IoT Plug & Play
Other design considerations

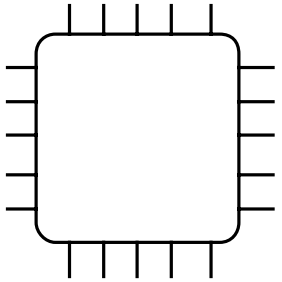
Other design considerations

Model Repository

Where are capability models stored?

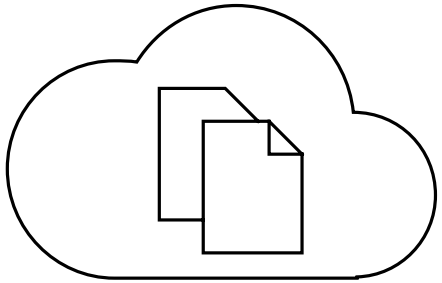
Device sends capability model ID and version expected for the solution to know

If unknown, the following are the model retrieval options for the solution:



Device Sent

Stored and sent by the device to the solution. Quick and easy but device must be updated if model changes

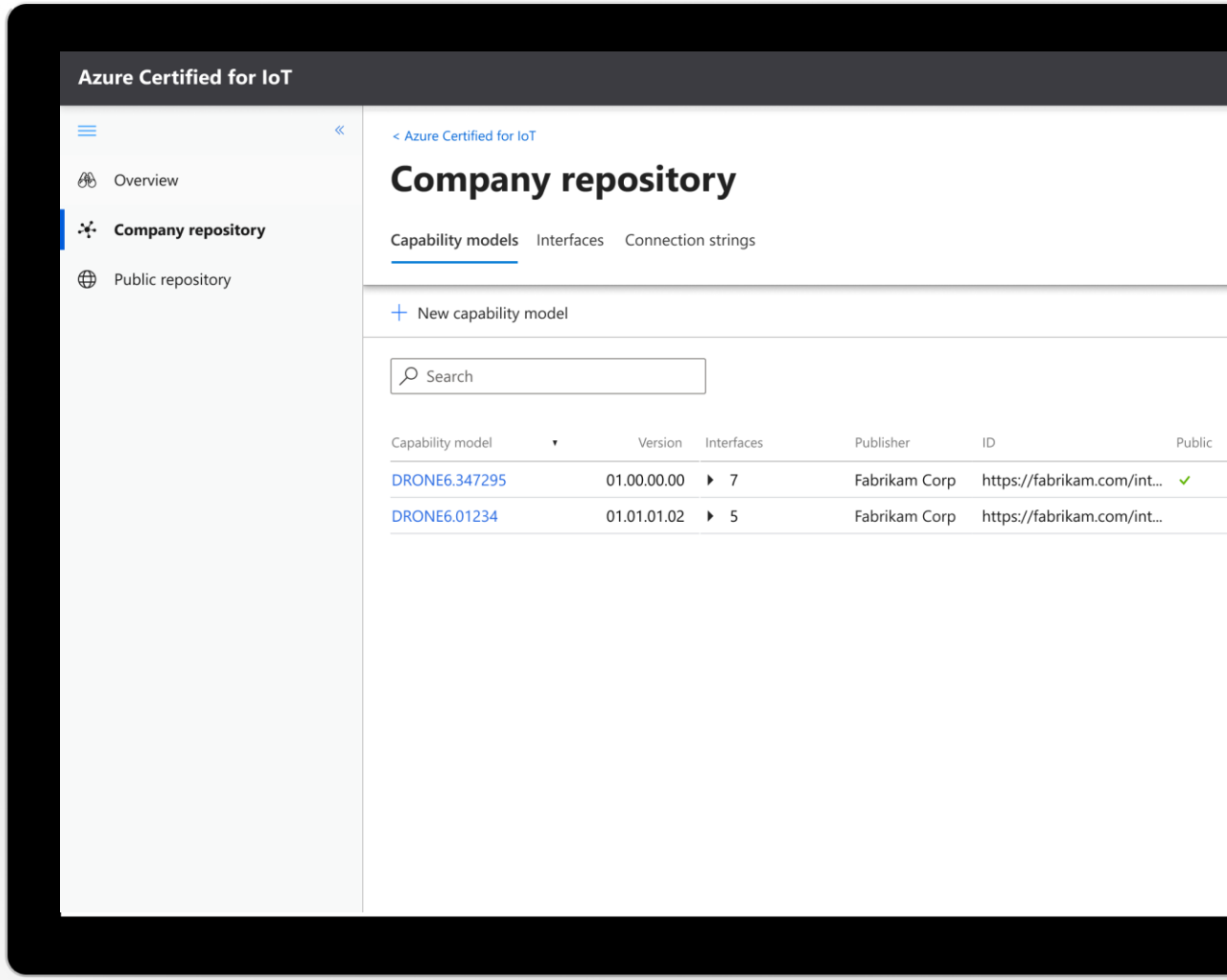


Capability Model Repository

Can be pre-cached by Azure solutions. Includes publish-time validation/versioning and integration with Azure dev tooling

IoT Plug and Play Repository*

- Capability model and interface workspace and publishing repository experience
- Publishing integrated in VS Code and Azure CLI for both interfaces and capability models
- Automated validation, collision checks and versioning support
- Search, filter, sort, view models & their graphs in model repository UX
- Works out-of-the-box with any Azure IoT solution
- Will be made available as an open-source project
- Microsoft will also host a fully managed, multi-tenant instance for always up-to-date for Azure Certified devices; integrated into certification flow



**Model repository is never required for IoT Plug and Play*

Other design considerations

Device Discovery

Device discovery

Purpose:

- Identify the capability model ID and interface IDs supported by a device
- Identify the interface definitions

Model definition locations

- Device
- Global model repository (anonymous public access)
- Organization model repository (connection string shared by the organization)
- Future: URI endpoint

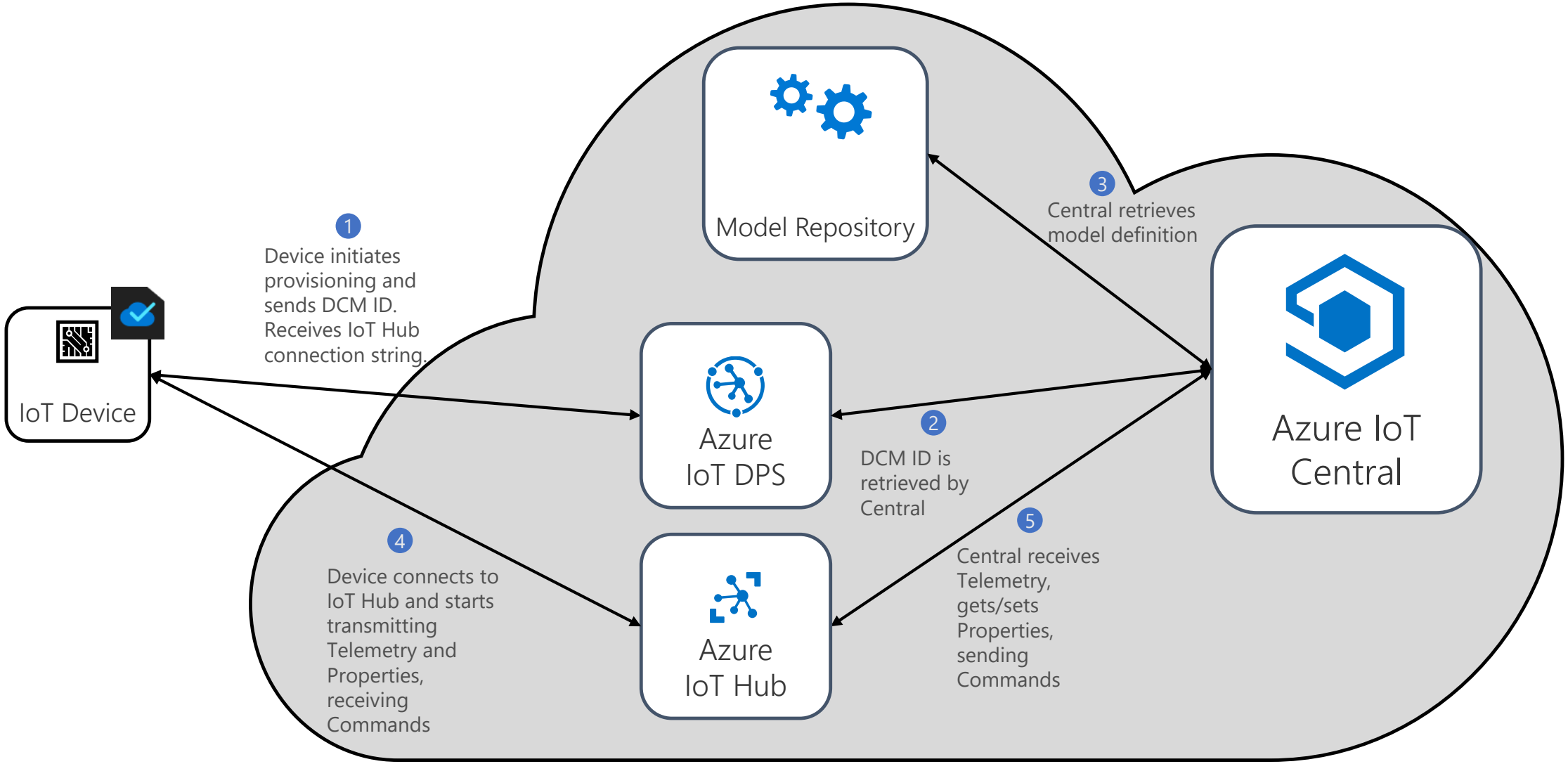
Model discovery - basics

- When an IoT Plug and Play device first connects to your IoT hub, it sends a model information telemetry message.
- This message includes the IDs of the interfaces the device implements.
- For your solution to work with the device, it must resolve those IDs and retrieve the definitions for each interface.

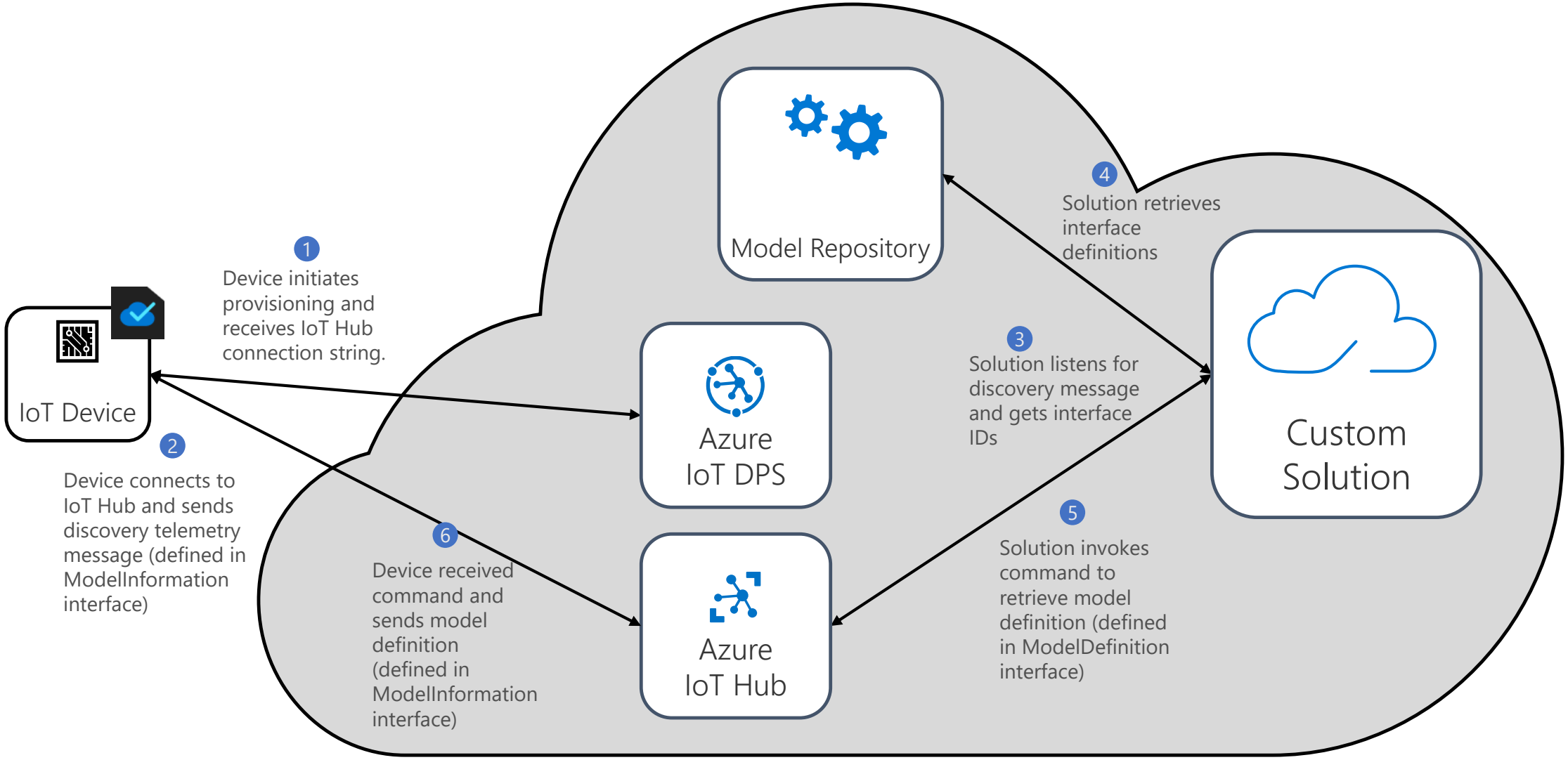
Model Discovery

- Purpose-built IoT solutions
  - A purpose-built IoT solution works with a known set of IoT Plug and Play device capability models and interfaces.
- Model-driven IoT solutions
  - A model-driven IoT solution can work with any IoT Plug and Play device.
  - To build a model-driven IoT solution, you need to build logic against the IoT Plug and Play interface primitives: telemetry, properties, and commands.
  - Your solution must also subscribe to notifications from the IoT hub it uses.

IoT Plug and Play device discovery in Central via DPS



IoT Plug and Play device discovery for a custom solution via IoT Hub



Other design considerations

Common Interfaces

IoT Plug and Play Preview common interfaces

- All IoT Plug and Play devices are expected to implement some common interfaces.
- Certification requires your device to implement several common interfaces.
- You can retrieve common interface definitions from the public model repository.

Name	ID	Description	Implemented by Azure IoT SDK	Must be declared in capability model
Model Information	urn:azureiot:ModelDiscovery:ModelInformation:1	For devices to declare the capability model ID and interfaces. Required for all IoT Plug and Play devices.	Yes	No
Digital Twin Client SDK Information	urn:azureiot:Client:SDKInformation:1	Client SDK for connecting the device with Azure. Required for certification	Yes	No
Device information	urn:azureiot:DeviceManagement:DeviceInformation:1	Hardware and operating system information about the device. Required for certification	No	Yes
Model Definition	urn:azureiot:ModelDiscovery:ModelDefinition:1	For devices to declare the full definition for its capability model and interfaces. Must be implemented when model definitions aren't hosted in a model repository.	No	Yes
Digital Twin	urn:azureiot:ModelDiscovery:DigitalTwin:1	For solution developers to retrieve the capability model ID and interface IDs for a digital twin. This interface isn't declared or implemented by an IoT Plug and Play device.	No	No

Other design considerations

Quotas & Throttling

Limits & throttles

Limits & restrictions	Value
# of Private Model Repositories per AAD tenant	1
# of auth keys per repo	10
# of models (DCMs or Interfaces) per Private Model Repository	1500
# of DCMs or Interfaces that can be registered per Hub and resolved	1500
# of models (DCMs or Interfaces) in the Public Repository per AAD tenant	1500
# of Interfaces that can be registered per device	40
# of DCMs that can be registered per device	1
# of Interfaces per DCM (excluding standard interfaces)	30
# of capabilities (Properties, Telemetry & Commands) per Interface	30
Max size of DCM & interface URI	512 bytes

Throttles	Value
# of DCM or Interface being created/Updated in a private or global repository per second	10
# of DCM or Interface being deleted in a private repository per second	10