Developer Guide for IBM ConcertDef v1.0.2

Resources

- IBM Concert Documentation
  - https://www.ibm.com/docs/en/concert
- Sample ConcertDef files
  - https://github.com/IBM/Concert/tree/main/use-case-data-files/concertdef-samples
- IBM Concert Toolkit
  - https://www.ibm.com/docs/en/concert?topic=methods-using-concert-toolkit
- Sample YAML template files supported by the *Concert Toolkit*
  - https://github.com/IBM/Concert/tree/main/toolkit-enablement/concert-sample/templates
- *Developer Guide* for the *ConcertDef Schema*
  - https://github.com/IBM/Concert/tree/main/toolkit-enablement/concert-utils/concertdef_schema
- The command below shows the contents of the Build Inventory SBOM schema in use by the instance at "myhost.com:1234" supporting API key type "C_API_KEY". The API path is "**/core/api/v1/concertdef/schema**" and query parameter for **sbom_type** is **build** (which can be **deploy** or **app**). <key> and <id> are the API key and instance id in use, respectively. The output can be saved into a local file via the command option "-o".

```
curl -k 'https://myhost.com:1234/core/api/v1/concertdef/schema?sbom_type=build'\
   --header 'Authorization: C_API_KEY <key>' --header 'InstanceId: <id>'
```

- The command below shows how schema validation for a ConcertDef SBOM can be performed via the Concert service in use.

```
curl --location 'https://myhost.com:1234/core/api/v1/validate_concertdef_sbom' \
   --header 'Accept: application/json' \
   --header 'Authorization: C_API_KEY <key>' \
   --header 'InstanceId: <id>' \
   --header 'Content-Type: multipart/form-data' \
   --header 'Content-Type: multipart/form-data' \
   --form 'filename=@"/sample/concert-sample-devsecops-build.json"'
```

Documentation Modification Date: 2025-02-20

This edition applies to the IBM Concert Defined (ConcertDef) Schema v1.0.2.

Contents

1 Overview

IBM Concert promises automatically generating "*Application 360 Insights*" when sufficient data is loaded into it. Sample factual data regard software development lifecycle tasks, *software bill-of-materials* (***SBOMs***), deployment evidences, endpoint certificates, continuous vulnerability scan results, and continuous compliance posture measures.

Concert's core data model comprises three types of *Concert-defined* (***ConcertDef***) data in terms of *evidence inventory types* and application-oriented *evidence-selection blueprints*.

1. ***Build Inventories****,* e.g.,
   - Evidence data for a specific *Container Image* and the *Code Repo* used to build it.
2. ***Deploy Inventories****,* e.g.,
   - Evidence data for a specific Kubernetes cluster and the *Container Images* deployed to it in a specific deployment environment, *whose name is unique in the Concert service in use*.
3. ***Application Blueprints****,* e.g.,
   - Selection criteria for in-scope *Build* and *Deploy Inventories* in terms of their name and version.

For container-based use cases, the following is a summary of the relationships between *Build Inventories*, *Deploy Inventories,* and *Application Blueprints.*

- One ***Build Inventory*** can include only one ***Code Repo***, which can be used to build one or more ***Container Images***.
- One ***Build Inventory*** can include only one ***Container Image*** built via several ***Code Repos***.
- Every ***Deploy Inventory*** is bound with one and only one ***Deployment Environment***, which may comprise several Kubernetes clusters and namespaces.
- Several ***Container Images*** can be deployed to one ***Deployment Environment*** evidenced by one ***Deploy Inventory***.
- One ***Container Image*** can be deployed to several ***Deployment Environments*** evidenced via several ***Deploy Inventories***.
- *One* **Build Inventory** can be associated with several ***Application Blueprints*** per the name and version specified in its metadata.
- *One* ***Application Blueprint*** can select (without details) several ***Build Inventories*** in terms of their (component) names and the versions.

All Build and Inventory SBOMs in Concert are "versioned" and "immutable" once successfully loaded. The "inventory" data serve as "evidence" that can be collected *before or after* a build (or deploy) process is completed. As a result, a Build (or Deploy) Inventory SBOM may not always contain all the evidence for a completed build (or deploy) process. For example, a Build Inventory SBOM may only include a factual reference to the source code (such as a repo URL or a file system root path) used in a pre-build process. In this case, the build process is incomplete, and a deployable artifact has not yet been generated.

An App Blueprint SBOM contains a set of structured Inventory SBOM selectors that enable business-oriented, 360-degree views of selected Build and/or Deploy Inventories. These views integrate other evidence data (e.g., package SBOMs, vulnerability scan results, endpoint certificates) along with business support data (e.g., business organization details, application criticality settings).

These typed *ConcertDef* SBOM files must be formatted in JSON when they are loaded into a specific instance of Concert via the instance's Console GUI or the instance's API service endpoint. The typed JSON files are usually named as "build", "deploy", and "app" SBOMs, respectively.

An App Blueprint SBOM may include a "selector" for in-scope "Build Inventory" components *before* the corresponding Build Inventory SBOMs are loaded into Concert. That means *the selection is dynamic* rather than static at the time of App Blueprint SBOM loading.

A complete *ConcertDef JSON Schema* specification is available at the URL below.
https://github.com/IBM/Concert/tree/main/toolkit-enablement/concert-utils/concertdef_schema
It can be used to check for *syntax* errors of the JSON-formatted ConcertDef SBOMs (via tools like `check-jsonschema`) and to enable schema-assisted editing of the ConcertDef files via modern text editors like `vscode`. Home page of the tool `check-jsonschema` is at https://github.com/python-jsonschema/check-jsonschema
A good how-to summary for schema-based editing in `vscode` is available at:
https://frontaid.io/blog/json-schema-vscode/

One JSON Schema specification (with `description` fields and `examples` objects) is provided for each type of ConcertDef SBOM files. They are more consumable than the complete ConcertDef Schema specification and can detect more type-specific schema errors.

This document provides three sample ConcertDef SBOMs for a sample application named "`concert-sample-devsecops`" in the appendix. Required JSON objects within a specific JSON object are highlighted using colors, e.g., `.bomFormat` is colored because that is required for all the ConcertDef SBOMs.

- Appendix A: Sample Build Inventory SBOM
- Appendix B: Sample Deploy Inventory SBOM
- Appendix C: Sample Application Blueprint SBOM

Every `component` or `service` JSON object in a ConcertDef SBOM must have a `name`.
Every ConcertDef SBOM must have three top-level JSON objects:

- `bomFormat`: It must be a string value "`ConcertDef`".
- *`specVersion`*: It must be a string value "`1.0.2`".
- *`metadata`*: An object of type "`build`", "`deploy`", or "`application`".

Other sample ConcertDef SBOMs are available at:
https://github.com/IBM/Concert/tree/main/use-case-data-files/concertdef-samples

2 ConcertDef JSON Schema

ConcertDef Schema is a *JSON Schema* specification for *ConcertDef SBOMs*. Custom extensions of the JSON-based "syntax" specification can be done via JSON Schema `properties` objects, which are *JSON* `arrays` of two-field objects, each of which comprises `name` and `value` fields. After a specific *ConcertDef SBOM* is loaded into the Concert service in use, "semantic" errors are checked by the service, e.g., a Build Inventory should not include several `container` objects and several `code` objects, though that is not considered as a formatting error. Additionally, the Concert service links the data included in the loaded ConcertDef files with the data in other types of files that Concert supports, such as CycloneDX package SBOMs, CVE vulnerability scans, endpoint certificates, and compliance postures. Finally, the service links all the data with relevant built-in and 3rd-party data, e.g., National Vulnerability Database (NVD) data.

Type-specific ConcertDef Schema specifications can be accessed from the Concert service instance in use via the CURL command. For example, the command below shows the contents of the Build Inventory SBOM schema in use by the instance at "myhost.com:1234" supporting API key type "C_API_KEY". The API path is "**/core/api/v1/concertdef/schema**" and query parameter for **sbom_type** is **build**. <key> and <id> are the API key and instance id in use, respectively. The output can be saved into a local file via the command option "-o".

```
curl -k 'https://myhost.com:1234/core/api/v1/concertdef/schema?sbom_type=build'
--header 'Authorization: C_API_KEY <key>' --header 'InstanceId: <id>'
```

The CURL command below exemplifies how schema validation for a ConcertDef SBOM can be performed via the Concert service in use.

```
curl --location 'https://myhost.com:1234/core/api/v1/validate_concertdef_sbom' \
   --header 'Accept: application/json' \
   --header 'Authorization: C_API_KEY <key>' \
   --header 'InstanceId: <id>' \
   --header 'Content-Type: multipart/form-data' \
   --form 'filename=@"/sample/concert-sample-devsecops-build.json"'
```

JSON-formatted *ConcertDef SBOMs* can be generated via ConcertDef YAML templates. The *Concert Toolkit image* can be used to transform a YAML-formatted ConcertDef template file into a *ConcertDef SBOM*. More details about the IBM Concert Toolkit are available at: https://www.ibm.com/docs/en/concert?topic=methods-using-concert-toolkit

Sample YAML template files supported by the *Concert Toolkit* are available at: https://github.com/IBM/Concert/tree/main/toolkit-enablement/concert-sample/templates

2.1 ConcertDef Build Inventory SBOMs

The top-level JSON objects that are specific to Build Inventory SBOMs are listed below. The JSON object `components` is required, though that is optional for other types of ConcertDef SBOM files.

- `components`: An array of typed *build inventory* objects listed below. It must include at least one `container` or `library` object. In the pre-build context (where evidence is not available for any of those two types of objects), placeholder values can be used to create a `container` or `library` object so that a `code` object with evidence can be included in the Build SBOM.
  - `code`: A structured set of inventory data for a **source code repository**.
  - `container`: A structured set of inventory data for a **container image**.
  - `library`: A structured set of inventory data for a **software library**.
- `properties`: An array of name-value objects.
- `tags`: An optional array of strings that can be used for searching or filtering.

Appendix A shows a sample Build Inventory SBOM with build-specific top-level objects: `metadata` and `components`.

- `metadata`: The `type` field (on line 6) is required, and the value must be `"build"` . The SBOM creation `timestamp` (on line 5) is optional. The `name` (on line 8), `version` (on line 9), and `build-number` (on line 10) are used by Concert to compose an internal unique id for the Build SBOM. ConcertDef Application SBOMs use the `name` and `version` to select this Build *SBOM*.
- *`components`*: This sample exemplifies the use of two container-related objects:
  - `container`: Its `name` (on line 16) is the image path (without tag or digest), which includes the image repository name as the last naming component delimited by character "/". Its `uri` (on line 17) is *unique resource identifier* composed of the *image path*, *image tag*, and/or *image digest*. The format is "<path>:<tag>", "<path>@<digest>" or "<path>:<tag>@<digest>". That is, *image path* (a.k.a., image repository path) is required. An *image tag* labels a specific image instance in the image repository mainly for human consumption, and an *image digest* is a *content-based* version indicator for a specific image instance. When this `uri` field is not included in the SBOM, Concert composes the identifier in terms of the *defined* values of `name`, `tag`, and `digest`. In practice, `latest` is the default tag value when neither tag nor digest is specified, though common best practices do not use that to tag image instances.
  - `code`: Its `name` (on line 23) is associated with its `purl` (on line 24), which refers to the source code repository/filesystem used for creating the `container` object. When a git repository is used for hosting the source code, `commit_shar` (on line 26) is needed to uniquely identify the version of the source code (as shown in this sample). Without `commit_shar`, Concert cannot know which version of the repository was used for the generation of a specific source-code-based CVE vulnerability scan. For non-git use cases, any version string can be used as the value for `commit_shar. branch` (on line 27) is optional. That is used to facilitate human consumption, and cannot be used as a unique version identifier.

`cyclonedx-bom-link` on line 25 is the unique *BOM-Link* of a CycloneDX SBOM. The sample exemplifies how a source-code-based generation of a CycloneDX *Package SBOM* can be associated with a Build Inventory SOM. `cyclonedx-bom-link` would be included in the `container` object if an image-based CycloneDX *Package SBOM* was generated for the container image. In the Concert console, the *component name* associated with the loaded SBOM is `.metadata.component.name` (on line 8).

2.2 ConcertDef Deploy Inventory SBOMs

Appendix B shows a sample Deploy Inventory SBOM. We note that when a Deploy SBOM pertains to a specific ConcertDef environment, independent of the Build Inventory SBOMs of a specific ConcertDef application component, the `.metadata.component` object is not required. However, if it is associated with Build SBOMs, the `component` object must be specified. All the top-level JSON objects listed below are optional.

- `components`: An array of inventory objects for *deployment* repositories
  - `code`: A structured set of inventory data for a *deployment* repository
- `runtime-components`: An array of inventory objects for deployment runtime
  - `kubernetes`: A Kubernetes-based container runtime.
  - `vm`: A VM or server deployment runtime.
  - `zOS`: A zOS deployment runtime.
- `services`: An array of inventory objects for API services
- `dependencies`: An array of dependency relationship objects
  - `ref`: A `bom-ref` which uniquely identifies a `component` or `service` within the SBOM.
  - `dependsOn`: A `bom-ref` array of the dependent components and/or services
- `properties`: An array of name-value objects
- `tags`: An array of strings that can be used for searching or filtering.

The JSON objects listed in Appendix B are detailed below.

- `metadata`: The `type` field (on line 6) is required, and `"deploy"` must be the value. The `environment` field (on line 7) must be set as the deployment environment name, which *must be unique in the Concert service in use*. Concert composes a unique ID for this Deploy SBOM using `name` (on line 9), `version` (on line 10), and `deploy-number` (on line 11). The `name` and `version` are used by an Application SBOM to select such Deploy SBOMs (each of which has its own `deploy-number` value). SBOM creation `timestamp` (on line 5) and `change-request-url` (on line 12) are optional.
- `components`: This sample exemplifies the inclusion of a `code` object for the deployment repo (or file system) that hosts the deployment artifacts (e.g., scripts, Kubernetes manifests, Helm charts, etc.).

The fields `type` (one line 26), `name` (on line 27), and `components` (on line 51) are required for the `runtime-components` object on line 24. Value of the type is `"kubernetes"`. Value of the `name` in this sample is the value of `cluster_id` (on line 39), which is unique in the `cluster_platform` `"roks"` (on lines 36).

The `components` object (on line 51) is an array of `namespace` objects, each of which includes an array of `container` or `library` objects as its `components`. For example, line 54 shows namespace `"cd4concert"` is used for deploying a container image with its URI specified at line 60.

We note that since each `runtime-components` object is bound to a typed deployment runtime, one *Deploy SBOM* can comprise, e.g., the `container` objects deployed to several

deployment runtimes. Moreover, every deployment runtime can include several `namespace` objects, each of which can be used to deploy several container images.

The `services` object (on line 67) exemplifies how the base URL (on line 74) and network exposure (on line 78) of an API service (named on line 70) can be included in a Deploy SBOM. The `dependencies` object (on line 84) relates the `container` object (referenced on line 85) with the API service object (referenced on line 87) via their `bom-ref` values.

2.3 ConcertDef Application Blueprint SBOMs

Appendix C lists two top-level JSON objects that are specific to Application SBOMs:

- `components`: An array of *blueprint* objects for selecting candidate Build Inventories (of the candidate application components) in terms of `name` and `version`. Qualified Deploy Inventories are chosen as well for the selected Build Inventories. Evidence data included in the selected Build and Deploy Inventories are linked together for each *versioned* Application SBOM, i.e., different versions of a specific Application SBOM may select different sets of versioned Build and Deploy Inventories.
- `properties`: An array of name-value objects

We note that, in practice, IT-oriented *microservice* Application SBOMs can be generated through DevOps pipelines without incorporating business-level data or selection criteria for composite microservices. These IT-oriented Application SBOMs can then be loaded into the Concert service alongside business-oriented Application SBOMs.

The JSON objects listed in Appendix C are detailed below.

- `metadata`: The `type` field (on line 6) is required, and `"application"` must be the value. The `name` (on line 8) and `version` (on line 9) are set for the ConcertDef application. Contents of the `business` object are determined by the business manager of the Concert service account in use.
- `components`: An array of the `build` *objects* selected in terms of the `name` and `version` of the candidate Build SBOMs. Inventory details are expected available from the selected Build SBOMs as well as Deploy SBOMs (created *before or after* the Application SBOM is loaded into Concert).
- `properties`: An array of application-level properties, e.g., application criticality.

3 Automated Generation and Upload of ConcertDef SBOMs

ConcertDef Build Inventory and Deploy Inventory data can be automatically gathered and loaded into Concert with high cost-efficiency via CI/CD (Continuous Integration & Continuous Deployment) pipeline/workflow automation scripts. Such automation can be done with insignificant impact to existing pipeline scripts. For example, IBM *DevSecOps-Concert* provides such automation support at its core for container-based ConcertDef SBOMs. The built-in capabilities can be exploited easily by configuring pipeline settings and/or by coding simple inventory data gathering scripts. The steps below exemplify how the automation scope of an existing *IBM DevSecOps* CI pipeline can be extended for Concert. Existing CD and CC (Continuous Compliance) pipelines can be extended similarly. The CC pipelines periodically generate vulnerability scan results to Concert under the model of continuous monitoring.

1. Configure the required and optional Concert-specific pipeline settings
2. Optionally, add JSON-formatted `concert_deploy` file creation scripts for Deploy SBOM generation
3. Run the CI pipeline and, optionally, confirm the uploads via the Concert Console in use

4 Linking ConcertDef SBOM Data with Other Concert Data

ConcertDef SBOMs enable integration of various siloed enterprise data, such as vulnerability scan results, package SBOMs (formatted in CycloneDX), endpoint certificates, and runtime performance and compliance postures:

- **Code based vulnerability CVEs** are linked with *code* objects in **Build Inventory SBOMs**.
- **Image based vulnerability CVEs** are linked with the *container* objects in **Build Inventory SBOMs**.
- **Package SBOMs** sourced from source (or images) are linked with the *code* (or *container*) objects in **Build Inventory SBOMs**.
- **Endpoint certificates** are linked with the HTTPS endpoints (or base URLs) in **Deploy Inventory SBOMs**.
- **Runtime performance and compliance postures** for Kubernetes-like clusters are linked with the *kubernetes* objects in **Deploy Inventory SBOMs**.

We note that when a specific code-based package SBOM is loaded, the associated code repository URL (or pathname of the source tree root) must be provided as metadata such that the uploaded SBOM can be listed in the Console's *software composition* GUI, otherwise only the constituent packages are listed in the GUI. After a specific Build Inventory SBOM is uploaded with BOM-Link references to package SBOMs, the GUI shows the inventory SBOM's (component) name with each of the referenced package SBOMs.

A. Sample Build Inventory SBOM


```
{
  "bomFormat": "ConcertDef",
  "specVersion": "1.0.2",
  "metadata": {
    "timestamp": "2024-12-06T15:23:57Z",
    "type": "build",
    "component": {
      "name": "concert-sample-devsecops",
      "version": "1.0.0",
      "build-number": "204"
    }
  },
  "components": [
    {
      "type": "container",
      "name": "us.icr.io/icr4mcc/concert-sample-devsecops",
      "uri": "us.icr.io/icr4mcc/concert-sample-devsecops:204-main-
a0a7e8eab28b5d3ca7abc4014080cda6309b68fa@sha256:dbd24de377d42d4e4d3a24004d5fb8664d8
7c74ec30ca9d26ed72cee1e7bee57",
      "tag": "204-main-a0a7e8eab28b5d3ca7abc4014080cda6309b68fa",
      "digest":
"sha256:dbd24de377d42d4e4d3a24004d5fb8664d87c74ec30ca9d26ed72cee1e7bee57"
    },
    {
      "type": "code",
      "name": "concert-sample-devsecops",
      "purl": "https://github.ibm.com/concert/concert-sample-devsecops",
      "cyclonedx-bom-link": "urn:uuid:ddec31c8-caae-4bf6-807a-
bbb6e0bc3f1c/1",
      "commit_sha": "a0a7e8eab28b5d3ca7abc4014080cda6309b68fa",
      "branch": "main"
    }
  ]
}
```

B. Sample Deploy Inventory SBOM

```
{
  "bomFormat": "ConcertDef",
  "specVersion": "1.0.2",
  "metadata": {
    "timestamp": "2024-12-06T17:41:43Z",
    "type": "deploy",
    "environment": "acme-test",
    "component": {
      "name": "concert-sample-devsecops”,
      "version": "1.0.0"
      "deploy-number": "190",
      "change-request-url": "https://us-south.git.cloud.ibm.com/rong/concert-sample-devsecops-change/issues/101"
    }
  },
  "components": [
    {
      "type": "code",
      "name": "concert-sample-devsecops-deployment",
      "purl": "https://github.ibm.com/rong/concert-sample-devsecops-deployment",
      "commit_sha": "c637635161906c6c7c4bfa104d83f98c176e9892",
      "branch": "master"
    }
  ],
  "runtime-components": [
    {
      "type": "kubernetes",
      "name": "cjsal64w0g8rl335fso0",
      "api-server": "https://172.20.0.1:2040",
      "properties": [
        {
          "name": "platform",
          "value": "ibmcloud"
        },
        {
          "name": "cluster_platform",
          "value": "roks"
        },
        {
          "name": "cluster_id",
          "value": "cjsal64w0g8rl335fso0"
        },
        {
          "name": "cluster_region",
          "value": "us-east"
        },
        {
          "name": "cluster_name",
          "value": "roks1"
        }
      ],
      "components": [
        {
          "type": "namespace",
          "name": "cd4concert",
          "components": [
```

```
            {
              "bom-ref": "container:us.icr.io/icr4mcc/concert-sample-devsecops",
              "type": "container",
              "name": "us.icr.io/icr4mcc/concert-sample-devsecops",
              "uri": "us.icr.io/icr4mcc/concert-sample-devsecops:204-main-a0a7e8eab28b5d3ca7abc4014080cda6309b68fa@sha256:dbd24de377d42d4e4d3a24004d5fb8664d87c74ec30ca9d26ed72cee1e7bee57"
            }
          ]
        }
      ]
    }
  ],
  "services": [
    {
      "bom-ref": "appapi:concert-sample-devsecops",
      "name": "concert-sample-devsecops",
      "properties": [
        {
          "name": "base_url",
          "value": "https://concert-sample-devsecops-service-cip-route-cd4concert.roks1-b12d73cc7b0aedf0e30addbf16d8fc5a-0000.us-east.containers.appdomain.cloud"
        },
        {
          "name": "network_exposure",
          "value": "private"
        }
      ]
    }
  ],
  "dependencies": [
    {
      "ref": "container:us.icr.io/icr4mcc/concert-sample-devsecops",
      "dependsOn": [
        "appapi:concert-sample-devsecops"
      ]
    }
  ]
}
```

C. Sample Application Blueprint SBOM

```
{
  "bomFormat": "ConcertDef",
  "specVersion": "1.0.2",
  "metadata": {
    "timestamp": "2024-12-06T15:23:57Z",
    "type": "application",
    "component": {
      "name": "concert-sample-devsecops",
      "version": "1.0.0"
    },
    "business": {
      "name": "Acme Inc.",
      "units": [
        {
          "name": "Unit 1",
          "email": "myemail@acme.com",
          "phone": "(123) 123-1234"
        }
      ]
    }
  },
  "components": [
    {
      "type": "build",
      "name": "concert-sample-devsecops",
      "version": "1.0.0"
    }
  ],
  "properties": [
    {
      "name": "application_criticality",
      "value": "3"
    }
  ]
}
```