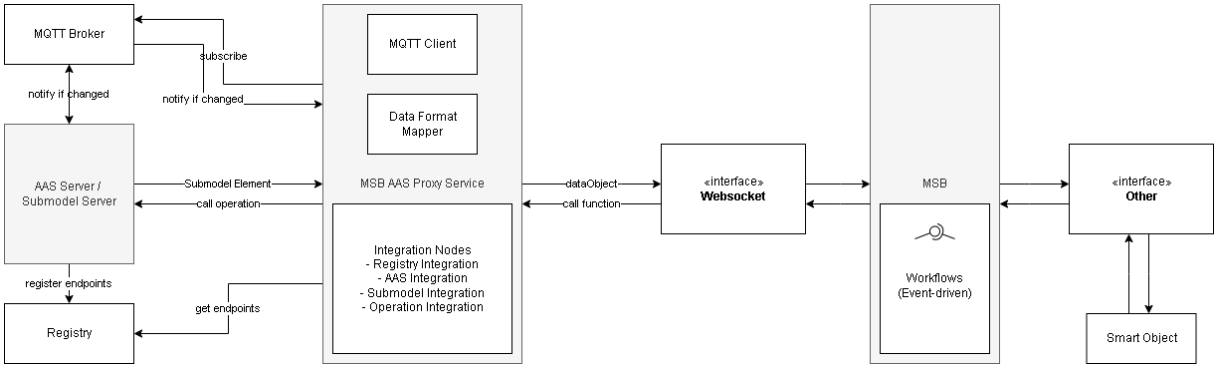

Abbildung 22: Konzeptbild des MSB AAS Proxy Service

Der Proxy Service basiert auf dem Low-Code Entwicklungswerkzeug Node-Red und erweitert dieses durch eine Reihe von eigenentwickelten Bausteinen (Nodes), die über die grafische Benutzeroberfläche zu Flows verbunden werden können. Durch die vorbereiteten Nodes, deren Konfiguration und Verknüpfung können die notwendigen Integrationsschritte so ohne Programmierkenntnisse umgesetzt werden. Der Proxy Service wird im ersten Schritt für eine BaSyx Registry und eine ausgewählte Verwaltungsschale konfiguriert. Damit wird der Zugriff auf die Verwaltungsschale auch bei variablen Endpunkten gewährleistet und alle vorbereiteten Nodes adressieren automatisch den aktuellen Endpunkt der Verwaltungsschale. Eine Übersicht der wichtigsten Nodes und ein Beispiel für die Konfiguration einer Node für das Monitoring eines Wertes innerhalb der Verwaltungsschale ist in Abbildung 23 dargestellt.

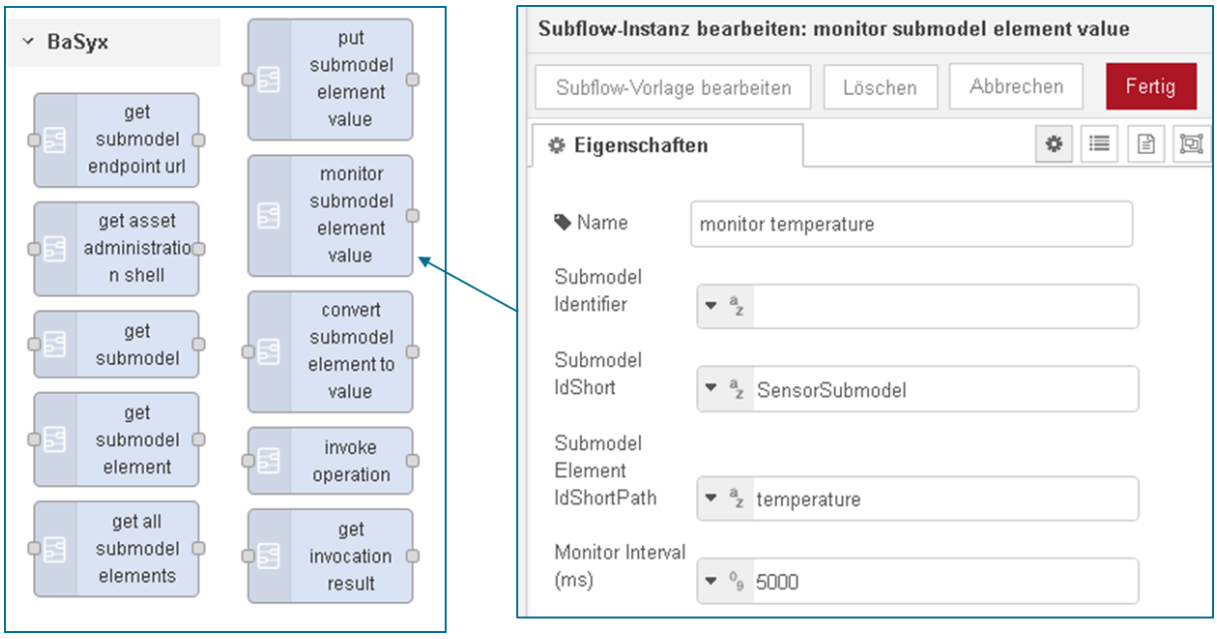

Abbildung 23: Auszug der AAS Proxy Service Nodes

Abbildung 24 zeigt am Beispiel der Node „monitor submodel element value“, wieviel Logik innerhalb der vorbereiteten Nodes in Form eines Subflow gekapselt sein kann, und diese damit als wiederverwendbare Bausteine den Aufwand der Modellierung einer Proxy Service Instanz vereinfachen.

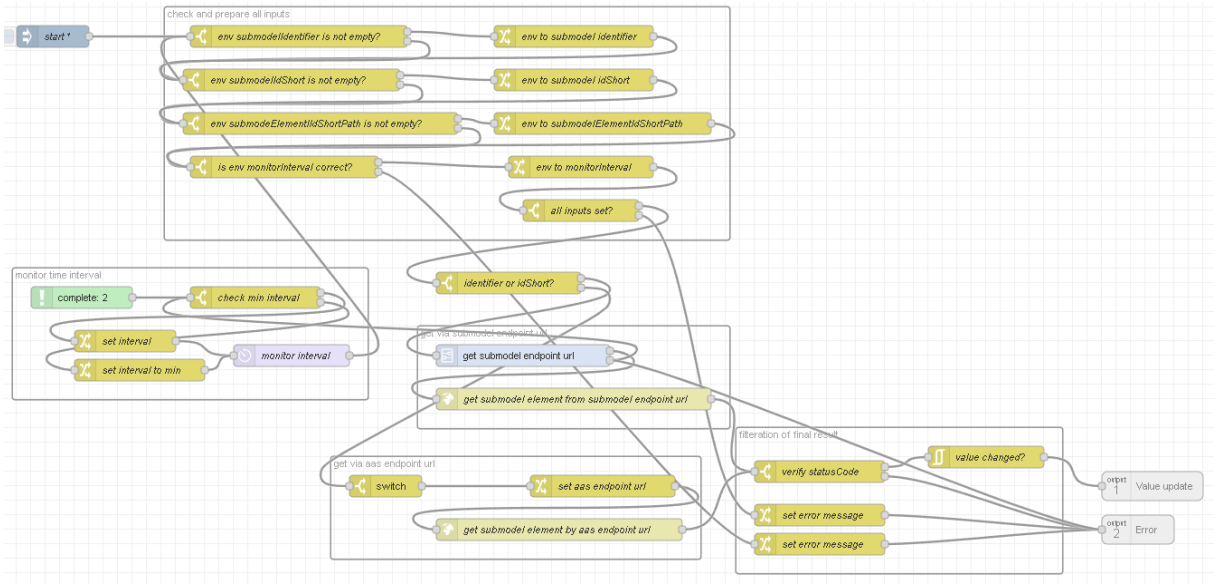

Abbildung 24: Beispiel für einer entwickelte Node des AAS Proxy Service Templates als Low-Code Baustein

Die eigenentwickelten Nodes werden dabei durch eine Vielzahl von Nodes aus der Node-Red Bibliothek ergänzt. Mit den Nodes können nicht nur alle wesentlichen API-Funktionen eines Verwaltungsschalenservers aufgerufen, sondern auch Werte überwacht, Datenobjekte konvertiert und Operationen synchron und asynchron aufgerufen werden. In Kombination mit der Konfiguration der Proxy-Instanz gegen eine Verwaltungsschale lässt sich diese so einfach und grafisch mit einem Drittsystem integrieren. Dabei gibt es zwei Integrationsstrategien:

1. Proxy-Interface: Der Proxy Service integriert alle relevanten Elemente der Verwaltungsschale, bietet eine eigene mit dem Zielsystem kompatible Schnittstelle an und vermittelt die Aufrufe. Die Aufrufe werden dabei meist vom Zielsystem initiiert.
2. Proxy-Mapping: Der Proxy Service integriert alle relevanten Elemente der Verwaltungsschale und die Schnittstellen des Zielsystems. Entsprechende Low-Code Flows initiieren und vermitteln dabei die Kommunikation zwischen beiden Systemen bspw. basierend auf ausgelösten Events oder Werteänderungen.

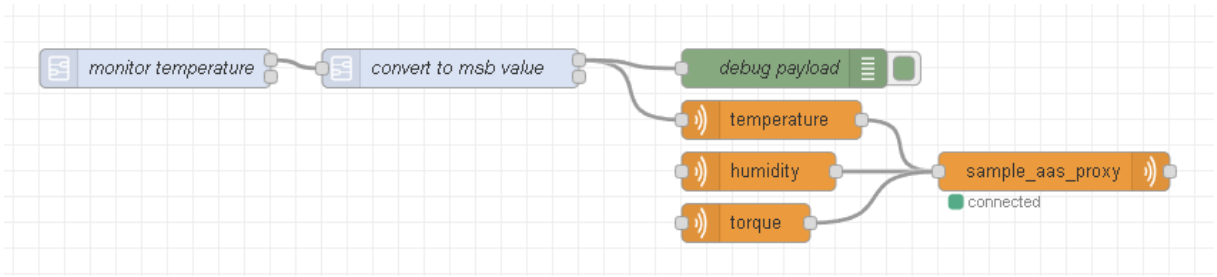

Abbildung 25: Einfache Nutzung des Low-Code Bausteins

Beide Strategien lassen sich kombinieren. In Abbildung 25 ist ein einfacher Flow des Proxy Service zur Vermittlung zwischen Verwaltungsschale und MSB dargestellt. Dabei wird Strategie 2 angewandt. Der Temperatur-Wert der Verwaltungsschale wird überwacht und bei einer Werte-Änderung wird der Flow initiiert. Der Wert wird in das Format des MSB (Zielsystem)

konvertiert und als Event weitergeleitet. So kann der Proxy Service und seine Events in die Workflows des Bus Systems eingebettet werden.

4 Zusammenfassung und Ausblick

Die hier beschriebene Architektur und erweiterten Konzepte zeigen am Beispiel des „Virtual Fort Knox Research – Manufacturing Service Bus“ (MSB), wie ein bestehender Service Bus mit der Verwaltungsschale integriert werden kann. So können Dienste des MSB mit Diensten aus BaSyx einfach und automatisiert verbunden werden. Die Architektur und Konzepte können auf andere Bus Systeme übertragen werden und Basis für deren Integration mit der Verwaltungsschale sein.

Im Rahmen des Teilvorhabens des Fraunhofer IPA des „Verbundprojekt BaSys überProd – BaSys für die Unternehmensübergreifende Produktionsunterstützung“ wird das beschriebene MSB BaSyx Interface und der AAS Proxy Service implementiert, validiert und als wiederverwendbare Industrie 4.0 Dienste bereitgestellt. Damit soll ein wesentlicher Beitrag zur Erweiterung des Verwaltungsschalen-Ökosystems und dessen Operationalisierung in unternehmensübergreifenden Kontext geleistet werden.

Literatur

[1] D. Schel u. a., „Manufacturing Service Bus: An Implementation“, Procedia CIRP, Bd. 67, S. 179–184, Jan. 2018, doi: 10.1016/j.procir.2017.12.196.

[2] „Eclipse BaSy open-source Industry 4.0 middleware“, projects.eclipse.org. https://projects.eclipse.org/projects/dt.basyx (zugegriffen 4. September 2023).

[3] „The C4 model for visualising software architecture“. https://c4model.com/ (zugegriffen 4. September 2023).

[4] „Json Schema Specification“, JSON Schema. https://json-schema.org/specification.html (zugegriffen 4. September 2023).

[5] „Details of the Asset Administration Shell - Part 1“, Plattform Industrie 4.0. https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html (zugegriffen 4. September 2023).

[6] M. M. Eyada, W. Saber, M. M. El Genidy, und F. Amer, „Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments“, IEEE Access, Bd. 8, S. 110656–110668, 2020, doi: 10.1109/ACCESS.2020.3002164.

[7] „basyx-applications“, Eclipse BaSyx. https://github.com/eclipse-basyx/basyx-applications (zugegriffen 4. September 2023).

[8] „Details of the Asset Administration Shell - Part 2“, Plattform Industrie 4.0. https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html (zugegriffen 4. September 2023).

[9] „Spezifikationen der Verwaltungsschale“. https://www.plattform-i40.de/IP/Redaktion/DE/Standardartikel/spezifikation-verwaltungsschale.html (zugegriffen 4. September 2023).

[10] „Specification of the Asset Administration Shel Part 1 Metamodel”, IDTA, IDTA Number 01001-3-0, 2023.