Catena-X

Backward Compatibility Guide 2025

Saturn Release

Version 0.8 – Preview

Table of Content

Disclaimer

The current document version (0.8) is still work in progress. However, it already highlights major areas of impact.

Purposed of this document

Change management / backwards compatibility is a concept that is addressed by multiple different product teams / expert groups. However, compatibility is only given in the tight conceptual integration across components.

This document provides a consolidated view on the backward compatibility of the CX-Saturn release and provides guidance on how Catena-X participants need to prepare themselves for the CX-Saturn release.

Introduction

Backward compatibility is an essential requirement of most productive IT solutions. Especially in complex, multi-enterprise environments, a dedicated release date cannot be fixed across the whole ecosystem. There will not be a centrally coordinated upgrade day for such an ecosystem. Instead, participants may decide to stay on a previous version of any relevant ecosystem component.

Backward Compatibility Motivation

Challenges

Catena-X is an innovation initiative that is highly influenced by the needs and business pains of its members. It does not have a history of decade(s) of development and maturity but is still evolving. At the same time, data providers and data consumers have invested in implemented solutions based on Catena-X and relying on its well-functioning. Business Application Providers, Enablement Service Providers and Core Service Provider have gone through development, testing and certification.

With the next major release (Saturn), Catena-X is pushing new innovations & adjustments into the ecosystem but all parties need to ensure operational stability and investment security to its (productive) stakeholders is given.

As an overall guiding principle we state:
Operational stability for productive participants always outweighs new feature delivery.

Expectations

As the Catena-X ecosystem continues to be adopted by various participants of the automotive supply chain, operational stability is of utmost importance. Companies rely on Catena-X and even physical shipping processes may be negatively impacted by software defects. Therefore, breaking changes are not an appropriate mechanism, therefore we are rather following the "**1+1 Release Schedule**" paradigm.

Specifically, 1+1 Release Schedule means:

- Every participant in the network is allowed to use either the current major release version or the previous major release version (e.g. Jupiter or Saturn)
- To ensure overall CX network stability, there MUST NOT be breaking changes at any time (as it would break the participants' communication)
- Major changes will be introduced via **new major versions** on various levels (API, AAS sub model), not as update into a minor version
- Clear communication of **deprecation date** well in advance is needed if new versions are introduced
  (with introduction of each new major version, it will also be communicated for how long the former versions will be supported)
- All software component versions (provider, consumer side) need to be capable of identifying the version of the remote component they are communicating with and both need to agree on a common denominator to ensure compatibility
- An old software component version that obviously is not aware of a newer version still needs to be capable of communicating with this newer version or an older version that is provided by the communication partner

Backward Compatibility & Certification

With the release of CX-Saturn, CX-Jupiter is still a valid and supported release as elaborated above. However, there are a few implications when it comes to certification:

- After the point of the CX-Saturn release into the community, new solutions cannot be certified against CX-Jupiter anymore. With the release of CX-Saturn, solutions can only be certified against this newest version.
- A solution which has already been certified against CX-Jupiter has to be operable without any changes. A solution certified for CX-Saturn therefore MUST be able to integrate with an unchanged Jupiter certified solution providing the expected Jupiter-like behavior.
- (Optional) A CX-Jupiter certified solution MAY be re-certified against CX-Saturn and the newest set of standards, but the new CX-Saturn certificate MUST assure compatibility with a CX-Jupiter certified solution.
- Only solutions which are certified against CX-Saturn are allowed to advertise with a “CX-Saturn certified” label.

Compatibility per Capability

Technology Capabilities

Sovereign Data Exchange (Data Space Protocol)

Expectation towards Data Exchange according to Catena-X Release Strategy

In the 25.09 release (CX-Saturn) backwards compatibility to the EDC of 24.09 (CX-Jupiter) is expected. This means that data providers have the freedom to decide if they offer already EDC assets in the network based on CX-Saturn release, or if they still provide their offers according to the CX-Jupiter release. The same is true for consumers; they can freely choose if they deploy/operate EDC already based on CX-Saturn or still on CX-Jupiter release. A newer consumer connector needs to be able to initiate an interaction based on the DSP 2025.1 version. However, it needs to also be able to successfully communicate with the data provider if the data provider is still on Jupiter and does not support protocol version negotiation.

How Backwards Compatibility is enabled in Data Exchange

A Saturn based connector MUST provide access via the new DSP protocol 2025-1 as well as the established DSP protocol 0.8 (CX-Jupiter). Depending on the role in the interaction, the connector has to react differently to an exchange partner using a Jupiter-certified stack.

For a new provider connector, the situation is quite easy. A connector MUST implement the different protocol versions in separate endpoint sub-trees, e.g., the protocol version 0.8 has as root path for all DSP endpoint the path https://provider.com/api/v1/dsp, whereas the root path for the 2025-1 DSP endpoints is https://provider.com/api/v1/dsp/2025/1. As the api root path for the old version has not been moved, an old connector naturally calls the old api sub tree and the provider connector can detect the call on the old protocol and act accordingly.

For a new consumer connector, the version to use has to be explicitly stated in the call to the management api. The provider connector has a metadata endpoint that provides information on supported versions and the root path for that version. It is in the apps responsibility to call the connector with the right protocol version parameters. As for an old provider connector, the Saturn app has to choose the old protocol and execute the dsp call explicitly with that version. All new connectors need to provide version metadata, so missing metadata has to be interpreted as an old connector only supporting 0.8 version.

In the reference implementation, there is support for that by a consumer connector functionality, that does the extraction of the version information and that provides the exact parameter to be used by a consumer app, but this is convenience of the reference implementation, not standardized behavior.

Another topic is the compatibility of data offers, the new connector version comes with new features, new policy constraints or the non-finite push transfers. Using these on the provider side could lead to hiccups on the consumer side, as the contract offers might be misunderstood. In the best case, they are filtered out properly on the consumer side, but the behavior of apps is not predictable. To mitigate this risk of unpredictability, the chapter "Usage Policies" gives advise how data offers need to be provided in Saturn based connectors.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

The awareness is only relevant for Saturn-certified participants as they have to act, especially on the consumer side. As is our understanding today, the provider side has to support both versions and act passively according to the called version endpoint. The consumer has to implement version awareness in order to act Saturn-certifiable.

A provider has to consider his consumers in the sense, that he might need knowledge on the opposite sites stack to provide an adequate contract offer, i.e., he should only use new features if he is sure, that the consumer can handle them. If not known, he should stick to, e.g., the old policy setup, or only use existing transfer types from the past

Self-Sovereign Identity

Expectation towards Self-Sovereign Identity according to Catena-X Release Strategy

There are productive participants in the network that have already received a digital identity and various verifiable credentials such as membership credential.

A participant is not actively required to renew / update its digital identity or verifiable credentials. In case of a required technical update on the Core Service Provider side, it will not affect any existing communication that is already setup.

How Backwards Compatibility is enabled in Self-Sovereign Identity

- DCP 1.0 and other requirements from CX-0149 will be adopted by all existing and new wallets, hence no backward compatibility issues
- Backwards compatibility to Jupiter-Connectors ensured by requiring wallets to support DCP v0.8.1 presentation flow (in addition to v1.0)
- No changes to DIDs
- (need to add Connector endpoint to DID doc to use “new connector discovery flow”)
- (DID docs will need to be updated with R25.12)

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

- Not applicable

BDRS & Connector Discovery Service

Expectation towards BDRS according to Catena-X Release Strategy

Today, Catena-X is heavily relying on the business partner number (BPN) for business identification as well as for technical routing. However, this close coupling has turned out to be a suboptimal design decision. In CX-Saturn, BPN will be removed from the EDC/DSP layer, which will work on DIDs rather than BPNs. Consequently, this has an effect on routing and discovery functionality as it was originally. With Saturn, we still need to support participants, which communicate based on the old paradigm, while at the same time enable other participants to work with the new approach.

How Backwards Compatibility is enabled in BDRS & Connector Discovery

The BDRS has not changed between the versions, as his major functionality is mapping BPNLs to DIDs which is stable. The role has changed, i.e., based on the used DSP version a connector will use BPNLs (0.8) or DIDs (2025-1) as basic identifier in DSP messages. The BDRS was introduced for version 0.8 to support the DCP protocol which requires the use of DIDs. With the consistent use of DIDs in all DSP interactions, the need to find out the DID from the BPNL is not required in the connector for the 2025-1 implementation. But it might be, that the App has to find out the DID from the BPNL, so the visibility level is now on App level, not on connector level.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

The changes described in the connector section needed on consumer side have to consider this issue as well. The reference implementation provides the mentioned helper that does the trick, but other implementations need to get the DID for a BPNL, using a call to the BDRS.

Digital Twin Registry - Semantics

Expectation towards Digital Twins according to Catena-X Release Strategy

Participants of the Catena-X data space have already modeled and deployed thousands of digital twins productively. These twins are used in various use cases.

Our participants need to understand how to deal with existing DT and how can their data be exchanged uninterruptedly even in a multi-release environment where certain participants upgrade to a newer digital twin model.

By design, a change of an underlying data model of a digital twin generates a new primary key (SemanticID) of that object, because the specific model version (e.g. 1.0.2) is part of the SemanticID. This even holds, if the Aspect Model itself is backward compatible.

How Backwards Compatibility is enabled in Digital Twins

Supporting two semantic versions of a Submodel will technically mean to create two versions of the DSP asset. However, newer versions of Aspect Models only need to be offered if requested by the corresponding use case standard in the release. It is the responsibility of the use case to decide whether an update to the newest version of an Aspect Models shall be performed or not. Sometimes changing an Aspect Model may also have impact on specific calculation logic and would require new use case implementations. This can only be decided by the use case.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

- Data providers need to determine based on the individual use cases if assets have to be duplicated or not.

Example: two Submodel Descriptors for the different versions of the Aspect Model need to be provided:


```
"submodelDescriptors": [
    {
      "id": "e5c96ab5-896a-482c-8761-efd74777ca97",
      "semanticId": {
        "type": "ExternalReference",
        "keys": [
          {
            "type": "GlobalReference",
            "value":"urn:samm:io.catenax.material_for_recycling:1.1.0#MaterialForRecycling"
          }
        ]
      },
```

```
"submodelDescriptors": [
    {
      "id": "e5c96ab5-896a-482c-8761-efd74777ca98",
      "semanticId": {
        "type": "ExternalReference",
        "keys": [
          {
            "type": "GlobalReference",
            "value":"urn:samm:io.catenax.material_for_recycling:1.2.0#MaterialForRecycling"
          }
        ]
      },
```

Expectation towards Digital Twin Registry according to Catena-X Release Strategy

Participants of the Catena-X data space have already modeled and deployed thousands of digital twins productively. Consequently, those digital twins have been registered into various decentralized Digital Twin Registries across the Catena-X ecosystems.

- ☐ With the CX -Saturn release the DTR interfaces have been updated according to a new Asset Administration Shell (AAS) version 3.1.

From in downward compatibility perspective, access to (under CX-Jupiter) already registered DTR entries and DTs need to be given. Also access needs to be possible with the AAS3.1 interfaces as well as the ones that have been implemented with CX-Jupiter (AAS3.0).

How Backwards Compatibility is enabled in Digital Twin Registry

All APIs from v3.0 are still valid in v3.1, where v3.1 extends v3.0. Since the major version is kept the same and only minor version is changed, no compatibility issues are there.

The DTR (AAS)-version is given in the EDC-catalog to a data consumer calling it. For example : below property can be configured during the asset creation and the value for the property can be set to 3.1, which denotes the IDTA AAS version.

```
"cx-common:version": "3.1"
```

Backward compatibility can also be elaborated by below mentioned points:

1. Semantic Versioning (SemVer)

The IDTA APIs adhere to Semantic Versioning (SemVer). This means:

- Versions are labeled with major.minor.patch.
- Breaking changes increase the major version.
- Non-breaking enhancements or additions bump the minor version.
- **Bug fixes** update the patch number.
  This structured versioning clearly communicates compatibility levels to implementers.

2. Explicit Interface Version Declaration in Endpoints

API endpoints explicitly indicate their version through the interface attribute. For instance, a system might declare support for:

{
"interface": "SUBMODEL-3.0",
"protocolInformation": { ... }
},
{
"interface": "SUBMODEL-VALUE-3.1"
"protocolInformation": { ... }
}

This enables clients to recognize and connect to interfaces that match the version they support, facilitating compatibility while allowing evolution.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

For Data provider relying on Tractus-X reference implementation or yaml-specification only: The only change is that while creating the Shell, now there is a new regex pattern for *IdShort* field contained in the .yaml, and hence the value of IdShort must follow that regex. However, the constraint itself is not new (Constraint AASd-002).

Behavior of reference implementation: The existing/old IdShort values are still valid, as there is no change in GET APIs to read the Shell content.

Business Partner Data Management

Expectation towards BPDM according to Catena-X Release Strategy

Business Partner Data Management (BPDM) is a crucial data space functionality provided by the Core Service Provider B (CSP-B). Various roles, such as the Core Service Providers, Onboarding Service Providers, Business Application Providers as well as Data Providers & Consumers rely on the cleansed business partner data from the Golden Record to uniquely identify the Data Space Participants in a legally secure way and thus establish data exchange contracts between them. An upgrade to the (central) BPDM components in CX-Saturn needs to ensure that none of the communication from the consumers of the API breaks.

How Backwards Compatibility is enabled in Business Partner Data Management

Backward compatibility in BPDM is achieved through a multi-layered approach that allows for a parallel phase of support for major BPDM API versions. This strategy ensures that while new features and improvements are introduced in a new major release, such as CX-Saturn (25.09), existing components relying on the previous release, CX-Jupiter (24.09), continue to function without interruption.

The core mechanism for this backward compatibility is a combination of path versioning and distinct EDC assets. The BPDM Pool and Gate API use a path versioning scheme (e.g., https://<host>/pool/api/v6/participants for the CX-Jupiter release and https://<host>/pool/api/v7/participants for CX-Saturn), which allows both versions of the API to be accessible simultaneously. Instead of a single asset pointing to a single API, there are two separate assets for the same API. Each asset is configured to point to its corresponding versioned API path.

This distinction is further clarified by the mandatory asset attribute cx-common:version, which indicates the API version of the API. For example, an asset representing the BPDM Pool API for the Jupiter release would have cx-common:version=6, while the asset for the Saturn release would have cx-common:version=7. This attribute allows consumers of the API to discover and connect to the correct API version, enabling the simultaneous operation of both the old and new version within the Catena-X data space.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

For Core Service Provider B, who operates both the BPDM Pool and the dedicated BPDM Gates, enabling backward compatibility means that different versions of the BPDM Pool and Gate API (e.g., CX-Jupiter v6 and CX-Saturn v7) must be operated at the same time. As implemented in the BPDM reference implementation, both API versions are part of the same OpenAPI service instance. The API versions share the same database, which ensures that data is held compatible between the versions, and as a result, data migration is not required. So, the Core Service B Provider manages both versions, ensuring consumers of the API can access consistent information, regardless of which version their application is using.

For all consumers of the API, this strategy is crucial for operational stability. During the parallel phase of the Jupiter and Saturn releases, a consuming party can continue to consume data from the v6 Pool and Gate APIs even as new applications or services are developed or have started to use the v7 Pool and Gate APIs. When updating their software, consumers of the API can simply conclude a new data exchange contract for the asset, which points to the /v7/ endpoint. The cx-common:version attribute provides a clear, programmatic way to identify the correct asset for their needs. This eliminates the risk of a breaking change and allows for a smooth, planned migration of applications from one major release to the next, adhering to the 12-month grace period for updates.

Business (Use Case) Capabilities

API Changes

There is an overall expectation that APIs are being versioned. However, at this point there is no consensus in Tractus-X community to agree on a common versioning approach.

How Backwards Compatibility is enabled in API versioning

Incremental versions of APIs need to be provided at any time when changes to an API signature occur, which are not backward compatible (like mandatory fields, structural changes). Consumers must select the correct API version.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

Data providers must support parallel API versions according to the Catena-X 1+1 release strategy.

Data Consumers must be able to determine the API version that is to be used according to the Catena-X release version that the consumer app is supporting or certified against.

Data Model Adjustments

Expectation towards Data Model adjustments according to Catena-X Release Strategy

Data Models will evolve over time due to new or changing business requirements. However, data models (semantic) is what business applications use for further processing. There, an application that is using a semantic model (data model) from the CX-Jupiter release, MUST NOT be forced to adjust its data model due to a CX-Saturn change. Instead, two applications conceptually need to agree on the least common denominator when it comes to semantic data models.

How Backwards Compatibility is enabled in Data Model adjustments

Supporting two semantic versions of a Submodel will technically mean to create two versions of the DSP asset. However, newer versions of Aspect Models only need to be offered if requested by the corresponding use case standard in the release. It is the responsibility of the use case to decide whether an update to the newest version of an Aspect Models shall be performed or not. Sometimes changing an Aspect Model may also have impact on specific calculation logic and would require new use case implementations. This can only be decided by the use case.

The versioning of the semantic models follow the best practices provided here: https://eclipse-esmf.github.io/samm-specification/snapshot/appendix/model-evolution.html

In overall:

- A patch (0.0.X) only contains corrections related to the semantic descriptions, as typos or required clarification. But no changes of the data model itself. Hence, remains fully compatible with the major version.
- A minor (0.X.0) mainly contains additional fields which are optional. However, the minor changes are always compatible with its major version on the semantical level. For example an Aspect Model v1.1.0 or v1.2.0 MUST always validate also against v1.0.0 or all previous versions within the same major version.

Note:

- This is approach works well on the semantic level. However, the DTR / asset registration always requires the complete version, e.g. v1.0.0. Although, version 1.1.0 is fully compatible, a solution which requests v1.1.0, but only v1.0.0 is provided, no answer is provided, even if there is a valid and compatible version.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

See chapter on digital twins for details.

Access Policies and Usage Policies

Expectation towards Policies according to Catena-X Release Strategy

With release 25.09 (Saturn), the compliance of Access Policies and Usage Policies to the CX-Standards, in particular CX-0152, is validated. Furthermore, the number of pre-defined constraints has been enriched. Therefore, the following scenarios are described to ensure that in every case provider and consumer can successfully negotiate the data access.

1. Data Provider uses a connector certified based on Conformity Assessment Criteria of Saturn Release (short: Saturn connector) and Data Consumer uses a connector certified based on Conformity Assessment Criteria of Jupiter (short: Jupiter Connector)
2. Data Provider uses a Jupiter connector and Data Consumer uses a Saturn connector

How Backwards Compatibility is enabled in Policies

The new constraints introduced with the Saturn release are all optional. The validation rules have also existed before, however, they were not enforced by a validation and thus many policies had errors. For backwards compatibility, this means that a policy, which was valid in Jupiter release is still valid in Saturn release. If the validation shows errors for existing access and usage policies, it means, that the errors have existed before.

Based on the 2 scenarios mentioned at the beginning of this chapter, it should be explained how the backwards compatibility is ensured.

1. Data Provider uses a Saturn connector and Data Consumer uses a Jupiter Connector

If a Data Provider creates a new data offer consisting of access policy and usage policy, he can use the enriched constraints offered in Saturn Release. In this case, the Data Consumer who requests access to connector's catalogue and uses a policy filter will not accept and thus not negotiate the data offer as the contained constraints are not allowed from a Jupiter connector point of view. For this reason, the Data Provider MUST create for every dataset 2 data offers, one having policies with constraints allowed in Saturn release and a second one having policies only using constraints allowed in Jupiter release. As the second Jupiter compliant data offer leaves out some contract constraints defined in the first Saturn compliant data offer, the delta constraints SHOULD be

- either formulated in a referenced contract, e.g. ContractID250620197, which is linked as right operand of the constraint "ContractReference". This would look like

- or are individually listed as single right operands of the constraint “ContractReference” in the following way “{{leftOperand}}/{{operator}}/{{rightOperand}}. This would mean for the newly introduced constraints contained in the Saturn compliant data offer:

```
{
  "leftOperand": "AffiliatesBpnl",
  "operator": "isAnyOf",
  "rightOperand": [
    "BPNL012345678910"
  ]
},
{
 "leftOperand": "DataUsageEndDefinition",
 "operator": "eq",
 "rightOperand": "cx.dataUsageEnd.unlimited:1"
}
```

  the following syntax in the Jupiter compliant data offer:

```
{
  "leftOperand": "ContractReference",
  "operator": "isAllOf",
  "rightOperand": [
    "AffiliatesBpnl,isAnyOf,BPNL012345678910",
    “DataUsageEndDefinition,eq,cx.dataUsageEnd.unlimited:1”
  ]
}
```

to ensure that both data offers have the same legal meaning.

Comment: Enablement Service Providers offering a Saturn connector might offer the feature to automatically generate the Jupiter compliant data offer based on a manually created Saturn compliant data offer.

If the Data Consumer now accesses the catalog of the Data Provider with the 2 alternative data offers for 1 dataset, there are two options:

a) If the Data Consumer uses a filter to only see data offers, which are acceptable based on his own Consumer Policy*, the Data Consumer sees only the data offer based on Jupiter constraints and negotiates it. SUCCESS
b) If no data sovereignty filters are used, the Data Consumer sees both data offers and needs to decide which offer to negotiate. What happens in this case depends on the connector, which is used by the Data Consumer. Either a manual interaction is necessary, to decide which offer should be negotiated or the connector is just negotiating one arbitrary data offer. SUCCESS
   The handling how to act in case of more than 1 data offer, fitting the filter criteria is necessary anyway already in Jupiter connector to Jupiter connector negotiations as it might always happen that several policies for one dataset exist.

2. Data Provider uses a Jupiter connector and Data Consumer uses a Saturn connector

If a Data Consumer requests a data offer from Data Provider's catalog, he sees all data offers as long as they are compliant to the CX-Standards, in particular CX-0152. To ensure that no errors are in the data offers, it is recommended that all Data Providers check that their existing policies are compliant after the Saturn Release is introduced to the Catena-X data space.

What does it mean for Data Provider & Data Consumers (what do they need to do or be aware of)

Non-compliant data offers need to be corrected:

The validation of policies is done if new policies are created. So, if Data Providers continue to use access policies and usage policies, they have created before the Saturn they switched to a Saturn connector, they MUST validate existing policies against the validation offered by a Saturn Connector. In this way, they ensure that there were no errors in existing policies and backwards compatibility is really ensured. In case there are errors, they need to re-create the policy and the data offer.

To ensure that no errors are in data offers after the introduction of the Saturn Release to the Catena-X data space, it is recommended that all Data Providers check that their existing policies are compliant.

2 alternative data offers need to be created for every dataset in Saturn Connector:

Every Data Provider using a Saturn connector MUST create for every dataset two data offers, one using in the policies constraints allowed in Saturn release and a second one only using in the policies only the subset of constraints allowed in the Jupiter release.

Business Applications

Business applications are provided in Catena-X by various different software vendors. There is no a singular time frame where application providers (re)-certify their applications for the Saturn release. Hence, all applications that are applying Saturn concepts need to be capable to integrate with an interoperable solution of a different software vendor, which is still only compliant to the Jupiter standards.

General Assumption:

A business application changes from Jupiter to Saturn Standard. In this cases, we shall mandate that this business application also requires a Saturn EDC on its side (Backward compatibility on protocol level will be ensured by EDC)

Demand & Capacity Management

- Interface has not been changed
- New status codes (error codes) have been added (improved error handling)
- Manly clarifications added
- Could potentially lead to incompatibilities

Product Carbon Footprint

- Changes in data model (mandatory / optional fields)
- Fields have been moves in the structure as well as renamed
- Currently considered to be not backward compatible

Digital Product Port

- New Aspect Model Version (changes in Data Model)

Engineering Use Cases

- New in Saturn, no compatibility to Jupiter needed

Logistics

- New in Saturn, no compatibility to Jupiter needed

Quality Management

- Added and updated data models
- Backward compatibility most likely not given

Puris

- Will stay with Jupiter, will not be CX-0018 Saturn compatible

Company Certificate Management

- Old standard was too vague and was leading to misinterpretation
- Standard could not be changed, but a recommendation will be added to the standard
- API changes to be considered not backward compatible

Technical Overview on Compatibility

Compatibility is based on the orchestration of various different APIs and concepts as outlined in this document.

The following diagram (WIP) is going to visualize these overall concepts.