European Plastics and Rubber Machinery

EUROMAP

EUROMAP 79	OPC UA interfaces for plastics and rubber machinery – Data exchange between injection moulding machines and robots

Release Candidate 1.00.00, 2021-11-08

EUROMAP 79 (Release Candidate 1.00.00) is identical with

OPC 40079 (Release Candidate 1.00.00) and VDMA 40079:2022-01

Contents

Figures

Tables

Foreword

OPC UA is a machine to machine communication technology to transmit characteristics of products (e.g. manufacturer name, device type or components) and process data (e.g. temperatures, pressures or feed rates). To enable vendor unspecific interoperability the description of product characteristics and process data has to be standardized utilizing technical specifications, the OPC UA companion specifications.

This specification was created by a joint working group of the OPC Foundation and EUROMAP. It is adopted identically as VDMA Specification.

EUROMAP

EUROMAP is the European umbrella association of the plastics and rubber machinery industry which accounts for annual sales of around 13.5 billion euro and a 40 per cent share of worldwide production. Almost 75 per cent of its European output is shipped to worldwide destinations. With global exports of 10.0 billion euro, EUROMAP's around 1,000 machinery manufacturers are market leaders with nearly half of all machines sold being supplied by EUROMAP members.

EUROMAP provides technical recommendations for plastics and rubber machines. In addition to standards for machine descriptions, dimensions and energy measurement, interfaces between machines feature prominently. The provision of manufacturer independent interfaces ensures high levels of machine compatibility.

OPC Foundation

OPC is the interoperability standard for the secure and reliable exchange of data and information in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard.

OPC UA is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework. This multi-layered approach accomplishes the original design specification goals of:

- Platform independence: from an embedded microcontroller to cloud-based infrastructure
- Secure: encryption, authentication, authorization and auditing
- Extensible: ability to add new features including transports without affecting existing applications
- Comprehensive information modelling capabilities: for defining any model from simple to complex

1 Scope

OPC 40079 describes the interface between injection moulding machines (IMM) and robots for data exchange. The target of OPC 40079 is to provide a unique interface for IMM and robots from different manufacturers to ensure compatibility.

The first version of this specification is intended to enable a rather simple implementation with a fixed set of equipment supported. This leads to a fixed number of instances of child elements in several type definitions as described in the "OPC 40079 IMM Fixed DataSet Server Facet" and "OPC 40079 Robot Fixed DataSet Server Facet". The next versions are going to add further *Facets* allowing unlimited equipment and other topics for higher level of integration on production floor.

The first version covers these functionalities:

- Machine is OPC UA client and can optionally be an OPC UA server
- Robot is OPC UA server and can optionally be an OPC UA client
- Realtime exchange of signals via PubSub to prevent mechanical collisions
- Position signals from the IMM: 1 mould, 2 ejectors, 10 cores and 1 additional axis
- Enabling signals from the robot to the IMM to enable/disable movements of the IMM depending on the robot position
- Multiple pub sub connections are possible
- Part tracking
- Signals from the IMM representing the availability of parts in each mould
- Signals from the robot on inserting and removing of parts
- Exchange of production datasets between machine and robot
- Exchange of basic part quality data from IMM to the Robot

Future extensions cover these functionalities:

- Machine and robot are OPC UA server and client as well
- Starting of robot interaction into running machine production cycles
- Detailed part quality for each cavity
- Signals from the IMM on the part quality as determined by the machine
- Signals from the robot on the part quality as determined by the robot (e.g. connected vision systems)
- Dynamic scope of equipment as well as modelling of injection units

Functionalities/Information not intended to be covered in OPC 40079:

- Robot work space definitions for integration in IMM processes.
- Position/speed of robot axes
- Direct control of machine/robot movements. These are part of the machine/robot program.
- Safety Signals: For safety signals like emergency stop the EUROMAP 81 shall be used. As an alternative the relevant signals of an existing EUROMAP 67 interface can be used.

NOTE: Additional features from the robotics CS can be used but are not mandatory for the use of OPC 40079.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments and errata) applies

OPC 10000-1, *OPC Unified Architecture - Part 1: Overview and Concepts*

http://www.opcfoundation.org/UA/Part1/

OPC 10000-3, *OPC Unified Architecture - Part 3: Address Space Model*

http://www.opcfoundation.org/UA/Part3/

OPC 10000-4, *OPC Unified Architecture - Part 4: Services*

http://www.opcfoundation.org/UA/Part4/

OPC 10000-5, *OPC Unified Architecture - Part 5: Information Model*

http://www.opcfoundation.org/UA/Part5/

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*

http://www.opcfoundation.org/UA/Part6/

OPC 10000-7, *OPC Unified Architecture - Part 7: Profiles*

http://www.opcfoundation.org/UA/Part7/

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*

http://www.opcfoundation.org/UA/Part8/

OPC 10000-14, *OPC Unified Architecture - Part 14: PubSub*

http://www.opcfoundation.org/UA/Part14/

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*

http://www.opcfoundation.org/UA/Part100/

OPC 40083, *OPC UA for Plastics and Rubber Machinery – General Type Definitions*

http://www.opcfoundation.org/UA/PlasticsRubber/GeneralTypes

OPC 40001-1 – *OPC UA for Machinery – Part 1: Basic Building Blocks*

http://www.opcfoundation.org/UA/Machinery

3 Terms, definitions and conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling are understood in this specification. This specification will use these concepts to describe the OPC 40079 Information Model. For the purposes of this document, the terms and definitions given in the documents referenced in Clause 2 apply.

Note that OPC UA terms and terms defined in this specification are *italicized* in the specification.

3.2 Conventions used in this document

The conventions described in OPC 40083 apply.

3.3 Abbreviations

IMM	Injection Moulding Machine

4 General information to OPC UA interfaces for plastics and rubber machinery and OPC UA

For general information on OPC UA interfaces for plastics and rubber machinery and OPC UA see OPC 40083.

5 Use cases

OPC 40079 covers the following functionalities:

- Realtime exchange of signals to prevent mechanical collisions
- Part tracking
- Exchange of production datasets between machine and robot
- Exchange of basic part quality data from IMM to the Robot

6 OPC 40079 Information Model overview

For the communication between IMM and robot both are publishers and subscribers for the exchange of real-time data. In addition, the robot shall provide (non-real-time) information in an OPC UA server with an address space (see 0). For the first version with fixed structure of the position and enabling signals it is not mandatory that the IMM provides an OPC UA server with an address space. However, the information contained in the *IMMToRobotType* (see 7) is the source for published *DataSets*.

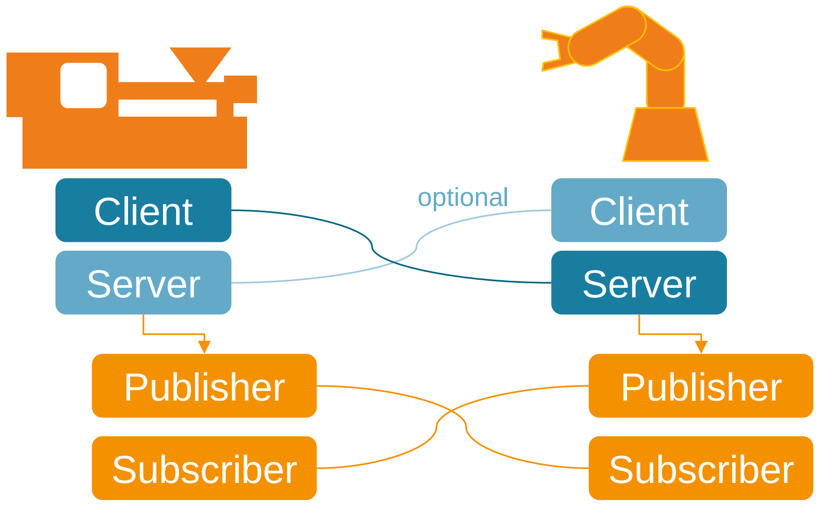

Figure 1 – Combination of Client-Server and PubSub communication

The IMM publishes a fixed set of equipment (mould, ejector, cores, additional axis) containing the position signals in a publisher dataset. The robot subscribes to this dataset from the IMM and publishes a fixed set of enable signals. The IMM again subscribes to this dataset from the robot to receive the enable signals for the same set of equipment. In this manner the IMM and robot establish a bidirectional 1:1 pub-sub-connection.

It is intended that one published dataset from the IMM contains all equipment for one interacting robot.

Nevertheless, two robots can connect to the same IMM when the IMM publishes two datasets with the equal or differing equipment to be controlled by each one of the robots.

It is free to the IMM to preselect equipment into the publishing dataset controllable by the robot and to skip equipment not to be controlled from the robot.

NOTE: All equipment not used from the machine and all enable signals not used from the robot shall be have the value binary value 0 that can be either “InPosition1”, “0.0[mm]”, “NOT_MOVING_0”.

Some exemplary usages:

- One IMM with one mould and one robot → only one dataset needed
- One IMM with two moulds and one robot → IMM publishes two separate datasets, one for each mould; robot publishes two datasets
- Two IMMs with one mould each and one robot → each IMM publishes one dataset for its mould, robot publishes two datasets
- One IMM with two moulds and two robots → IMM publishes two datasets for its moulds; each robot publishes one dataset
- One IMM with one mould and two robots → IMM publishes one dataset for its mould; each robot publishes one datasets → IMM needs to combine the enable signals

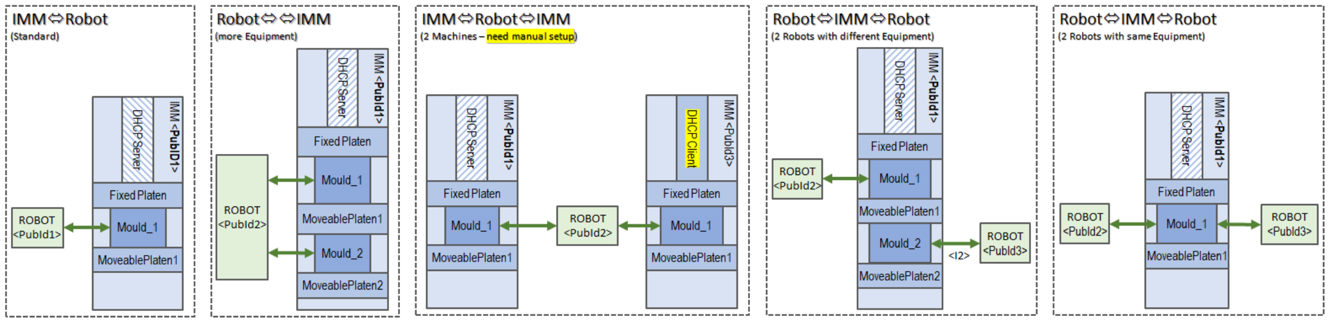

Figure 2 – Use cases with one or two interface instances

A handshake mechanism between the Robot application level and the IMM application level is provided by a *RobotMessageID* send from the robot (see 8.4) and *RobotMessageID_confimed* send from the IMM (see 7.2). When the robot receives in *RobotMessageID_confimed* the value sent from *RobotMessageID* back this means a confirmation that all signals within the same publishing dataset have been considered by the IMM.

The IMM shall control all equipment movements according to the received enable signals from the robot and copy back as *RobotMessageID_confirmed* only when having considered all enable signal changes of the robot and has commanded the IMM axes as defined by the robot. The robot has to get back the requested value before entering any critical regions. This mechanism ensures that no contradicting movements are initiated from the machine after this confirmation. However, it can take some time until already existing movements are stopped. This is why the robot needs to check the position/movement signals from IMM before interaction.

Due to this handshake, there is no need to wait until signal communication and message processing times expire although a robot could use a Watchdog for maximum timeout of the *RobotMessageID_confirmed*. This mechanism enables a reliable pub-sub-communication with different signal runtimes or cycle times and would even work on slow delayed communication channels.

7 IMMToRobotType

7.1 Overview

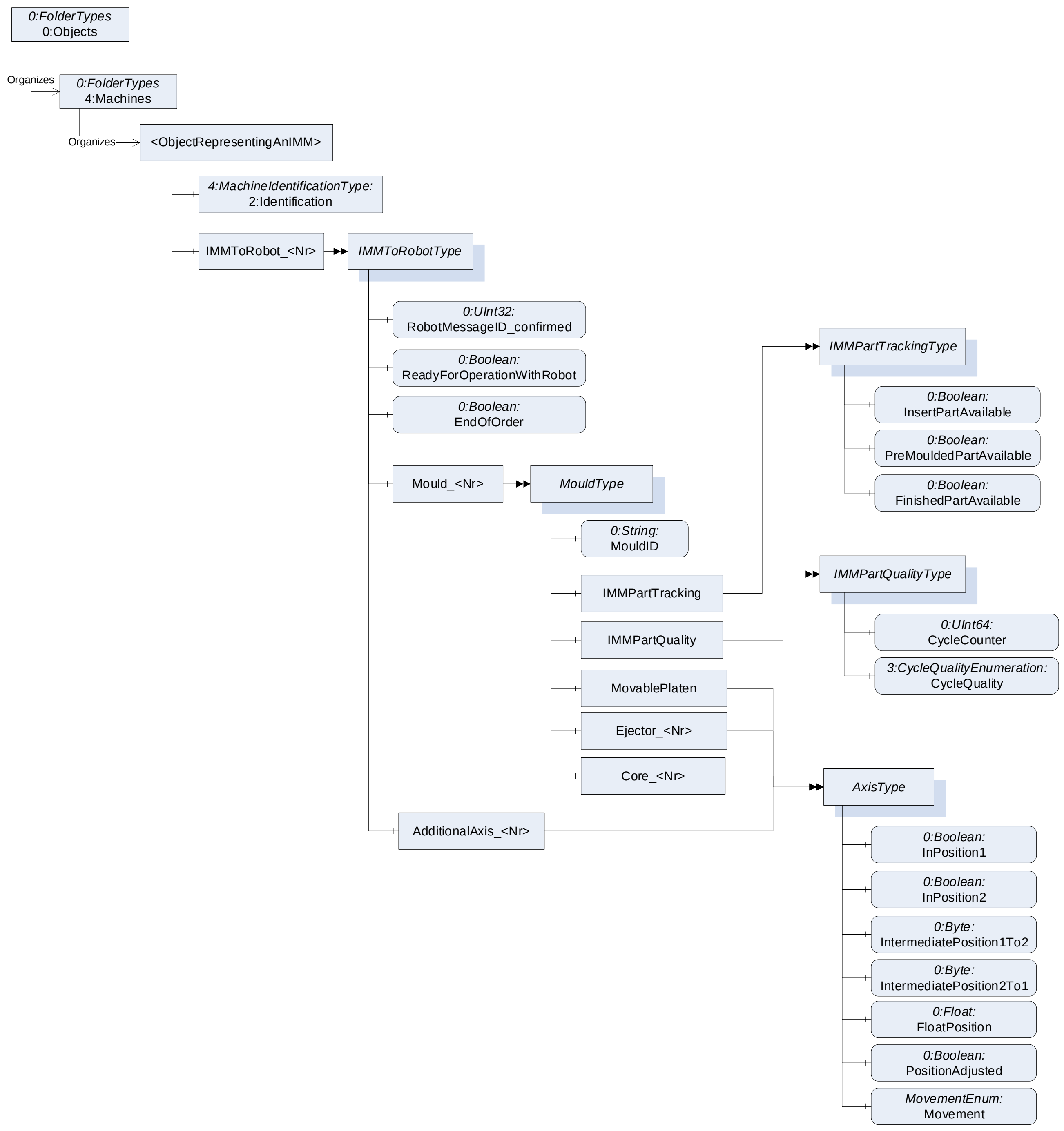

Figure 3 – Overview IMMToRobotType

The *IMMToRobot_Type* represents the interface of the injection moulding machine to the robot and contains all its subcomponents relevant for the interaction with one robot.

The IMM can provide an OPC UA server with an address space (see 6). If so, the instance(s) of *IMMToRobotType* shall be located under an *Object*, which represents the IMM and which is directly located under the *Machines Object* of the Server and has an *Object Identification* as defined in OPC 40001-1.

The BrowseName of the instances shall be “IMMToRobot_1“, “IMMToRobot_2“…

IMM uses the *StartPubSub Method* from 8.2 to tell the robot to subscribe to the dataset using the PublisherId of the IMM and to start publishing its own dataset with the enable signals.

The *IMMToRobotType* is formally defined in Table 1.

Table 1 – IMMToRobotType Definiton

Attribute	**Value**				
BrowseName	IMMToRobotType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Variable	RobotMessageID_confirmed	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ReadyForOperationWithRobot	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	EndOfOrder	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Object	Mould_<Nr>		MouldType	OP
0:HasComponent	Object	AdditionalAxis_<Nr>		AxisType	OP

The modelling rule optional placeholder is used for the mould and additional axis (e.g. turn-table, shuttle-table) for future use of multiple instances.

For the first version with the “OPC 40079 IMM Fixed DataSet Server Facet”, the instances are fixed: One mould *Mould_1* and one additional axis *AdditionalAxis_1*.

7.2 RobotMessageID_confirmed

With this *Variable* the IMM confirms, that it has received and considered the enable signal changes coming from the robot with *RobotMessageID* (see 8.4) and has commanded its axes as defined by the robot.

7.3 ReadyForOperationWithRobot

This *Variable* indicates that the IMM is able to be operated with the robot. This signal shall not be used to start the robot. If the signal turns to *false* during the operation mode of the robot "operation with injection moulding machine", it is recommended that the robot continues its automatic cycle until the end position.

7.4 EndOfOrder

With this signal the IMM indicates that the order is complete. Example: The robot must remove the finished part but not pick a new insert.

7.5 MouldType

The *MouldType* includes information about the mould, its position and movement and includes part tracking and quality information. It is also modelled as container object for providing information about the included axis (ejectors, cores). It is formally defined in Table 2.

Table 2 – MouldType Definiton

Attribute	**Value**				
BrowseName	MouldType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	0:NodeVersion	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	MouldID	0:String	0:PropertyType	O, RO
0:HasComponent	Object	MovablePlaten		AxisType	O
0:HasComponent	Object	Ejector_<Nr>		AxisType	OP
0:HasComponent	Object	Core_<Nr>		AxisType	OP
0:HasComponent	Object	IMMPartTracking		IMMPartTrackingType	O
0:HasComponent	Object	IMMPartQuality		IMMPartQualityType	O
0:GeneratesEvent	ObjectType	0:GeneralModelChangeEventType			

For the first version with the “OPC 40079 IMM Fixed DataSet Server Facet”, *MovablePlaten*, *IMMPartTracking* and *IMMPartQuality* shall be used.

The modelling rule optional placeholder is used for the ejectors and cores for future use of multiple instances. For the first version with the “OPC 40079 IMM Fixed DataSet Server Facet”, the instances are fixed to

– two ejectors Ejector_1 and Ejector_2

– ten cores Core_1 … Core_10

to achieve constant instances of the *MouldType*.

7.5.1 MouldId

This Property is used as identifier of the mould. It can be used for reports.

7.5.2 MovablePlaten

This *Object* describes the movement and position of the movable platen of the mould. The *AxisType* is defined in 7.5.3.

7.5.3 AxisType

The *AxisType* is used to describe the movement and position of one axis.

It is formally defined in Table 3.

Table 3 – AxisType Definiton

Attribute	**Value**				
BrowseName	AxisType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Variable	InPosition1	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	InPosition2	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	IntermediatePosition1To2	0:Byte	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	IntermediatePosition2To1	0:Byte	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	FloatPosition	0:Float	0:BaseDataVariableType	O, RO
0:HasProperty	Variable	PositionAdjusted	0:Boolean	0:PropertyType	O, RO
0:HasComponent	Variable	Movement	MovementEnum	0:BaseDataVariableType	O, RO

For the first version with the “OPC 40079 IMM Fixed DataSet Server Facet”, the following optional variables shall be used to achieve fixed models:

– For movable platen, ejectors and additional axis: all optional variables shall be used

– For cores: *InPosition1*, *InPosition2*, *IntermediatePosition1To2*, *IntermediatePosition2To1* and *Movement* shall be used. *FloatPosition* and *PositionAdjusted* shall not be used.

7.5.3.1 InPosition1, InPosition2

InPosition1 and *InPosition2* are mandatory of type *Boolean* giving the discrete pre-set final positions of the axis in both movement directions. If the axis is not in a final (=end-, =limit-) position, the signal is FALSE. Although, the positions are dependent on the manufacturer/programming, the following is agreed:

Table 4 – Defined positions for mould, core, ejector and turn-table as additional axis

	InPosition1	**InPosition2**
Linear Axes		
Mould (Movable Platen)	Closed	Full open
Ejector	Finally backward (e.g. after the number of its set cycles)	Forward / ejection pos. (e.g. after the number of its set cycles)
Core	Core pullers in moulding position	Core pullers in position to remove moulding
Rotational axes		
Turn-table	*Not used → always false*	Turn-table in working position → if it is important to know, in which position the turn-table is, the *IntermediatePosition1To2* and/or the *FloatPosition* shall be used

7.5.3.2 IntermediatePosition1To2

The IMM can define intermediate position thresholds individually for each movement direction and the equipment is stopped at this position. These positions between the limits *InPosition1* and *InPosition2* are called intermediate positions and are ordered from 1, 2, …, 255.

The values of *IntermediatePosition1To2* are defined for the moving direction starting with *InPosition1=TRUE* to *InPosition2=TRUE*.

A new value in *IntermediatePosition1To2* is set by the IMM, when the actual position is equal or greater than the corresponding threshold position. To have clearance to move to an intermediate position, the IMM expects an according enable signal from the robot side. For the movement to intermediate position 1 the IMM presupposes *EnableIntermediatePosition1To2≥1* and will set *IntermediatePosition1To2=1* after reaching the threshold for intermediate position 1.

If intermediate positions are passed in a very short time (fast movements) it could be possible that *IntermediatePosition1To2* could skip some values. Thus, the robot cannot expect to always receive the values in sequence.

Example for moulds:

When the mould is closed *InPosition1* variable is TRUE, *IntermediatePosition1To2* is 0 and the *FloatPosition* is approximately 0.0 [mm]. Opening the mould results in an increasing *FloatPosition* and *IntermediatePosition1To2* gets the value 1 when the threshold for the first intermediate position is reached. After reaching the threshold for the second intermediate position the *IntermediatePosition1To2* turns from the value 1 to the value 2. After reaching the last threshold for an opening intermediate position the variable *InPosition2* turns true and the *FloatPosition* gets a maximum value. *IntermediatePosition1To2* stays in the last value.

Example for no Intermediate position:

The *IntermediatePosition1To2* should remain 0 all over.

Example mould with 3 opening intermediate positions and 2 closing intermediate positions:

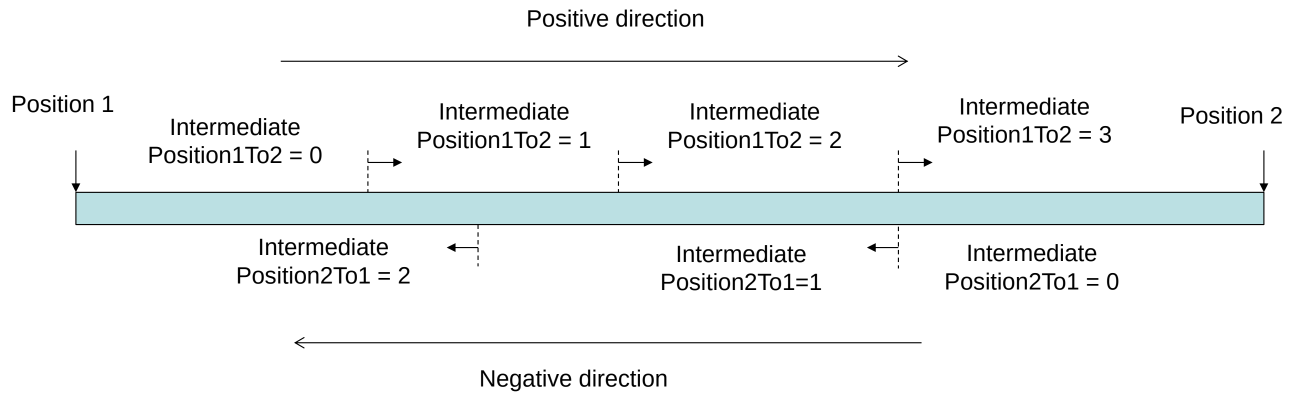
Positive direction
Position 1
Intermediate Position1To2 = 0
Intermediate Position1To2 = 1
Intermediate Position1To2 = 2
Intermediate Position1To2 = 3
Position 2
Intermediate Position2To1 = 2
Intermediate Position2To1=1
Intermediate Position2To1 = 0
Negative direction

Figure 4 – Definition of intermediate positions signals

The value of *IntermediatePosition1To2* shall always be valid, even when the current movement is in the other direction.

7.5.3.3 IntermediatePosition2To1

The intermediate positions for the direction *InPosition2* towards *InPosition1* are ordered from 1, 2, …, 255.

The values of *IntermediatePosition2To1* are defined for the moving direction starting with *InPosition2=TRUE* to *InPosition1=TRUE*.

A new value in *IntermediatePosition2To1* is set by the IMM, when the actual position is equal or less than the corresponding threshold position. To have clearance to move to an intermediate position, the IMM expects an according enable signal to from the robot side. For the movement to intermediate position 2 the IMM presupposes *EnableIntermediatePosition2To1≥2* and will set *IntermediatePosition2To1=2* after reaching the threshold for intermediate position 2.

If intermediate positions are passed in a very short time (fast movements) it could be possible that *IntermediatePosition2To1* could skip some values. Thus, the robot cannot expect to always receive the values in sequence.

7.5.3.4 FloatPosition

The *FloatPosition Variable* allows to give the current position as float value using fixed millimetre unit for linear axes and degrees for rotational axes.

7.5.3.5 PositionAdjusted

The *PositionAdjusted Property* indicates that the position of the axis is correct (homing has been done) and a position of 0 mm corresponds to a fully closed mould, ejector finally backward, core in moulding position (for cores not used in first step). During bootup or while changing a mould *PositionAdjusted* = FALSE shows the robot not to rely on the *FloatPosition* values.

7.5.3.6 Movement

The *Movement Variable* indicates whether the axis is moving or not. The enumeration *MovementEnum* is defined in Table 5.

Table 5 – MovementEnum Items

Name	Value	Description
NOT_MOVING	0	The axis is at rest.
MOVING_IN_POSITIVE_DIRECTION	1	The axis is moving from position 1 to 2
MOVING_IN_NEGATIVE_DIRECTION	2	The axis is moving from position 2 to 1

Its representation in the *AddressSpace* is defined in Table 6.

Table 6 – MovementEnum Definiton

Attribute	Value				
BrowseName	MovementEnum				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*Enumeration* type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

NOTE: For rotational axis (e.g. turn tables with position 1, 2 and 3) the movement from the highest position (3) to the lowest (1) is also a positive direction.

7.5.4 IMMPartTrackingType

The *IMMPartTrackingType* signals the availability of insert parts, pre-moulded parts and finished parts in a mould. It is formally defined in Table 7.

It is expected that the IMM sets the *FinishedPartAvailable* as well as the *PreMouldedPartsAvailable* to TRUE before the mould movements releasing the parts are started (e.g. mould opening). The *InsertPartAvailable* should be set FALSE at the same timepoint.

Note: Providing *InsertPartAvailable* by IMM could for example be relevant during start-up so that the robot knows that insert parts are already inserted and the robot does not try to insert another one.

The status variables are set back to FALSE if they are triggered by the appropriate robot removal signals (see *RobotPartTracking* in 8.6.4) or removed from the IMM (e.g. dropped by ejectors or cores).

Table 7 – IMMPartTrackingType Definiton

Attribute	Value				
BrowseName	IMMPartTrackingType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Variable	InsertPartAvailable	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PreMouldedPartAvailable	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	FinishedPartAvailable	0:Boolean	0:BaseDataVariableType	M, RO

Note: If the process has several steps creating pre-moulded parts (e.g. multi-colour machines), *PreMouldedPartAvailable* becomes TRUE after the first step. *FinishedPartAvailable* becomes TRUE after the last step.

During automatic production with robot, *FinishedPartAvaible* stays TRUE until the robot has sent *FinishedPartRemoved*. (During start-up the IMM sets *FinishedPartAvaible* back to FALSE when the part fell down.)

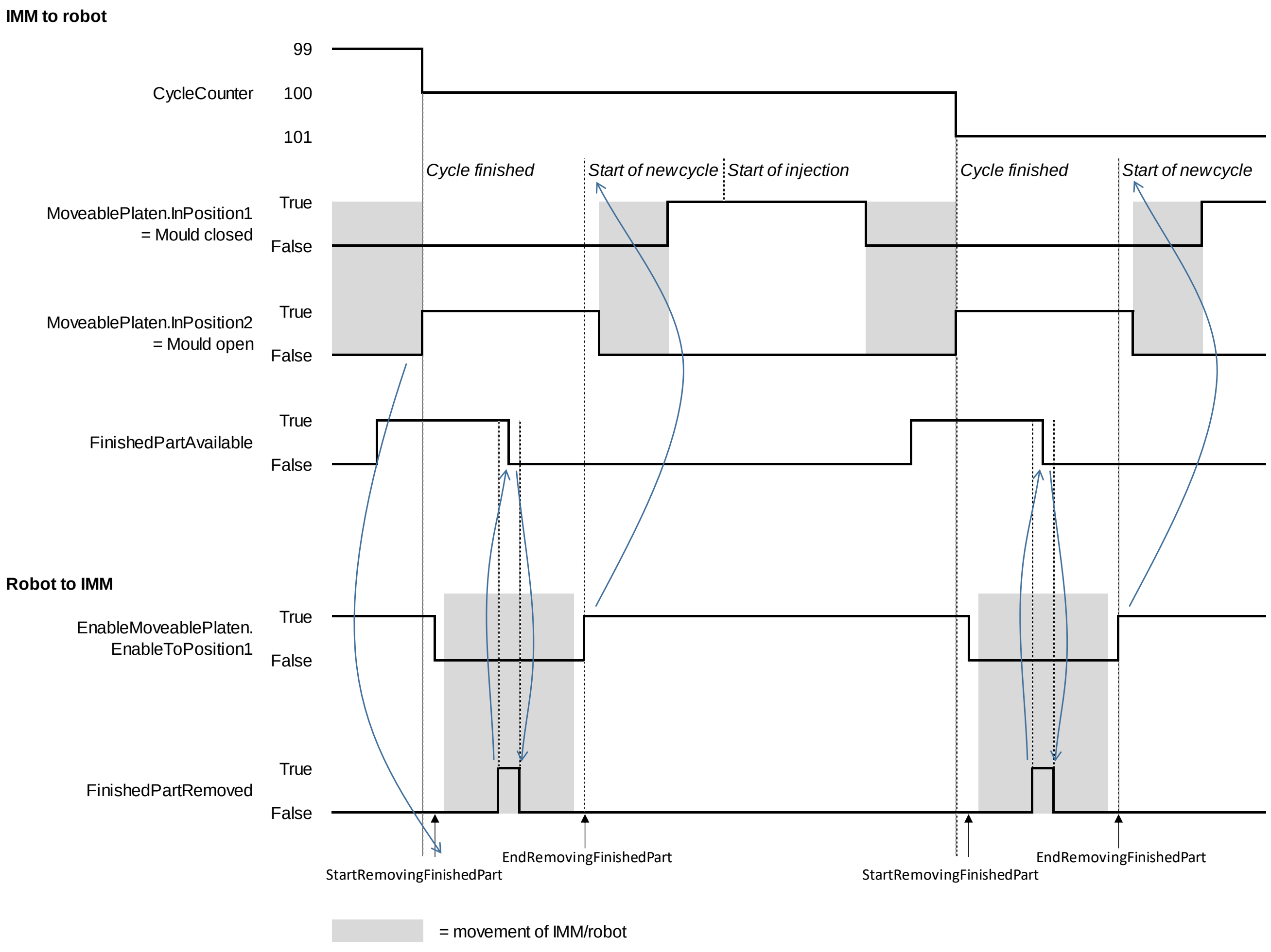

Figure 5 – Timing for signals for part tracking

7.5.5 IMMPartQualityType

The *IMMPartQualityType* provides summarized information about the quality of produced parts and is formally defined in Table 8.

Table 8 – IMMPartQualityType Definiton

Attribute	**Value**				
BrowseName	IMMPartQualityType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Variable	CycleCounter	0:UInt64	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	CycleQuality	3:CycleQualityEnumeration	0:BaseDataVariableType	M, RO

The IMM can provide detailed information about the cycle including quality information for each cavity by calling the *Method IMMToRobotCycleInformation* (see 8.6.5.3)

7.5.5.1 CycleCounter

Number of completely finished cycles in the current production. When the machine increases the *CycleCounter* the machine should call the *IMMToRobotCycleInformation* Method from the robot with the cavity quality details (see 8.6.5.3).

7.5.5.2 CycleQuality

Summarised information on the quality of the whole cycle as determined by the IMM. The *CycleQualityEnumeration* is defined in OPC 40083.

8 RobotToIMMType

8.1 Overview

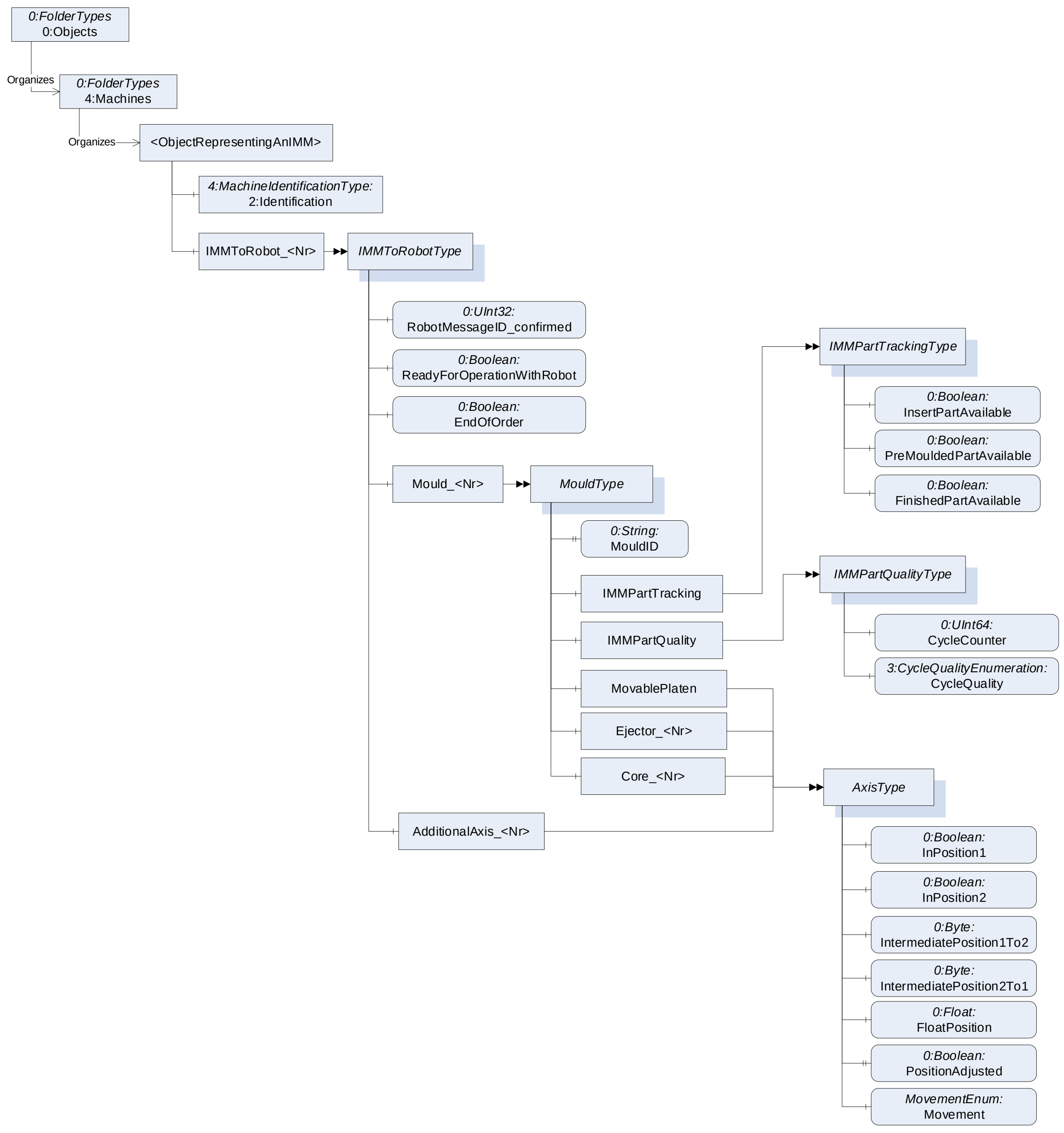

Figure 6 – Overview RobotToIMMType

This OPC UA *ObjectType* is used for the root *Object* representing a robot to IMM connection with all its subcomponents relevant for the interaction between this robot and IMM.

The instance(s) of *RobotToIMM_InterfaceType* shall be located under an *Object*, which represents the robot and which is directly located under the *Machines Object* of the Server and has an *Object Identification* as defined in OPC 40001-1.

The robot provides one instance for each interaction between one mould of an IMM and the robot. The *BrowseName* shall be “RobotToIMM_ 1”, “RobotToIMM_ 2”, … . It is up to the manufacturer/integrator to provide a server with a flexible or fixed number of instances. In the first case, new instances can be created via the control system of the robot. In the second case, the number of instances corresponds with the maximal possible connections. In unused instances, the *Method StartPubSub* will not be called so that so unnecessary data is put into the real-time data exchange.

The *RobotToIMMType* is formally defined in Table 9.

Table 9 – RobotToIMMType Definiton

Attribute	**Value**				
BrowseName	RobotToIMMType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Object	2:Identification		4:MachineIdentificationType	M
0:HasComponent	Method	StartPubSub			M
0:HasComponent	Method	StopPubSub			M
0:HasComponent	Variable	RobotMessageID	0:UInt32	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	ReadyForOperationWithIMM	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Object	MouldInteraction_<Nr>		MouldInteractionType	OP
0:HasComponent	Object	EnableAdditionalAxis_<Nr>		EnableType	OP
0:HasComponent	Object	ProductionDatasetManagement		3:ProductionDatasetManagementType	O
0:HasComponent	Object	UserLogin		UserLoginType	O

In the first version with the “OPC 40079 Robot Fixed DataSet Server Facet”, part exactly one mould *MouldInteraction_1* and one additional axis *EnableAdditionalAxis_1* shall be used.

8.2 StartPubSub

The *Method StartPubSub* initiates the PubSub connection between 1 IMM and 1 robot interface.

The input parameters contain information about the configuration of the publisher in the IMM for the IMMProcessValues. The robot uses this for configuration of the subscription.

The output parameters contain information about the configuration of the publisher in the robot for the RobotSignals. The IMM uses this for configuration of the subscription.

The signature of this *Method* is specified below. Table 10 and Table 11 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
StartPubSub (
   [in]    0:String        IMMTransportProfileUri,
   [in]    0:String        IMMAddress,
   [in]    0:UInt64        IMMPublisherId,
   [in]    0:UInt16        IMMWriterGroupId,
   [in]    0:UInt16        IMMDataSetWriterId,
   [in]    0:Byte          IMMProtocolMajorVersion,
   [in]    0:Byte          IMMProtocolMinorVersion,
   [out]   0:String        RobotTransportProfileUri,
   [out]   0:String        RobotAddress,
   [out]   0:UInt64        RobotPublisherId,
   [out]   0:UInt16        RobotWriterGroupId,
   [out]   0:UInt16        RobotDataSetWriterID,
   [out]   0:Byte          RobotProtocolMajorVersion,
   [out]   0:Byte          RobotProtocolMinorVersion);
```

Table 10 – StartPubSub Method Arguments

Argument	Description
IMMTransportProfileUri	Transport Profile used by IMM. The possible values are defined as URI of the transport protocols defined as PubSub transport Facet in OPC 10000-7.
IMMAddress	Network address of IMM publisher
IMMPublisherId	PublisherId of the IMM
IMMWriterGroupId	WriterGroupId of the IMM's writer group which contains the OPC 40079 DataSets
IMMDataSetWriterId	DataSetWriterId of IMM process values for the called interface
IMMProtocolMajorVersion	Supported major version of the protocol by IMM → this defines the structure of the published dataset
IMMProtocolMinorVersion	Supported minor version of the protocol by IMM → this defines the structure of the published dataset
RobotTransportProfileUri	Transport Profile used by the robot. The possible values are defined as URI of the transport protocols defined as PubSub transport Facet in OPC 10000-7.
RobotAddress	Network address of robot publisher
RobotPublisherId	PublisherId of the robot
RobotWriterGroupId	WriterGroupId of the robot's writer group which contains the OPC 40079 DataSets
RobotDataSetWriterID	DataSetWriterId of robot signals for the called interface
RobotProtocolMajorVersion	Supported major version of the protocol by robot → this defines the structure of the published dataset and informs the IMM if the received datasets are interpreted in the correct way
RobotProtocolMinorVersion	Supported minor version of the protocol by robot → this defines the structure of the published dataset and informs the IMM if the received datasets are interpreted in the correct way

For this version of OPC 40079 ("1.0")

– *IMMProtocolMajorVersion* and *RobotProtocolMajorVersion* shall be set to 1,

– *IMMProtocolMinorVersion* and *RobotProtocolMinorVersion* shall be set to 0.

– *IMMTransportProfileUri* and *RobotTransportProfileUri* shall be set to http://opcfoundation.org/UA-Profile/Transport/pubsub-udp-uadp ("PubSub UDP UADP" Profile)

Table 11 – StartPubSub Method AddressSpace Definiton

Attribute	Value				
BrowseName	StartPubSub				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory
0:HasProperty	Variable	0:OutputArguments	0:Argument[]	0:PropertyType	0:Mandatory

The *Method* should return the status code BadMaxConnectionsReached,0x80B70000 if a second IMM tries to start PubSub when there is already a first IMM connected.

The following examples show the use of the *Method*. An established session between IMM-Client and Robot-Server is assumed. The example also assumes that each participant already has a unique *PublisherId*.

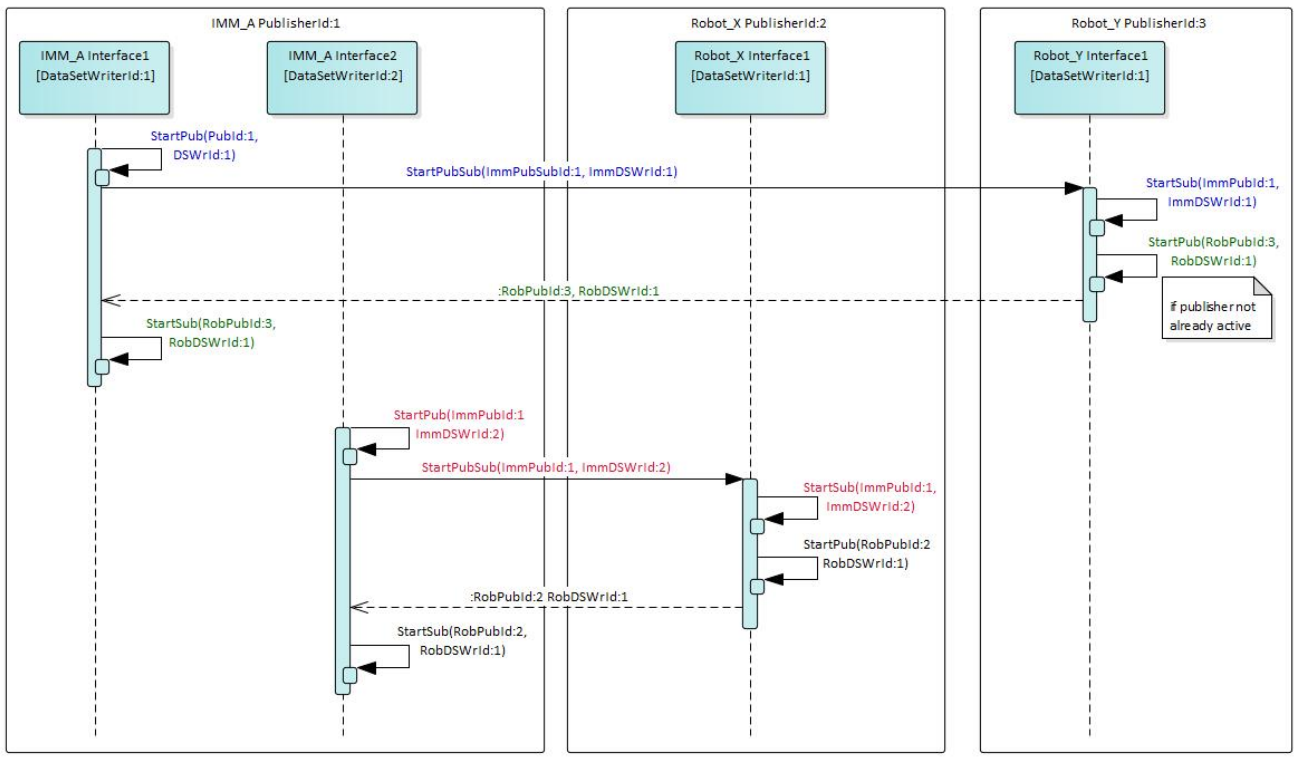
IMM_A PublisherId:1
Robot_X PublisherId:2
Robot_Y PublisherId:3
IMM_A Interface1 [DataSetWriterId:1]
IMM_A Interface2 [DataSetWriterId:2]
Robot_X Interface1 [DataSetWriterId:1]
Robot_Y Interface1 [DataSetWriterId:1]
StartPub(PubId:1, DSWrId:1)
StartPubSub(ImmPubSubId:1, ImmDSWrId:1)
StartSub(ImmPubId:1, ImmDSWrId:1)
StartPub(RobPubId:3, RobDSWrId:1)
:RobPubId:3, RobDSWrId:1
if publisher not already active
StartSub(RobPubId:3, RobDSWrId:1)
StartPub(ImmPubId:1 ImmDSWrId:2)
StartPubSub(ImmPubId:1, ImmDSWrId:2)
StartSub(ImmPubId:1, ImmDSWrId:2)
StartPub(RobPubId:2 RobDSWrId:1)
:RobPubId:2 RobDSWrId:1
StartSub(RobPubId:2, RobDSWrId:1)

Figure 7 – Example for StartPubSub for 1 IMM with 2 robots

In the sequence shown, the robots use the *StartPubSub Method* as a trigger to start the publisher. As an alternative, the robot can also start its publisher independently earlier.

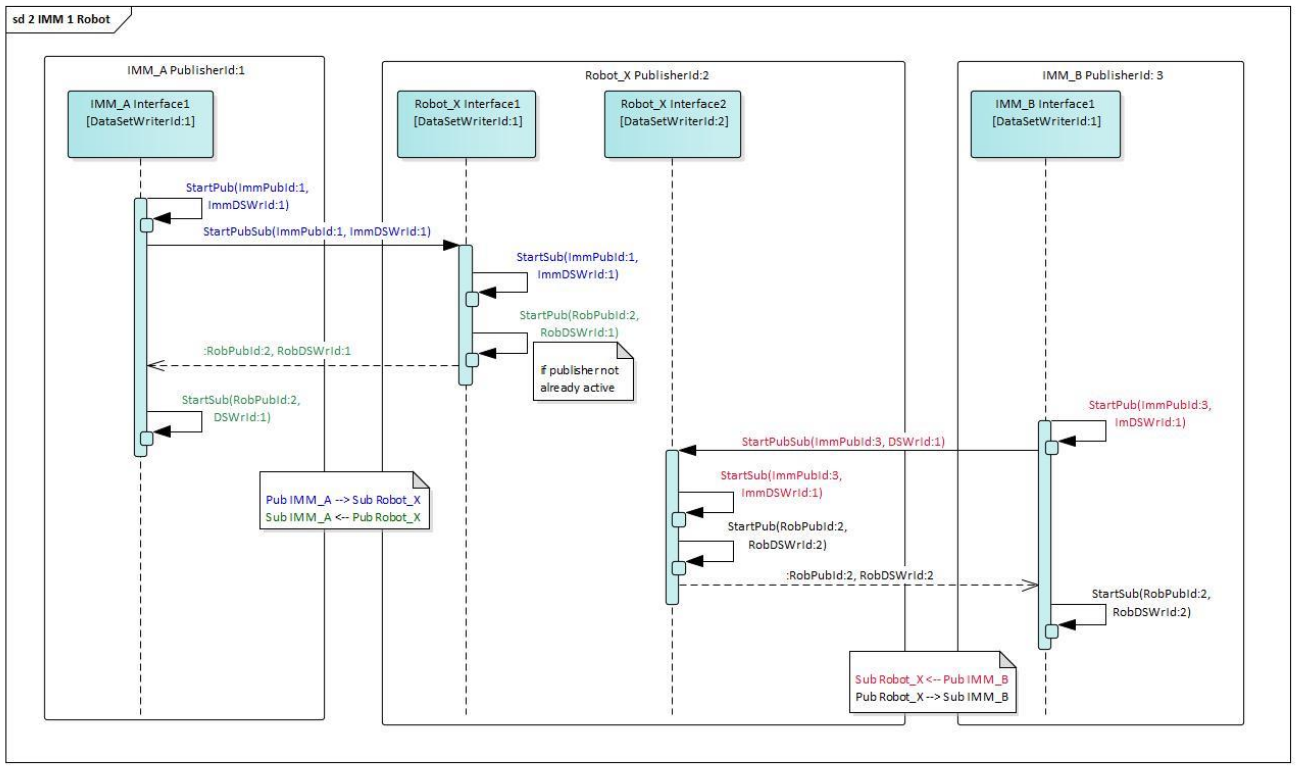
sd 2 IMM 1 Robot
IMM_A PublisherId:1
Robot_X PublisherId:2
IMM_B PublisherId: 3
IMM_A Interface1 [DataSetWriterId:1]
Robot_X Interface1 [DataSetWriterId:1]
Robot_X Interface2 [DataSetWriterId:2]
IMM_B Interface1 [DataSetWriterId:1]
StartPub(ImmPubId:1, ImmDSWrId:1)
StartPubSub(ImmPubId:1, ImmDSWrId:1)
StartSub(ImmPubId:1, ImmDSWrId:1)
StartPub(RobPubId:2, RobDSWrId:1)
if publisher not already active
:RobPubId:2, RobDSWrId:1
StartSub(RobPubId:2, DSWrId:1)
Pub IMM_A --> Sub Robot_X
Sub IMM_A <-- Pub Robot_X
StartPub(ImmPubId:3, ImDSWrId:1)
StartPubSub(ImmPubId:3, DSWrId:1)
StartSub(ImmPubId:3, ImmDSWrId:1)
StartPub(RobPubId:2, RobDSWrId:2)
:RobPubId:2, RobDSWrId:2
StartSub(RobPubId:2, RobDSWrId:2)
Sub Robot_X <-- Pub IMM_B
Pub Robot_X --> Sub IMM_B

Figure 8 – Example for StartPubSub for 1 robot with 2 IMM

In the sequence shown, the robot uses the *StartPubSub Method* as a trigger to start the publisher.

As an alternative, the robot can also start its publisher independently earlier.

The IMM should request user acknowledge whenever the *RobotPublisherId* has changed since the last time the *Method* was called. This could mean that the entire Robot or the behaviour of the Robot has changed. The same applies when the *RobotDataSetWriterId* has changed – it could mean that a Robot with multiple instances of the interface has changed the instance(interface).

The Robot should request user acknowledge whenever the *IMMPublisherId* has changed since the last time the *Method* was called. This could mean that the entire IMM or the behaviour of the IMM has changed. The same applies when the *IMMDataSetWriterId* has changed – it could mean that an IMM with multiple instances of the interface has changed the instance(interface).

8.3 StopPubSub

The *Method StopPubSub* is used to terminate the PubSub connection between 1 IMM and 1 robot interface without creating alarms. This *Method* should only be called when IMM and robot are in a safe position. It is recommended, that after the call of the *Method* there are no movements of IMM or robot unless confirmed by an operator.

Signature

```
StopPubSub (
   [in]    0:UInt64         IMMPublisherId,
   [in]    0:UInt16         IMMDataSetWriterId,
   [in]    0:UInt64         RobotPublisherId,
   [in]    0:UInt16         RobotDataSetWriterId);
```

Table 12 – StopPubSub Method Arguments

Argument	Description
IMMPublisherId	*PublisherId* of the IMM used by the dataset that will not be published any longer
IMMDataSetWriterId	*DataSetWriterId* of IMM process values dataset that will not be published any longer
RobotPublisherId	*PublisherId* of the robot used by the dataset that shall stop publishing
RobotDataSetWriterId	*DataSetWriterId* of robot signals dataset that shall stop publishing

Table 13 – StopPubSub Method AddressSpace Definiton

Attribute	Value				
BrowseName	StopPubSub				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory

The IMM shall keep publishing the position signals and subscribing (and considering) the enable signals until the robot has confirmed the received *Method* call by sending the response with (a good or bad) *StatusCode*.

Note: As the *DataSetWriteId* is unique within a *PublisherId*, it is not necessary to include all parameters used in *StartPubSub* also in *StopPubSub* to identify, which publisher is stopped.

8.4 RobotMessageID

The robot uses the *RobotMessageID* variable as a marker for the enable signals requested published for the IMM. As soon as the IMM returns the value back in the variable *RobotMessageID_confirmed* the robot can assume that the enable conditions are considered from the IMM. This means it is safe for the robot to enter the IMM if movements have been locked and the *RobotMessageID* was confirmed.

The *RobotMessageID* can be handled in several ways:

– Change the *RobotMessageID* in every published dataset

– Change the *RobotMessageID* every time any content of the published dataset is changed

– Change the *RobotMessageID* every time machine movements are disabled in order to move interfering areas

– Never change the RobotFrameID and wait on pub sub timeouts before entering IMM areas → not recommended

The *RobotMessageID* and *RobotMessageID_confirmed* should be handled in the robot and IMM application controlling the movements. This provides the validation that both applications are still alive and working proper.

8.5 ReadyForOperationWithIMM

The robot uses the *ReadyForOperationWithIMM* variable to indicate that it is able to operate with IMM.If *ReadyForOperationWithIMM = false*, the robot shall still fill all enable signals and the IMM still shall consider them. However, in most cases the robot will set all enable signals to allow all movements.

8.6 MouldInteractionType

The *MouldInteractionType* is used to enable movements and positions of a mould. It includes the objects for enabling movements and positions of the included equipment (ejectors and cores). It is formally defined in Table 14.

Table 14 – MouldInteractionType Definiton

Attribute	**Value**				
BrowseName	MouldInteractionType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Variable	MouldAreaFree	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Object	EnableMovablePlaten		EnableType	O
0:HasComponent	Object	EnableEjector_<Nr>		EnableType	OP
0:HasComponent	Object	EnableCore_<Nr>		EnableType	OP
0:HasComponent	Object	RobotPartTracking		RobotPartTrackingType	O
0:HasComponent	Object	RobotPartQuality		RobotPartQualityType	O

In the first version with the “OPC 40079 Robot Fixed DataSet Server Facet”, the number of instances is fixed:

– two ejectors EnableEjector_1, EnableEjector_2

– ten cores EnableCore_1, …, EnableCore_10

– the optional EnableMovablePlaten

– the optional RobotPartTracking

– the optional RobotPartQuality

shall be used to achieve constant instances of the *MouldInteractionType*.

8.6.1 MouldAreaFree

Note: This signal is kept for compatibility with EUROMAP 67.

The signal is TRUE when the robot is outside the mould area and does not interfere with mould opening and closing movements. The signal shall be FALSE when the robot is inside the mould area – the protected area shall be defined in the robot systems. However, the IMM always has to consider the enable signals coming from the robot as defined in 8.4.3 if *RelevantForInteraction* is true.

For mould closing, the signal is combined with the signals included in the *EnableType* (see 8.6.3) used for the movable platen in the way described in table Table 15:

Table 15 – Combination of MouldAreaFree and Enable Signals for mould closing

MouldAreaFree	**EnableToPosition1**	**EnableIntermediate Position2To1**	**Result**
TRUE	TRUE	any	closing movement allowed
TRUE	FALSE	Not used or 0	no closing movement allowed
TRUE	FALSE	x > 0	movement until given intermediate position x allowed
FALSE	TRUE or FALSE	Not used or 0	no closing movement allowed (*MouldAreaFree* overrides *EnableToPosition1*)
FALSE	TRUE or FALSE	x > 0	movement until given intermediate position x allowed (*MouldAreaFree* overrides *EnableToPosition1* but is overridden by *EnableIntermediatePosition2To1*)

For mould opening, the allowed movement only depends on the enable signals even if *MouldAreaFree* is FALSE.

8.6.2 EnableMovablePlaten

The *EnableMovablePlaten* releases the movements of the mould. The *EnableType* is defined in 8.6.3.

8.6.3 EnableType

The *EnableType* is used to release movements and positions of one axis and is formally defined in Table 16.

Table 16 – EnableType Definiton

Attribute	**Value**				
BrowseName	EnableType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	RelevantForInteraction	0:Boolean	0:PropertyType	M, RW
0:HasComponent	Variable	EnableToPosition1	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	EnableToPosition2	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	EnableIntermediatePosition1To2	0:Byte	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	EnableIntermediatePosition2To1	0:Byte	0:BaseDataVariableType	O, RO

In the first version with the “OPC 40079 Robot Fixed DataSet Server Facet” also the optional variables shall be used to achieve constant instances of the *EnableType*.

8.6.3.1 RelevantForInteraction

Indication if the axis is relevant for the IMM-robot-interaction. If true, the robot shall set the enabling signals according to the necessary interaction. If false, all enable signals shall be set to false / 0.

8.6.3.2 EnableToPosition1

The variable *EnableToPosition1* enables the complete movement in the direction towards to *InPosition1* (e.g. mould closing).

Attention: When the signal *EnableToPosition1* is set, all movements towards *InPosition1* are allowed regardless of the content of *EnableIntermediatePositions2to1*.

See Table 4 for a more detailed description of *InPosition1* and *InPosition2.*

8.6.3.3 EnableToPosition2

The variable *EnableToPosition2* enables the complete movement in the direction towards to *InPosition2* (e.g. mould opening).

Attention: When the signal *EnableToPosition2* is set, all movements towards *InPosition2* are allowed regardless of the content of *EnableIntermediatePositions1to2*.

8.6.3.4 EnableIntermediatePosition1To2

The *EnableIntermediatePosition1To2* is used to move axis and stop axis in defined intermediate positions e.g. while mould opening. The *EnableIntermediatePosition1To2* is only considered from the IMM if the *EnableToPosition2* is FALSE at the same time.

Example:

If *EnableIntermediatePosition1To2* is set to 1 and *EnableToPosition2* is set to FALSE the IMM shall stop the movement from *InPosition1* towards *InPosition2* at the intermediate position 1 of this direction ending up with a value of 1 in the variable *IntermediatePosition1To2*.

See 7.5.3.2 for more details of the intermediate signal definition.

8.6.3.5 EnableIntermediatePosition2To1

The *EnableIntermediatePosition2To1* is used to move axis and stop axis in defined intermediate positions e.g. while mould closing. The *EnableIntermediatePosition2To1* is only considered from the IMM if the *EnableToPosition1* is FALSE at the same time.

Example: If *EnableIntermediatePosition2To1* is set to 1 and *EnableToPosition1* is set to FALSE the IMM shall stop the movement from *InPosition2* towards *InPosition1* at the intermediate position 1 of this direction ending up with a value of 1 in the variable *IntermediatePosition2To1.*

8.6.4 RobotPartTrackingType

The *RobotPartTrackingType* contains variables triggering the insertion or removal of inserts, pre-moulded parts or finished parts by the robot.

Whenever the robot sets a inserted or removed variable to TRUE in the *RobotPartTrackingType* it should get the feedback in the appropriate availability status variable of the IMM (see *IMMPartTrackingType* in 7.5.4). After the IMM has adjusted the part status the robot should set the inserted or removed variable back to FALSE.

The raise of *FinishedPartAvailable* is the trigger for the robot that the finished part should be removed.

The raise of *PreMouldedPartAvailable* is the trigger for the robot that the pre-moulded part should be removed or replaced.

In the first version with the “OPC 40079 Robot Fixed DataSet Server Facet” the *RobotPartTracking* instance shall be used to achieve constant instances of the *MouldInteractionType*.

The *RobotPartTrackingType* is formally defined in Table 17.

Table 17 – RobotPartTrackingType Definiton

Attribute	**Value**				
BrowseName	RobotPartTrackingType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	UsedCavities	0:Boolean[]	0:PropertyType	M, RW
0:HasComponent	Variable	InsertPartInserted	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	InsertPartRemoved	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PreMouldedPartInserted	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	PreMouldedPartRemoved	0:Boolean	0:BaseDataVariableType	M, RO
0:HasComponent	Variable	FinishedPartRemoved	0:Boolean	0:BaseDataVariableType	M, RO

UsedCavities: Information which of the existing cavities are used in the current production. This can be set by the IMM to tell the robot where parts shall be inserted and/or removed. For the first version with the “OPC 40079 Robot Fixed DataSet Server Facet”, the array size is fixed to 256.

InsertPartInserted:	The robot has inserted insert part(s) in all used cavities
InsertPartRemoved:	The robot has removed insert part(s) from all used cavities
PreMouldedPartInserted:	The robot has inserted pre-moulded part(s) in all used cavities
PreMouldedPartRemoved:	The robot has removed pre-moulded part(s) from all used cavities
FinishedPartRemoved:	The robot has removed finished part(s) from all used cavities. The signal stays TRUE until the IMM has confirmed it by setting *FinishedPartAvailable* to FALSE.

8.6.5 RobotPartQualityType

The *RobotPartQualityType* provides summary information about the last finished cycle. An event for more detailed information is provided.

The robot should increase the *CycleCounter* variable as soon as the information is updated that is all quality checks are completed.

The machine should call the *IMMToRobotCycleInformation Method* as soon as the machine increases the *CycleCounter* of the machine.

The robot should raise a *RobotToIMMCycleInformationEvent* as soon as the robot increases the *CycleCounter* to receive the detailed quality and identification results from the robot.

The *RobotPartQualityType* is formally defined in Table 18.

Table 18 – RobotPartQualityType Definiton

Attribute	**Value**				
BrowseName	RobotPartQualityType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	CycleCounter	0:UInt64	PropertyType	M, RO
0:HasComponent	Variable	CycleQuality	3:CycleQualityEnumeration	0:BaseDataVariableType	M, RO
0:HasComponent	Method	IMMToRobotCycle Information			O
0:GeneratesEvent	ObjectType	RobotToIMMCycle InformationEventType			

Note: *CycleCounter* and *CycleQuality* are part of the PubSub communicaiton.

8.6.5.1 CycleCounter

Number of finished cycles in the current production. This number refers to the *CycleCounter* provided by the IMM.

8.6.5.2 CycleQuality

– Summarised information on the quality of the whole cycle as determined by the robot. The *CycleQualityEnumeration* is defined in OPC 40083.

– The robot provides detailed information about the cycle including quality information for each cavity, when the IMM calls the *Method RobotToIMMCycleInformation* (see 8.6.5.4)

8.6.5.3 IMMToRobotCycleInformation

When the machine calls the *IMMToRobotCycleInformation Method* detailed part quality and part identification information is transferred to the robot.

Signature

```
IMMToRobotCycleInformation (
   [in]    0:UInt64                               CycleCounter
   [in]    3:CavityCycleQualityEnumeration[]      CavityCycleQuality
   [in]    0:String[]                             PartId
   [in]    0:String                               CurrentLotName);
```

All input variables (except *CycleCounter*) are optional so that they can be NULL or empty.

Table 19 – IMMToRobotCycleInformation Method Arguments

Argument	Description
CycleCounter	Number of the cycle for which detailed quality information is provided
CavityCycleQuality	Information for each cavity of the quality of the cycle if the injection moulding machine is able to determine this. The *CavityCycleQualityEnumeration* is defined in OPC 40083. Array length = number of cavities. → For the first version with the “OPC 40079 Robot Fixed DataSet Server Facet”, the array length is fixed to 256
PartId	The PartId Property is an array with one entry for each cavity that represent the Ids of the parts produced in the cycle. NOTE: The Id(s) may be generated in the machine or coming from outside (e.g. from MES or labelling machine).
CurrentLotName	Name of the current production lot

Table 20 – IMMToRobotCycleInformation Method AddressSpace Definiton

Attribute	Value				
BrowseName	IMMToRobotCycleInformation				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory

8.6.5.4 RobotToIMMCycleInformationEventType

When the robot increases the *CycleCounter* information of a new cycle is available, it shall raise a *RobotToIMMCycleInformationEvent* to provide detailed part quality and/or part identification information determined by the robot.

The machine could aggregate this information with their part quality and identification information for the same cycle and – in case of an existing EM77 connection – provides the total information for the MES.

Table 21 – RobotToIMMCycleInformationEventType Definiton

Attribute	Value				
BrowseName	RobotToIMMCycleInformationEventType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:*BaseEventType* defined in OPC 10000-5					
0:HasProperty	Variable	CycleCounter	0:UInt64	PropertyType	M
0:HasProperty	Variable	CavityCycleQuality	3:CavityCycleQualityEnumeration[]	PropertyType	O
0:HasProperty	Variable	PartId	0:String[]	PropertyType	O
0:HasProperty	Variable	BoxId	0:String	PropertyType	O

CycleCounter:	Number of the cycle for which detailed quality information is given by the robot
CavityCycleQuality:	Information for each cavity of the quality of the cycle if the robot is able to determine this. The *CavityCycleQualityEnumeration* is defined in OPC 40083. Array length = number of cavities. → For the first version with the “OPC 40079 Robot Fixed DataSet Server Facet”, the array length is fixed to 256.

PartId: This Property is an array with one entry for each cavity that represent the Ids of the parts produced in the cycle.

NOTE: The Id(s) may be generated in the machine or coming from outside (e.g. from MES or labelling machine).

BoxId: Id of the current box, where the produced parts have been put in.

8.7 ProductionDatasetManagement

The “OPC 40083 OPC UA for Plastics and Rubber Machinery – General Type Definitions” contains the type definitions for the production dataset management. The robot can provide an instance of the *ProductionDatasetManagementType* in the root object.

The IMM is supposed to be the central manager of the production datasets of the entire production cell. These production data sets of several components of the production cell are thought to be BLOB files (binary large objects) and shall never be changed outside the provider of the data.

This means the IMM wraps all production datasets including the own one into one entire BLOB ready to be transferred t over OPCUA 40077 to a MES or onto an external file system e.g. USB data storage. It there is a MES system for a production cell the IMM is intended to be the gateway and the manager of production datasets and job information.

When the IMM gets a production request together with the original BLOB the IMM unwraps the received BLOB into the several production datasets (which are BLOBs again) and transfers them unchanged back to the various equipment of the production cell (e.g. robot, …). The *ProductionDatasetManagement* Object from OPC 40083 is used for managing the transfer of these production datasets to the robot.

The IMM shall use the instance of the *IMMToRobotInterfaceType* e.g. IMMToRobot_Interface_1.

Example

The IMM could pack the production data sets of two robots and several tempering devices in a packed container – each contained BLOB can recursively contain data of connected subcomponents…

- IMM_20200306 – Product A – Cell11.7z … BLOB production dataset entire cell
    - IMMToRobot_Interface_1 … Folder for 1st interface
        - ENGEL_Robot.dat … robot 1 BLOB-File
            - … subcomponents could be inside …
    - IMMToRobot_Interface_2 … Folder for 2nd interface
        - KUKA_Robot.bin … robot 2 BLOB-File
    - OPCUA40082.1_Interface_1 … Folder for the 1st tempering device
        - HBTHERM_Data.dat … tempering device 1 BLOB-File
    - OPCUA40082.1_Interface_2 … Folder for the 2nd tempering device
        - GWK_Data.dat … tempering device 2 BLOB-File
    - OPCUA40082.1_Interface_3 … Folder for the 3rd tempering device
        - GWK_Data.dat … tempering device 3 BLOB-File

The IMM can collect more information regarding the robot dataset calling the *Method GetProductionDatasetList* to obtain a *ProductionDatasetInformationType*.

It is recommended to use the *MachineIdentification AddIn* from “OPC 40001-1 OPC UA for Machinery, Part1: Basic Building Blocks UA for machinery” and to provide this information in the *ProductionDatasetInformationType* returned on *Method* calls of the *GetProductionDatasetList* Method.

Filling ProductionDatasetInformationType

The *ProductionDatasetInformationType* is defined in OPC 40083. In its description, references to *MachineInfomration* and *MachineConfiguration* are made. For OPC 40079, when the robot provides the *ProductionDatasetInformationType* structure the fields *Manufacturer*, *SerialNumber*, *Model* should match the entries with same name in the *Identification Object* in accordance with OPC UA for machinery. The *UserMachineName* should match the *CompontentName*.

8.8 User Login

This ObjectType allows to automatically transfer the login information of the user logged in on the IMM to the robot so that there no separate login is necessary.

The Robot as server shows the user who is currently logged in at the robot and provides *Method* s to pass the user currently logged in to the IMM to the robot. The Methods for the remote logging in/out shall be called automatically by the IMM, when a user logs in/out there. The robot can decide whether to accept the login of the IMM user via OPC UA or to deny the *Method* with an error code, e.g. if a user is already logged in at the robot itself or the IMM is not authorized.

Table 22 – UserLoginType Definiton

Attribute	**Value**				
BrowseName	UserLoginType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of the 0:*BaseObjectType* defined in OPC 10000-5					
HasComponent	Object	ActualUser		3: UserType	M
HasComponent	Method	LoginIMMUser			M
HasComponent	Method	LogoutIMMUser			M

8.8.1 ActualUser

The *ActualUser Object* holds always the actual active user on the robot system (HMI). If a remote login of an IMM user via the *Method LoginIMMUser* is successfully, the user should appear in this variable. If the IMM user is logged out via the *Method LogoutIMMUser,* the *IsPresent* property of the *ActualUser* becomes *false*.

8.8.2 LoginIMMUser

When the *Method LoginIMMUse* from the robot is called by the IMM, the requested user can be logged in to the robot system. If a user is already logged in to the robot or the passed arguments cannot be interpreted or supported, the server can deny the *Method* and *ActualUser* remains unchanged.

Signature

```
LoginIMMUser (
   [in]  0:String              Id,
   [in]  0:String              Name,
   [in]  0:String              CardUid,
   [in]  0:String              UserLevel,
   [in]  0:String              UserRole,
   [in]  0:LocaleId            Language);
```

Table 23 – LoginIMMUser Method Arguments

Argument	**Description**
Id	The *Id* of the user according to 3:*UserType*
Name	The *Name* of the user according to 3:*UserType*
CardUid	The *CardUid* of the user according to 3:*UserType*
UserLevel	The *UserLevel* of the user according to 3:*UserType*
UserRole	The *UserRole* of the user according to 3:*UserType*
Language	The *Language* of the user according to 3:*UserType*

All arguments may contain empty Strings if not supported.

Table 24 – LoginIMMUser Method AddressSpace Definiton

Attribute	Value				
BrowseName	LoginIMMUser				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory

8.8.3 LogoutIMMUser

The IMM shall report to the robot via the *Method LogoutImmUser* when a user is logged out. The *IsPresent* property of the ActualUser then becomes false again.

Signature

```
LogoutIMMUser();
```

The *Method* has no *Input*- or *OutputArguments*.

Table 25 – LogoutIMMUser Method AddressSpace Definiton

Attribute	Value				
BrowseName	LogoutIMMUser				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**

9 PubSub Communication

The first version with the “OPC 40079 IMM Fixed DataSet Server Facet” and the “OPC 40079 Robot Fixed DataSet Server Facet” needs one published data set for each connection IMM/mould to a robot. The structure of the datasets is always constant. For multiple connections multiple datasets are published and subscribed.

9.1 IMM side

9.1.1 PublisherId

The *PublisherId* is a unique identifier for a Publisher within a Message Oriented Middleware. It is included in sent messages on the network for purposes of identification or filtering.

9.1.2 Configuration of Publisher

The *IMMProcessValues_Interface_1* dataset is holding all process values including position signals, limit position flags, part tracking as well as the variable *RobotMessageID_confirmed* of part 1.

If multiple connections are provided the instance names shall be *IMMProcessValues_Interface_1, IMMProcessValues_Interface_2, …*

The *PublishedDataSets* of part 1 (*IMMProcessValues_Interface_1*, …) and the datasets of future parts are grouped in the folder *OPC40079_IMMPublishedDataSets*.

It is recommended to have a separate *WriterGroup* for each connected robot.

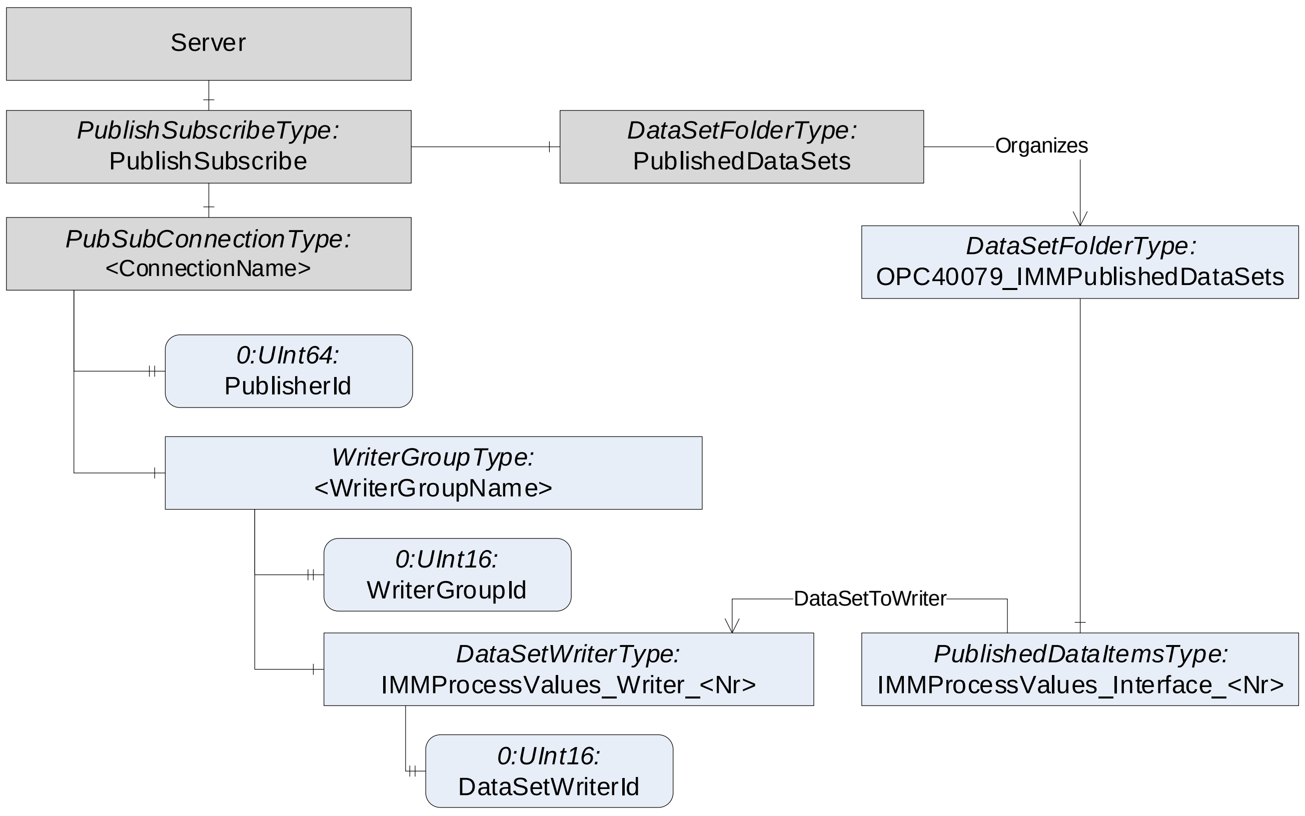

Figure 9 – PublishSubscribe for the IMM

As it is optional for the IMM to provide an OPC UA Server and an address space, the necessary parameters for the PubSub connection(PublisherId, WriterGroupId, DataSetWriterId) are put into the Method *StartPubSub* (see 8.2).

9.1.3 PublishedDataSet with IMM Process Values

The first version with the "OPC 40079 IMM Fixed DataSet Server Facet" uses a constant publishing data set on the machine side for all interfaces. The *DataSetClassId* of the fixed structure is {f0ade254-0008-4960-bbe6-eaaf6308ada2} (DataType Guid). It shall be included in the *NetworkMessages*.

```
Name:            "IMMProcessValues_Interface_<Nr>"
DataSetClassId:  {f0ade254-0008-4960-bbe6-eaaf6308ada2}

Fields: [1] Name=RobotMessageID_confirmed, Type=UInt32
        [2] Name= ReadyForOperationWithRobot, Type=Boolean
        [3] Name= EndOfOrder, Type=Boolean

        [4] Name=Mould_1.IMMPartTracking.InsertPartAvailable, Type=Boolean
        [5] Name=Mould_1.IMMPartTracking.PreMouldedPartAvailable, Type=Boolean
        [6] Name=Mould_1.IMMPartTracking.FinishedPartAvailable, Type=Boolean

        [7] Name=Mould_1.IMMPartQuality.CycleCounter, Type=UInt64
        [8] Name=Mould_1.IMMPartQuality.CycleQuality, Type=Enumeration → Int32

        [9] Name=Mould_1.MovablePlaten.InPosition1, Type=Boolean
        [10] Name=Mould_1.MovablePlaten.InPosition2, Type=Boolean
        [11] Name=Mould_1.MovablePlaten.IntermediatePosition1To2, Type=Byte
        [12] Name=Mould_1.MovablePlaten.IntermediatePosition2To1, Type=Byte
        [13] Name=Mould_1.MovablePlaten.FloatPosition, Type=Float
        [14] Name=Mould_1.MovablePlaten.PositionAdjusted, Type=Boolean
        [15] Name=Mould_1.MovablePlaten.Movement, Type=Enumeration → Int32

        [16] Name=Mould_1.Ejector_1.InPosition1, Type=Boolean
        [17] Name=Mould_1.Ejector_1.InPosition2, Type=Boolean
        [18] Name=Mould_1.Ejector_1.IntermediatePosition1To2, Type=Byte
        [19] Name=Mould_1.Ejector_1.IntermediatePosition2To1, Type=Byte
        [20] Name=Mould_1.Ejector_1.FloatPosition, Type=Float
        [21] Name=Mould_1.Ejector_1.PositionAdjusted, Type=Boolean
        [22] Name=Mould_1.Ejector_1.Movement, Type=Enumeration → Int32

        [23] Name=Mould_1.Ejector_2.InPosition1, Type=Boolean
         … // same for Ejector_2 as for Ejector_1

        [30] Name=Mould_1.Core_1.InPosition1, Type=Boolean
        [31] Name=Mould_1.Core_1.InPosition2, Type=Boolean
        [32] Name=Mould_1.Core_1.IntermediatePosition1To2, Type=Byte
        [33] Name=Mould_1.Core_1.IntermediatePosition2To1, Type=Byte
        [34] Name=Mould_1.Core_1.Movement, Type=Enumeration → Int32

        [35] Name=Mould_1.Core_2.InPosition1, Type=Boolean
         … // same for Core_2 to Core_10 as for Core_1

        [80] Name=AdditionalAxis_1.InPosition1, Type=Boolean
        [81] Name=AdditionalAxis_1.InPosition2, Type=Boolean
        [82] Name=AdditionalAxis_1.IntermediatePosition1To2, Type=Byte
        [83] Name=AdditionalAxis_1.IntermediatePosition2To1, Type=Byte
        [84] Name=AdditionalAxis_1.FloatPosition, Type=Float
        [85] Name=AdditionalAxis_1.PositionAdjusted, Type=Boolean
        [86] Name=AdditionalAxis_1.Movement, Type=Enumeration → Int32
```

9.2 Robot side

9.2.1 PublisherId

The *PublisherId* is a unique identifier for a Publisher within a Message Oriented Middleware. It is included in sent messages on the network for purposes of identification or filtering.

9.2.2 Configuration of Publisher

The robot publishes the *RobotSignals_Interface_1* dataset holding the enable and part tracking as well as the *RobotMessageID* for the first connection.

If multiple connections are provided the instance names shall be *RobotSignals_Interface_1, RobotSignals_Interface_2, …*

The *PublishedDataSets* instance of part 1 *RobotSignals_Interface_1*, … and datasets of following parts are grouped in the folder *OPC40079_RobotPublishedDataSets*.

It is recommended to have a separate *WriterGroup* for each connected IMM.

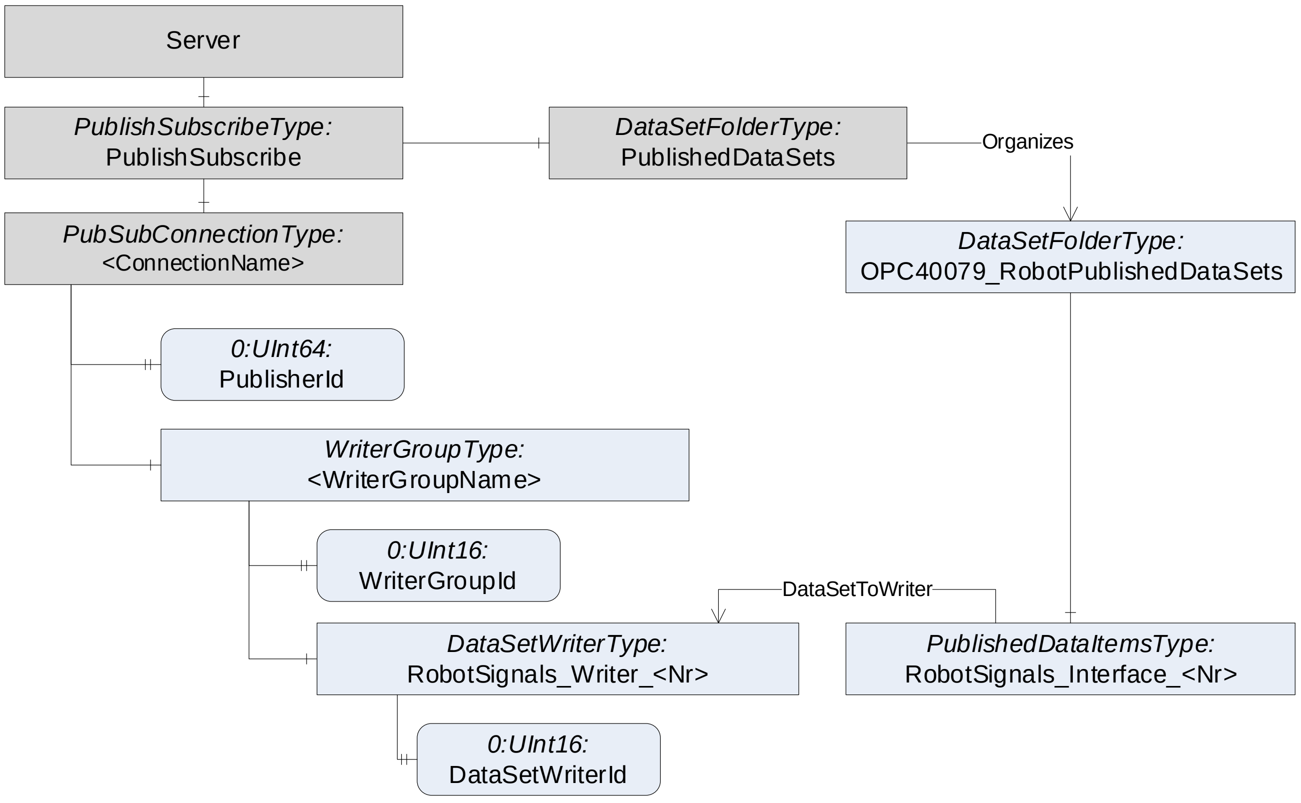

Figure 10 – PublishSubscribe for the robot

9.2.3 PublishedDataSet with Robot Signals

The first version with the “OPC 40079 Robot Fixed DataSet Server Facet” uses one constant publishing dataset on the robot side for all interfaces.

The *DataSetClassId* of the fixed structure is {830f4c53-2b6e-40fe-84b5-78abd16f08ed} (DataType Guid). It shall be included in the *NetworkMessages*.

```
Name:             "RobotSignals_Interface_<Nr>"
DataSetClassId:   {830f4c53-2b6e-40fe-84b5-78abd16f08ed}

Fields:  [1] Name=RobotMessageID, Type=UInt32
         [2] Name= ReadyForOperationWithIMM, Type=Boolean

         [3] Name=EnableMould_1.MouldAreaFree, Type=Boolean

         [4] Name=EnableMould_1.RobotPartTracking.InsertPartInserted, Type=Boolean
         [5] Name=EnableMould_1.RobotPartTracking.InsertPartRemoved, Type=Boolean
         [6] Name=EnableMould_1.RobotPartTracking.PremouldedPartInserted, Type=Boolean
         [7] Name=EnableMould_1.RobotPartTracking.PremouldedPartRemoved, Type=Boolean
         [8] Name=EnableMould_1.RobotPartTracking.FinishedPartRemoved, Type=Boolean

         [9] Name=EnableMould_1.RobotPartQuality.CycleCounter, Type=UInt64
         [10] Name=EnableMould_1.RobotPartQuality.CycleQuality, Type=Enumeration → Int32

         [11] Name=EnableMould_1.EnableMovablePlaten.RelevantForInteraction, Type=Boolean
         [12] Name=EnableMould_1.EnableMovablePlaten.EnableToPosition1, Type=Boolean
         [13] Name=EnableMould_1.EnableMovablePlaten.EnableToPosition2, Type=Boolean
         [14] Name=EnableMould_1.EnableMovablePlaten.EnablerIntermediatePosition1To2, Type=Byte
         [15] Name=EnableMould_1.EnableMovablePlaten.EnableIntermediatePosition2To1, Type=Byte

         [16] Name=EnableMould_1.EnableEjector_1.RelevantForInteraction, Type=Boolean
          …    // same as for EnableMovablePlaten

         [21] Name=EnableMould_1.EnableEjector_2.RelevantForInteraction, Type=Boolean
          …

         [26] Name=EnableSignals.Core_1.RelevantForInteraction, Type=Boolean
          …
         [71] Name=EnableSignals.Core_10.RelevantForInteraction, Type=Boolean     …
          …
         [76] Name=EnableSignals.EnableAdditionalAxis_1.RelevantForInteraction, Type=Boolean    …
          …
         [80] Name=EnableSignals.EnableAdditionalAxis_1.EnableIntermediatePosition2To1, Type=Boolean
```

10 Profiles and ConformanceUnits

10.1 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for OPC 40079.

Table 26 – Conformance Units for OPC 40079 – IMM side

Category	Title	Description
Server	OPC 40079 IMMToRobot Entry Point	Support of *IMMToRobotType* with all mandatory children as entry point. The instance(s) are located under an *Object*, which represents the IMM and which is directly located under the *Machines Object* and has an *Object Identification* as defined in OPC 40001-1.
Server	OPC 40079 IMM Fixed Position Signals	Support of the following instances in *IMMToRobotType*: - one mould *Mould_1* - two ejectors *Ejector_1* and *Ejector_2* with support of all optional children of *AxisType* - ten cores *Core_1 … Core_10* with support of *InPosition1*, *InPosition2*, *IntermediatePosition1To2*, *IntermediatePosition2To1* and *Movement* of *AxisType* - one additional axis *AdditionalAxis_1* with support of all optional of *AxisType*
Server	OPC 40079 IMM Part Tracking	Support of *IMMPartTrackingType* and provision of the *IMMtPartTracking* instance in the *MouldType*
Server	OPC 40079 IMM Part Quality	Support of *IMMPartQualityType* and provision of the *IMMPartQuality* instance in the *MouldType*
Server	OPC 40079 IMM Fixed DataSet	Support of publishing the fixed *DataSet* with *DataSetClassId* `{f0ade254-0008-4960-bbe6-eaaf6308ada2}` (see 9.1.3)

Table 27 – Conformance Units for OPC 40079 – robot side

Category	Title	Description
Server	OPC 40079 RobotToIMM Entry Point	Support of *RobotToIMMType* with all mandatory children as entry point. The instance(s) are located under an *Object*, which represents the robot and which is directly located under the *Machines Object* and has an *Object Identification* as defined in OPC 40001-1.
Server	OPC 40079 Robot Fixed Enable Signals	Support of the following instances in *RobotToIMMType*: - one instance *MouldInteraction_1* - use of *EnableMovablePlaten* - two ejectors *EnableEjector_1* and *EnableEjector_2* with support of all optional children of *EnableType* - ten cores Core_1 … Core_10 support of *InPosition1*, *InPosition2*, *IntermediatePosition1To2*, *IntermediatePosition2To1* and *Movement* of *AxisType* - one additional axis *EnableAdditionalAxis_1* with support of all optional of *AxisType*
Server	OPC 40079 Robot Part Tracking	Support of *RobotPartTrackingType* and provision of the *RobotPartTracking* instance in the *MouldInteractionType*
Server	OPC 40079 Robot Part Quality	Support of *RobotPartQualityType* and provision of the *RobotPartQuality* instance in the *MouldInteractionType*
Server	OPC 400779 Robot ProductionDatasetManagement	Support the ProductionDatasetManagement Object for the management and transfer of production datasets between IMM and robot is provided
Server	OPC 400779 Robot User Login	Support the ProductionDatasetManagement Object for the management and transfer of production datasets between IMM and robot is provided
Server	OPC 40079 Robot Fixed DataSet	Support of publishing the fixed *DataSet* with *DataSetClassId* {`830f4c53-2b6e-40fe-84b5-78abd16f08ed`} (see 9.2.3)

10.2 Profiles

10.2.1 Profile list

Table 28 lists all Profiles defined in this document and defines their URIs.

Table 28 – Profile URIs for OPC 40079-1

Profile	**URI**
OPC 40079 IMM Basic Server Profile	http://opcfoundation.org/UA-Profile/PlasticsRubber/IMM2Robot/Server/IMMBasic
OPC 40079 IMM Part1 Server Facet	http://opcfoundation.org/UA-Profile/PlasticsRubber/IMM2Robot/Server/IMMPart1
OPC 40079 Robot Basic Server Profile	http://opcfoundation.org/UA-Profile/PlasticsRubber/IMM2Robot/Server/RobotBasic
OPC 40079 Robot Part1 Server Facet	http://opcfoundation.org/UA-Profile/PlasticsRubber/IMM2Robot/Server/RobotPart1

NOTE: The names of the supported profiles are available in the *Server Object* under *ServerCapabilities.ServerProfileArray*

10.2.2 Server Facets for IMM

10.2.2.1 OPC 40079 IMM Basic Server Profile

This is the basic profile necessary on the IMM to establish an OPC 40079 connection.

Table 29 – OPC 40079 IMM Basic Server Profile Definition

Table 30 – OPC 40079 IMM Basic Server Profile

Group	**Conformance Unit / Profile Title**	**Mandatory / Optional**
Profile	BaseDevice_Server_Facet (defined in OPC UA Part 100)	M
Profile	Publisher UADP Periodic Fixed Layout Facet (defined in OPC UA Part 7)	M
IMM2Robot	OPC 40079 IMMToRobot Entry Point	M
IMM2Robot	OPC 40079 IMM Part Tracking	O
IMM2Robot	OPC 40079 IMM Part Quality	O

10.2.2.2 OPC 40079 IMM Part 1 Server Facet

This facet is used for part 1 with a fixed structure of signals to be exchanged between IMM and robot.

Table 31 – OPC 40079 IMM Fixed DataSet Server Facet

Group	**Conformance Unit / Profile Title**	**Mandatory / Optional**
IMM2Robot	OPC 40079 IMM Basic Server Profile	**M**
IMM2Robot	OPC 40079 IMM Fixed Position Signals	M
IMM2Robot	OPC 40079 IMM Part Tracking	M
IMM2Robot	OPC 40079 IMM Part Quality	M
IMM2Robot	OPC 40079 IMM Fixed DataSet	M

10.2.3 Server Facets for robot

10.2.3.1 OPC 40079 Robot Basic Server Profile

This is the basic profile necessary on the robot to establish an OPC 40079 connection.

Table 32 – OPC 40079 Robot Basic Server Profile

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	BaseDevice_Server_Facet (defined in OPC UA Part 100)	M
Profile	Publisher UADP Periodic Fixed Layout Facet (defined in OPC UA Part 7)	M
Profile	Method Server Facet (defined in OPC UA Part 7)	M
IMM2Robot	OPC 40079 RobotToIMM Entry Point	M
IMM2Robot	OPC 40079 Robot Part Tracking	O
IMM2Robot	OPC 40079 Robot Part Quality	O
IMM2Robot	OPC 400779 Robot ProductionDatasetManagement	O
IMM2Robot	OPC 400779 Robot User Login	O

10.2.3.2 OPC 40079 Robot Part 1 Server Facet

This facet is used for part 1 with a fixed structure of signals to be exchanged between IMM and robot.

Table 33 – OPC 40079 Robot Fixed DataSet Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
IMM2Robot	OPC 40079 Robot Basic Server Profile	M
IMM2Robot	OPC 40079 Robot Fixed Enable Signals	M
IMM2Robot	OPC 40079 Robot Part Tracking	M
IMM2Robot	OPC 40079 Robot Part Quality	M
IMM2Robot	OPC 40079 Robot Fixed DataSet	M

11 Namespaces

11.1 Namespace Metadata

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC 10000-6.

Table 34 defines the namespace metadata for this document. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See OPC 10000-5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in OPC 10000-5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC 10000-6.

Table 34 – NamespaceMetadata Object for this Document

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/PlasticsRubber/IMM2Robot/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/PlasticsRubber/IMM2Robot/
NamespaceVersion	String	RC 1.00.00
NamespacePublicationDate	DateTime	2021-10-28
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType []	0
StaticNumericNodeIdRange	NumericRange []	
StaticStringNodeIdPattern	String	

11.2 Handling of OPC UA Namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A *Node* in the UA *AddressSpace* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*. They are used to build a browse path between two *Nodes* or to define a standard *Property*.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* shall have the namespace of the standards body although the namespace of the *NodeId* reflects something else, for example the *EngineeringUnits Property*. All *NodeIds* of *Nodes* not defined in this document shall not use the standard namespaces.

Table 35 provides a list of mandatory and optional namespaces used in an OPC 40079 OPC UA *Server*.

Table 35 – Namespaces used in a OPC 40079 Server

NamespaceURI	Description	Use
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in an AutoID Device represented by the Server. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.	Mandatory
http://opcfoundation.org/UA/PlasticsRubber/GeneralTypes/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 40083. The namespace index is server specific.	Mandatory
http://opcfoundation.org/UA/PlasticsRubber/IMM2Robot	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.	Mandatory
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.	Optional
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.	Mandatory

Table 36 provides a list of namespaces and their indices used for *BrowseNames* in this document. The default namespace of this document is not listed since all *BrowseNames* without prefix use this default namespace.

Table 36 – Namespaces used in this document

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:NodeVersion
http://opcfoundation.org/UA/DI/	2	2:DeviceSet
http://opcfoundation.org/UA/PlasticsRubber/GeneralTypes/	3	3:CycleQualityEnumeration
http://opcfoundation.org/UA/Machinery	4	4:MachineIdentificationType

Annex A (normative)

OPC 40079-1 Namespace and mappings

A.1 Namespace and identifiers for OPC 40079-1 Information Model

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this specification. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

Where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *BrowsePath* for an *Instance Node* is constructed by appending the *BrowseName* of the instance *Node* to the *BrowseName* for the containing instance or type. An underscore character is used to separate each *BrowseName* in the path. Let's take for example, the *AxisType ObjectType Node* which has the *PositionAdjusted Property*. The **Name** for the *PositionAdjusted InstanceDeclaration* within the *AxisType* declaration is: *AxisType_PositionAdjusted*.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/PlasticsRubber/IMM2Robot/

The CSV released with this version of the specification can be found here:

– http://www.opcfoundation.org/UA/schemas/PlasticsRubber/IMM2Robot/RC1.00.0/NodeIds.csv [not available yet]

NOTE The latest CSV that is compatible with this version of the specification can be found here:

– http://www.opcfoundation.org/UA/schemas/PlasticsRubber/IMM2Robot/NodeIds.csv [not available yet]

A computer processible version of the complete Information Model defined in this specification is also provided. It follows the XML Information Model schema syntax defined in OPC 10000-6.

The Information Model Schema for this version of the document (including any revisions, amendments or errata) can be found here:

– http://www.opcfoundation.org/UA/schemas/PlasticsRubber/IMM2Robot/RC1.00.0/Opc.Ua.PlasticsRubber.IMM2Robot.NodeSet2.xml [not available yet]

NOTE The latest Information Model schema that is compatible with this version of the specification can be found here: http://www.opcfoundation.org/UA/schemas/PlasticsRubber/IMM2Robot//Opc.Ua.PlasticsRubber.IMM2Robot.NodeSet2.xml [not available yet]

Annex B
(informative)

Network aspects for interoperability

These recommendations shall be used to achieve plug and play behaviour in the field. Complex production cell with more machines and/or robot can provide detailed settings.

B.1 Preferred network connectors and speed

The Ethernet ports shall support at least 100Mbit/s. If 1000Mbit/s is supported Auto-Negotiation shall be supported to provide the possibility of both 100Mbit/s and 1000Mbit/s connections.

– RJ45, plug, for rates 100/1000Mbit/s

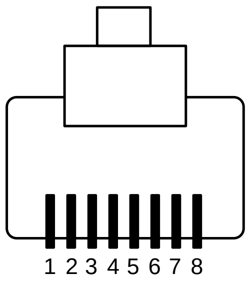

Figure 11 – RJ45 connector

– M8, plug, 4-pin A-coded, for rates 100Mbit/s

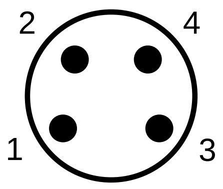

Figure 12 – M8 plug A-coded

B.2 Preferred safety connectors

Either the originated safety connector from EUROMAP 81 – or – the safety signals from the EUROMAP 67 shall be used.

B.3 IP-Addresses

The IMM shall provide a DHCP server. Each robot shall provide a DHCP client and adjust it’s IP address accordingly. If more than one IMM is connected over one network only one IMM shall be a DHCP server and the other shall use a proper static IP addresses manually assigned or use a DHCP client.

B.4 OPC UA server client communication

The OPCUA server endpoint of the robot shall be:

opc.tcp://<robot-ip>:4840/

The OPCUA server endpoint of the IMM (if provided) shall be:

opc.tcp://<imm-ip-address>:4840/

For Part 1 TLS encryption is optional. A not encrypted data exchange shall be possible. If IMM and robot support TLS encryption is shall be used.

B.5 OPC UA PubSub communication

Each IMM and each Robot can use Multicast as well as Unicast for publishing the datasets. Both should work. The PubSub multicast endpoint for IMM and Robot in the network shall be

opc.udp://239.0.0.1:4840/

The *PublisherId* of IMM and Robot in the network shall be unique – the proposal is to use the Mac-address as *PublisherId*. Example:

Table 37 – PublisherId und Mac-Address

Mac-Address of IMM	00:80:41:ae:fd:7e
IMM *PublisherId*	0x008041aefd7e [UInt64]

The *Method* call *StartPubSub* from 8.2 is used to exchange *PublisherId* from IMM to Robot and back to the IMM to achieve a bidirectional PubSub connection.

There is no upper limit for the pub sub cycle time as Part 1 defines a handshake mechanism for a safe communication. More the IMM and the robot can use different cycle times.

Nevertheless, higher cycle times speed up the handshake mechanism. To achieve at least comparable communication times as when using the EUROMAP 67 digital IO interface cycle times should be between 2 ms and 8 ms.

Annex C
(informative)

Mapping between EUROMAP 67(.1) and OPC 40079

EUROMAP 67 is the interface between IMM and robot based on hard wired signals. The following tables show where the signals defined in EUROMAP 67 are now located in the OPC 40079 information model.

Table 38 – Mapping of EUROMAP 67 signals to OPC 40079 – from IMM to robot

Signals in EUROMAP 67(.1)		**Matching Variable in OPC 40079**
Contact No.	**Signal designation**	
ZA1 ZC1	Emergency stop of machine, channel 1	*Not included in OPC 40079.* *(Use separate interface, e.g. EUROMAP 81)*
ZA2 ZC2	Emergency stop of machine channel 2	
ZA3 ZC3	Safety devices of machine channel 1	
ZA4 ZC4	Safety devices of machine channel 2	
ZA5 Optional	Reject	*IMMPartQualityType* → *CycleQuality* + *Method IMMToRobotCycleInformation* Note: *CycleQualityEnumeration* and *CavityCycleQualityEnumeration* are defined in OPC 40083
ZA6	Mould closed	*MouldType* → *MovablePlaten* → *InPosition1*
ZA7	Mould open position	*MouldType* → *MovablePlaten* → *InPosition2*
ZA8 Optional	Intermediate mould opening position	*MouldType* → *MovablePlaten* → *IntermediatePosition1To2 / IntermediatePosition2To1* Note: OPC 40079 allows 255 intermediate positions for each direction
ZB2	Enable operation with handling device / robot (Automatic)	*IMMToRobotType* → *ReadyForOperationWithRobot*
ZB3	Ejector back position	*MouldType* → *Ejector_1* → *InPosition1*
ZB4	Ejector forward position	*MouldType* → *Ejector_1* → *InPosition2*
ZB5 Optional	Core pullers 1 in position 1 (Core pullers 1 free for handling device / robot to approach)	*MouldType* → *Core_1* → *InPosition1*
ZB6 Optional	Core pullers 1 in position 2 (Core pullers 1 in position to remove moulding)	*MouldType* → *Core_1* → *InPosition2*
ZB7 Optional	Core pullers 2 in position 1 (Core pullers 2 free for handling device / robot to approach)	*MouldType* → *Core_2* → *InPosition1*
ZB8 Optional	Core pullers 2 in position 2 (Core pullers 2 in position to remove moulding)	*MouldType* → *Core_2* → *InPosition2*

Table 39 – Mapping of EUROMAP 67 signals to OPC 40079 – from robot to IMM

Signals in EUROMAP 67(.1)		Matching Variable in OPC 40079
Contact No.	**Signal designation**	
A1 C1	Emergency stop of handling device / robot Channel 1	*Not included in OPC 40079.* *(Use separate interface, e.g. EUROMAP 81)*
A2 C2	Emergency stop of handling device / robot Channel 2	
A3 C3	Mould area free	*MouldInteractionType → MouldAreaFree*
A6	Enable mould closure	*MouldInteractionType → EnableMovablePlaten → EnableToPosition1*
A7 Optional	Enable full mould opening	*MouldInteractionType → EnableMovablePlaten → EnableToPosition2*
B2	Handling device / robot operation mode (operation with handling device / robot)	*RobotToIMMType → ReadyForOperationWithIMM*
B3	Enable ejector back	*MouldInteractionType → EnableEjector_1 → EnableToPosition1*
B4	Enable ejector forward	*MouldInteractionType → EnableEjector_1→ EnableToPosition2*
B5 Optional	Enable movement of core pullers 1 to position 1 (Enable movement for handling device / robot to approach freely)	*MouldInteractionType → EnableCore_1 → EnableToPosition1*
B6 Optional	Enable movement of core pullers 1 to position 2 (Enable core pullers 1 to remove the moulding)	*MouldInteractionType → EnableCore_1 → EnableToPosition2*
B7 Optional	Enable movement of core pullers 2 to position 1 (Enable movement for handling device / robot to approach freely)	*MouldInteractionType → EnableCore_2 → EnableToPosition1*
B8 Optional	Enable movement of core pullers 2 to position 2 (Enable core pullers 2 to remove the moulding)	*MouldInteractionType → EnableCore_2 → EnableToPosition2*

EUROMAP 67.1 defines additional/modified signals for IMM with shuttle-/turntable, but which can be useful also for “normal” machines, e.g. “End of order”, “Insert part(s) inserted”. This is modeled as *AdditonalAxis* in OPC 40079.

Table 40 – Mapping of additional/modified EUROMAP 67.1 signals to OPC 40079 – from IMM to robot

Signals in EUROMAP 67(.1)		Matching Variable in OPC 40079
Contact No.	**Signal designation**	
ZA6 Optional	Start handling device/robot at position 2 (E)	Use of *AdditonalAxis_<Nr>* and its position signals
ZA7	Table in position, general	
ZA8 Optional	Start handling device/robot at position 3 (F)	
ZC5	End of order	*IMMToRobotType → EndOfOrder*
ZC6	Insert part(s) in mould	*IMMPartTrackingType → InsertPartAvailable*
ZC7	Part available	*IMMPartTrackingType → FinishedPartAvailable* (or *PreMouldedPartAvailable*)

Table 41 – Mapping of additional/modified EUROMAP 67.1 signals to OPC 40079 – from robot to IMMa

Signals in EUROMAP 67(.1)		Matching Variable in OPC 40079
Contact No.	**Signal designation**	
A3 C3	Table area free	No direct matching → Use of *EnableAdditonalAxis_<Nr>* and its signals for enabling movements
A6	Enable table motion	Use of *EnableAdditonalAxis_<Nr>* and its signals for enabling movements
C6	Insert part(s) inserted	*RobotPartTrackingType → InsertPartInserted*
C7	Moulded part(s) removed	*RobotPartTrackingType → FinishedPartRemoved* (or *PreMouldedPartRemoved*)