Azure Main Technology Features

Services

Deployment of the hybrid solution providing interoperability of the **machine-centric** cyber-physical systems designed atop of M2M reactive communication and emerging cloud computing as a **human-centric** front-end requires decisions addressing the selection of the services supporting web user interface capable to expose real-time process data. In this context, the service is any autonomous (with own identity) software component or module that is interfacing with selected cyber-physical systems for data collection, analysis, and also remote control. Microsoft Azure is a cloud-based product. It offers a vast variety of services. This virtual environment handles an unlimited number of users and devices organized using a solution concept. The solution aggregates users, devices, services, and required additional resources scoping on a selected scenario. The solution is a region that provides a scope to the identifiers (the names of devices, users, process data entities, etc) inside it. Solutions are used to organize deployment entities into logical groups and prevent identity collisions.

The `IoT Central` service provides a process data visualization user interface. To make this interface meaningful metadata called device template is used to describe devices.

Following the assumption that interconnection between the cyber-physical system and cloud services is designed based on the gateway concept, a middleware must be considered aa a coupler. It must be interconnected with the cyber-physical system using an in-bound protocol adhering to communications requirements (i.e. protocol profile, data encoding, time relationships, etc.) governing communication of the parts making it up. At the same time, it must support back-and-forth data transfer to the cloud using out-of-bound native for the cloud services. The transfer process requires data conversion from source to destination encoding. The `IoT Hub` is a service hosted in the cloud that supports `IoT Central` providing a robust messaging solution - it acts as a central message hub for bi-directional communication. This communication is transparent, i.e. it is not data types aware allowing any devices to exchange any kind of data. This service is responsible to manage the devices' identity and it offers the following protocol stacks: AMQP, MQTT, HTTPS.

Before process data can be exposed using a web user interface the data source must be associated with an appropriate solution and validated to make sure that the security rules are not violated. It is hard to assume that the security rules governing the cyber-physical system may also apply to the cloud-based services. In the gateway scenario, they can be mapped on each other or entirely independent. The `IoT Hub Device Provisioning Service` (DPS) is a helper service for `IoT Hub` that enables devices registration, authentication, and authorization of the requested operations including but not limited to data transfer updating the user interface.

It is worth stressing that interaction of the offered by the Azure services can be configured flexibly, and as a result, the presented above selection of services must be recognized as an example only. The ``IoT Central` can be also seamlessly integrated with other services as needed. The following services could also be considered to build cloud-based automation solution:

- `Industrial IoT` - discovering OPC UA enabled servers in a factory network and register them in Azure IoT Hub implemented using `IoT Edge`
- `Digital Twins` - managing the graph of digital twins, which are to represent some real-world process or entity

`Industrial IoT` promotes OPC UA client/server archetype used to achieve direct and interactive interoperability implemented using `IoT Edge` services that allow extracting initial data processing to local premises based on the edge concept. `Digital Twins` is an emerging concept to use an observer to replicate selected process state and behavior. The possibility to add value as a result of using these services must be subject to further research.

Data Interchange

System components interoperability means the necessity of the information exchange between them. The main challenge of interoperability implementation is that information is abstract – it is knowledge describing the process in concern state and behavior, e.g. temperature in a boiler, a car speed, an account balance, etc. Obviously, abstraction cannot be processed by the cyber-physical machines. It is also impossible to transfer abstraction from one place to another over the network.

Fortunately, computer science offers a workaround to address that impossibility - the information must be represented as a binary stream. In consequence, we can usually use both ones as interchangeable terms while talking about ICT systems. Unfortunately, these terms must be precisely observed in the context of further discussion, because we must be aware of the fact that the same information could have many different but equivalent representations. In other words, the same information can be represented by a vast variety of different binary patterns. For example, numbers may be represented using 2's Complement and Floating-Point binary representations.

It should be nothing new for us, as it is obvious that the same information printed as a text in regional newspapers in English, German, Polish, etc. does not resemble one another, but the text meaning should always be the same. To understand a newspaper we must learn the appropriate language. To understand the binary data we must have defined a data type – a description of how to create an appropriate bits pattern (syntax) and rules needed to assign the information (semantics), i.e. make any correct bits stream meaningful. Concluding, to make two systems interoperable, a semantic-context must be established. The type plays the role of metadata, a set of data that describes other data. Metadata term is frequently used if the semantic-context is defined using a native language to select built-in types engaging a general-purpose graphical user interface.

Using the data type definitions to describe information interchanged between communicating parties allows:

- Development against a type definition of the user interface
- Implementation of the functionality of the bits-streams conversion in advance

Having defined types in advance, a gateway may provide dedicated conversions functionality, i.e. replacing bits-stream used by the cyber-physical system by equivalent once for the cloud-based services. The Azure offers a vast variety of build-in types ready to be used in common cases, but not necessarily there are equivalent counterparts in use by the cyber-physical system. Additionally, the data conversion must address the following issues:

- usually to covert data from source to destination representation, the middleware software native types must be used
- if the out of the box set of types is not capable of fulfilling more demanding needs custom data types may be defined

Although the data conversion is a run-time gateway task the implementation of the conversion algorithms must be recognized as an engineering task, and therefore this topic is not considered for further discussion.

In `IoT Central` a cyber-physical system is represented as a set of devices. The characteristics and behaviors of each of a device kind are described by the device template. This Device Template (DT)contains also metadata describing the data (called telemetry) exchanged over the wire with the cyber-physical system called Device Capability Model (DCM). Additionally, the DT contains properties, customization, and views definitions used by the service locally. As an option, DCM expressed as a JSON can be imported into a Device Template. `IoT Central` allows also to create or edit a DCM using the dedicated web UI. A JSON file containing DCM can be derived from an information model used as a foundation to establish the semantic-context applied to achieve interoperability of the devices interconnected as the cyber-physical system. DCM development against any external information model is a design-time task and should be supported by dedicated development tools. In any case, the data interchanged between the cloud and the gateway must be compliant with the DCM, hence the middleware software must be aware of conversions that must be applied to achieve this interoperability.

Connectivity

From the cloud side, it is proposed to employ the `IoT Hub` service to handle the network traffic targeting the cyber-physical system. This service offers profiles of the AMQP, MQTT, HTTPS protocol stacks. In any case, process data (telemetry) is transparently transferred back-and-forth to the upper layer `IoT Central` service. Hence, the payload formatting is determined by the DCM associated with the `IoT Central` solution. All the mentioned protocols are standard ones. Consequently, it is possible to apply any available implementation compliant with an appropriate specification to achieve connectivity. In this case, all parameters required to establish connectivity and security-context is up to the external software responsibility. Alternatively, the API offered by the dedicated libraries may be used. Using this API the configuration process can be reduced significantly. Using these libraries, the selection of the communication protocol has an indirect impact on the interoperability features, including performance. The connectivity with `IoT Hub`, for example, can be obtained using:

- `Microsoft.Azure.Devices` - Service SDK for Azure IoT Devices
- `Microsoft.Azure.Devices.Client`- Device SDK for Azure IoT Hub
- `Microsoft.Azure.Devices.Provisioning.Client` - Provisioning Device Client SDK for Azure IoT Devices
- `Microsoft.Azure.Devices.Provisioning.Transport.Amqp` - Provisioning Device Client AMQP Transport for Azure IoT Devices
- `Microsoft.Azure.Devices.Provisioning.Transport.Http` - Provisioning Device Client Http Transport for Azure IoT Devices
- `Microsoft.Azure.Devices.Provisioning.Transport.Mqtt` - Provisioning Device Client MQTT Transport for Azure IoT Devices
- `Microsoft.Azure.Devices.Shared` - Common code for Azure IoT Device and Service SDKs