VDMA 40001-101

ICS 25.020; 35.240.50

Replaces
VDMA 40001-101:2023-01

OPC UA for Machinery – Part 101: Result Transfer

OPC UA for Machinery – Teil 101: Result Transfer

VDMA 40001-101:2025-07 is identical with OPC 40001-101 (Release 1.01.0)

Document comprises 32 pages

VDMA

Contents

Figures

Tables

OPC Foundation / VDMA

AGREEMENT OF USE

COPYRIGHT RESTRICTIONS

- This document is provided "as is" by the OPC Foundation and VDMA.
- Right of use for this specification is restricted to this specification and does not grant rights of use for referred documents.
- Right of use for this specification will be granted without cost.
- This document may be distributed through computer systems, printed or copied as long as the content remains unchanged and the document is not modified.
- OPC Foundation and VDMA do not guarantee usability for any purpose and shall not be made liable for any case using the content of this document.
- The user of the document agrees to indemnify OPC Foundation and VDMA and their officers, directors and agents harmless from all demands, claims, actions, losses, damages (including damages from personal injuries), costs and expenses (including attorneys' fees) which are in any way related to activities associated with its use of content from this specification.
- The document shall not be used in conjunction with company advertising, shall not be sold or licensed to any party.
- The intellectual property and copyright is solely owned by the OPC Foundation and VDMA.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC or VDMA specifications may require use of an invention covered by patent rights. OPC Foundation or VDMA shall not be responsible for identifying patents for which a license may be required by any OPC or VDMA specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC or VDMA specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

WARRANTY AND LIABILITY DISCLAIMERS

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OPC FOUDATION NOR VDMA MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OPC FOUNDATION NOR VDMA BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you.

RESTRICTED RIGHTS LEGEND

This Specification is provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor / manufacturer are the OPC Foundation, 16101 N. 82nd Street, Suite 3B, Scottsdale, AZ, 85260-1830.

COMPLIANCE

The combination of VDMA and OPC Foundation shall at all times be the sole entities that may authorize developers, suppliers and sellers of hardware and software to use certification marks, trademarks or other special designations to indicate compliance with these materials as specified within this document. Products developed using this specification may claim compliance or conformance with this specification if and only if the software satisfactorily meets the certification requirements set by VDMA or the OPC Foundation. Products that do not meet these requirements may claim only that the product was based on this specification and must not claim compliance or conformance with this specification.

TRADEMARKS

Most computer and software brand names have trademarks or registered trademarks. The individual trademarks have not been listed here.

GENERAL PROVISIONS

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

Forewords

Mechanical engineering is a broad-based industry, which is mainly associated with machines such as machine tools, woodworking machines or robots. Many other products such as measuring and testing equipment are also relevant to this field.

Since the information models in this document are intended to apply not only to machines (see in ISO 12100:2010, [1]), but to all other applications and products in the entire machinery industry. Each case cannot be represented individually, therefore the term “machine” is used uniformly for all in this document.

Compared with previous versions, the following changes have been made:

Version	Changes
OPC 40001-101 1.00 (identical with VDMA 40001-101:2022-04)	Initial release
OPC 40001-101 1.01 RC (identical with VDMA 40001-101:2025-02)	Added the optional Method AcknowledgeResults to ResultManagementType

This specification was created by a joint working group of the OPC Foundation and VDMA.

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

VDMA

The VDMA is Europe’s largest industry association with over 3300 member companies of the mechanical engineering industry. These companies integrate the latest technologies in products and processes. VDMA was founded in November 1892 and is the most important voice for the mechanical engineering industry today. With the headquarters located in Frankfurt, it represents the issues of the mechanical and plant engineering sector in Germany and Europe. The standard OPC UA has established itself in this industry sector. The VDMA defines OPC UA Companion Specifications for various sectors of the mechanical engineering industry, with more than 450 companies involved. Consequently, one of the main tasks is to harmonise and create consistency.

1 Scope

The OPC UA for Machinery specification contains various building blocks for Machinery that allow to address use cases across different types of machines and components of machines defined in various companion specifications.

For the general scope of the OPC UA for Machinery specification see OPC 40001-1.

This part contains a building block for

– Result Transfer

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 10000-20, OPC Unified Architecture – Part 20: File Transfer

http://www.opcfoundation.org/UA/Part20/

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

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

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 Machinery – Result Transfer Information Model. For the purposes of this document, the terms and definitions given in OPC 10000-1, OPC 10000-3, OPC 10000-4, OPC 10000-5, OPC 10000-7, OPC 40001-1, as well as the following apply.

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

3.2 OPC UA for Result Transfer terms

3.2.1
Configuration

Ensures that a recipe produces the same result on different entities of my machine. The configuration can optionally be used to make a program/recipe usable for different articles. The characteristics are often technology or industry specific.

3.2.2
Environment

The set of external entities working with the *System* in one way or another, e.g. PLC, MES, etc.

3.2.3
Job
The concrete implementation of a set of information that a machine needs to execute an order.

Note 1 to entry: This information can be composed of one or more programs or recipes, or it can consist only of metadata and parameters (production-specific), or of all of these.

3.2.4
Part
Describes a concrete instance or instances of the *Product* (e.g., a specific DIN A4 sheet).

Note 1 to entry: Typically has a serial number.

3.2.5
Program
Contains a number of executable activities for controllable units.

Note 1 to entry: A program can be part of a recipe. It can include metadata and parameters.

3.2.6
Product
Describes the output that is to become the core of the value creation (e.g. DIN A4 sheet).

Note 1 to entry: Typically has a unique identification.

3.2.7
Recipe
The set of information required to uniquely describe the production requirements for one or more specific products.

Note 1 to entry: It can include metadata and parameters (machine specific).

3.2.8
System
Any complex information processing system generating results.

3.2.9
System-wide unique
At any given time no other entity of the same type and meaning shall exist in the OPC UA Server with the same value.

Note 1 to entry: Used in conjunction with identifiers and handles

Note 2 to entry: No further assumptions about global or historical uniqueness are made; especially in the case of identifiers, however, globally unique identifiers are recommended.

3.2.10
Step
A domain- or manufacturer-specific definition that represents an intermediate stage of the job process.

Note 1 to entry: A job can consist of several steps.

3.3
Abbreviated terms
MES Manufacturing Execution System

PLC Programmable Logic Controller

3.4
Conventions used in this document
For conventions used in this document see OPC 40001-1.

4 General information to Machinery and OPC UA

For general information to Machinery and OPC UA see OPC 40001-1.

5 Use cases

The user would like to receive complex measurement results, that might be created over a certain amount of time and potentially require some processing of raw measurement values.

This leads to the requirements

- Clients shall be notified if new results are available (see section 8.1, ResultReadyEventType)
- Clients shall be able to access individual results (see section 7.1, ResultManagementType and section 7.1.2, GetResultById)
- Clients shall be able to filter for specific results (see section 7.1.3, GetResultIdListFiltered)

Note that the main purpose of the result transfer is to provide meta data together with the individual results. For a simple transfer of measured values like a flow or temperature without such meta data there are simpler mechanisms in OPC UA like providing a OPC UA Variable with a specific measurement value.

6 Result Transfer Information Model overview

6.1 General

The result transfer information model provides mechanisms to transfer results that are produced by a *Server* or its underlying system. The characteristics of such results is to contain meta data together with the individual results. *Servers* will provide the results for a certain amount of time so that *Clients* can receive the results using the result transfer.

The base modelling construct for the result transfer is the *ResultManagementType* (see Figure 1). *Servers* may provide several instances of that *ObjectType* to expose different sources of results. Each instance of *ResultManagementType* already can provide many results.

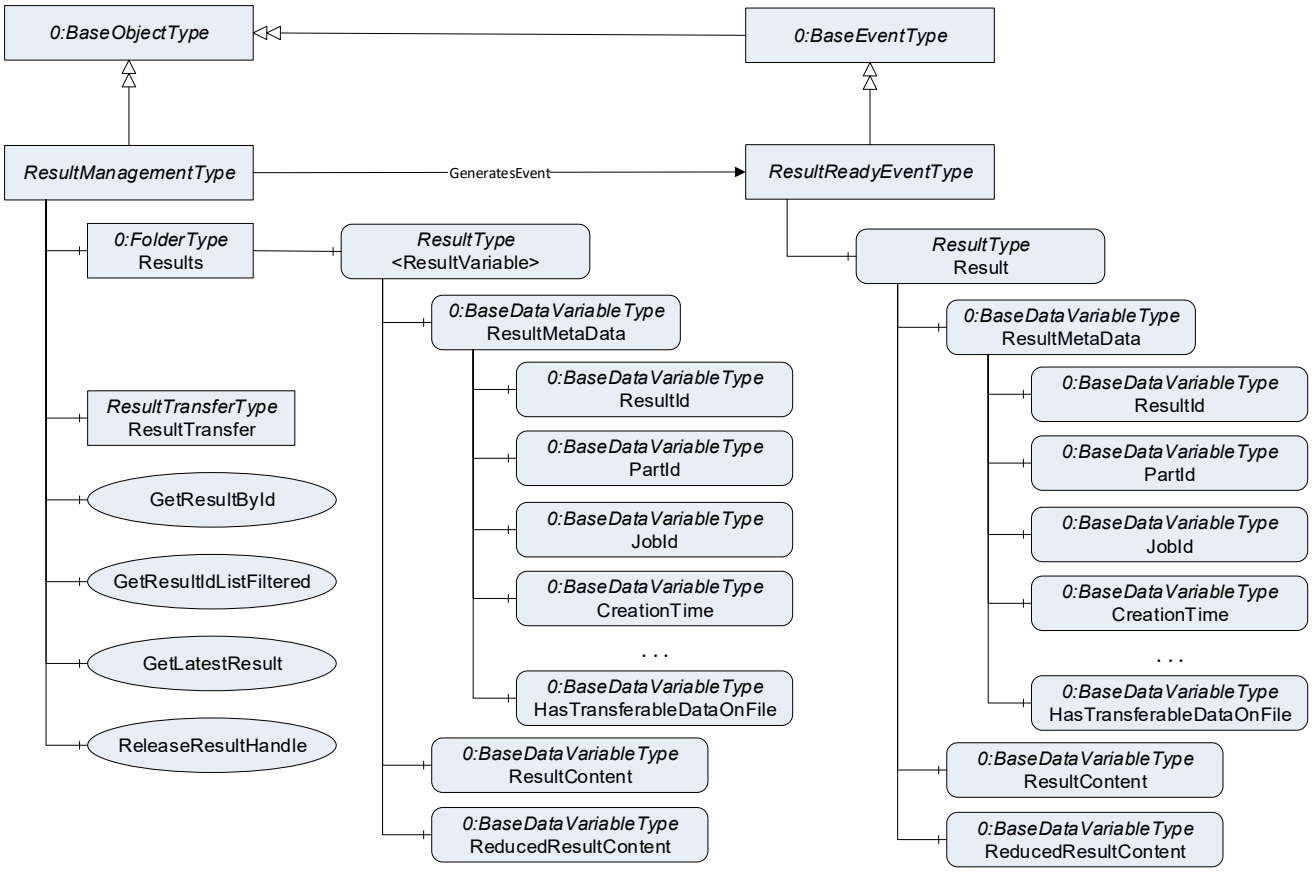

Figure 1 – Overview Result Transfer Information Model

Instances of *ResultManagementType* provide optionally all available results as *Variables* of *ResultType* under the *Results* folder. They may generate *Events* of *ResultReadyEventType* for new created results. And they provide optional *Methods* to retrieve individual results or filter for specific results.

6.2 Retrieve Information that Results exist

The model provides different mechanism how *Clients* can retrieve information about the currently available results provided by instances of the *ResultManagementType*. Not all implementations will provide all those possibilities.

The mechanisms are depicted in Figure 2. In order to get notified that a new result has been created, a *Client* can subscribe to *Events* of *ResultReadyEventType* if the *Server* supports those *Events*. In order to retrieve information about the available results, the *Client* can

- Browse the *Results* folder, each *Variable* of *ResultType* represents an individual available result
- Call the *GetLatestResult Method* to retrieve information about the latest created result
- Call the *GetResultIdListFiltered Method* to retrieve information about all available results according to the specified filter criteria

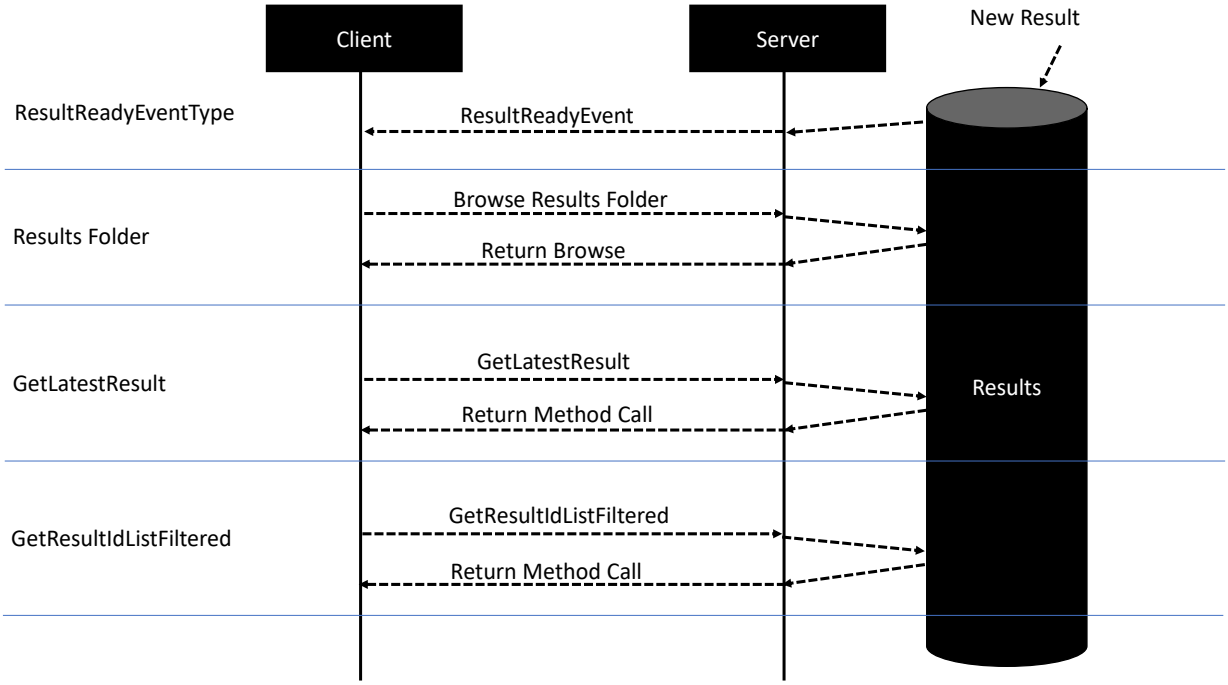

Figure 2 – Retrieve Information that Results exist

6.3 Transfer Results

The model provides different mechanism how *Clients* can transfer results provided by instances of the *ResultManagementType*. Not all implementations will provide all those possibilities.

The result consists of meta data describing the result, a relatively small amount of result data called *ResultContent* and optionally a larger amount of result data called *ResultFile*, transferred via the OPC UA file transfer. In some cases, the result only consists of parts of the meta data, the *ResultEvaluation*, which indicates whether the result is in a tolerance.

If a result has been successfully created, the *Server* shall at least provide the *ResultContent*, the *ResultFile* or the *ResultEvaluation*.

The *ResultReadyEvents* provide the meta data of the result as well as the *ResultContent*. *Clients* can either only subscribe to parts of the meta data and access the *ResultContent* via a different mechanism or get the *ResultContent* as well with the *Event*.

The *ResultContent* as well as the meta data can be transferred by calling the *GetResultById Method*. This requires knowledge on the *ResultId*, that can either be retrieved via the *ResultReadyEvents* or by calling the *GetResultIdListFiltered Method*. The latest *ResultContent* and its meta data can also be transferred by calling the *GetLatestResult Method*.

The *ResultContent* as well as the meta data is also provided by *Variables* under the *Results* folder and can be transferred by reading or subscribing to those *Variables*.

In Figure 3, the different options are displayed.

In case the result also contains a *ResultFile*, the *ResultTransfer Object* may be used to transfer the *ResultFile*. In order to use the *ResultTransfer* Object, the *ResultId* of the result is needed.

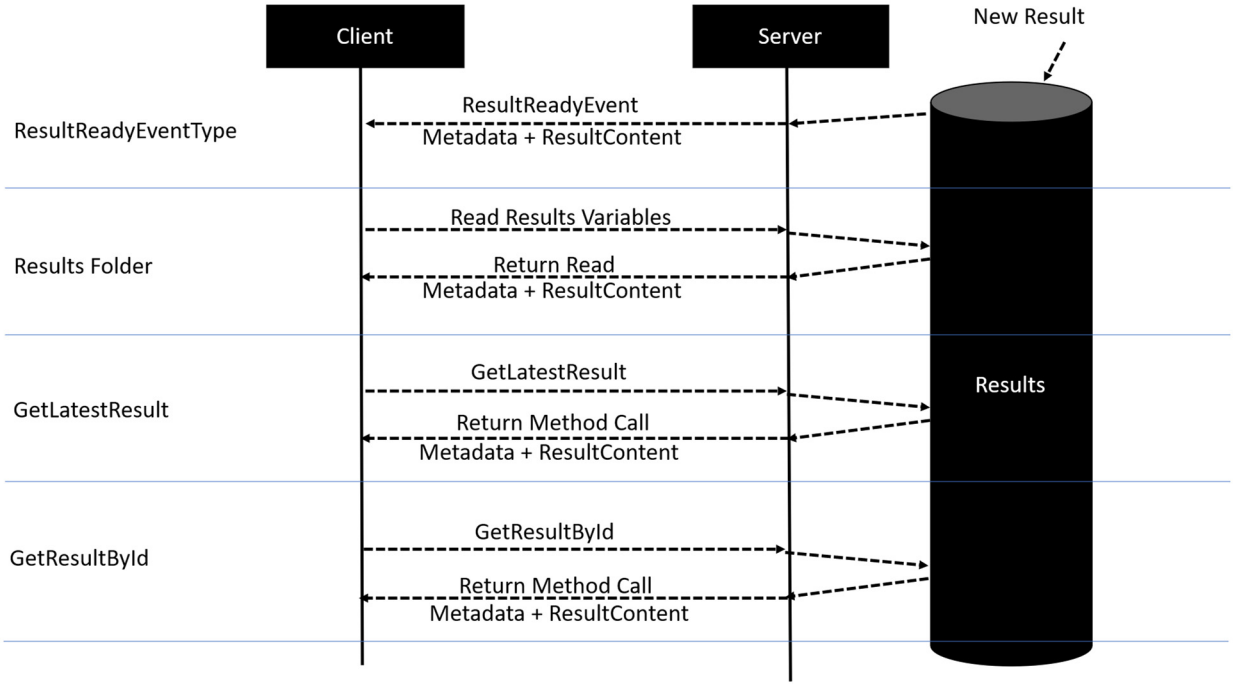

Figure 3 – Transfer ResultContent and Metadata

6.4 Lifetime of a Result

The lifetime of a result typically is limited as the management of the results is consuming resources of the *Server*. When using the *Methods GetResultById*, *GetLatestResult* or *GetResultIdListFiltered*, the *Client* can provide a timer how long it requests the retrieved results to be available, and receives a handle it can use to call the *ReleaseResultHandle Method* in order to hint to the *Server*, that the result is not required by the *Client* anymore. The mechanism is intended to secure the read process of results and not to lock a specific result for a longer time. There is no guarantee that the *Server* will keep the result until the *RelaseResultHandle* is called or the timeout is reached. The strategy of the *Server* is vendor-specific. A *Server* may, for example, stop producing new results when the old results should still be kept, or remove old results anyhow. A *Server* may remove results as soon as one *Client* has received the result data, or keep the results for a specific time since many *Clients* may need to access the result data.

Note: Since there is no guaranteed storage time even a *Server* without the capability to store more than one result locally can use this mechanism.

The handle and timeout are of specific importance when the result contains a *ResultFile*, or when the *GetResultIdListFiltered* was called and not all returned *ResultIds* have been received via the *GetResultById Method*.

Servers may support the *AcknowledgeResults Method* that can be used by a specific *Client* responsible to store the results to inform the Server, that the acknowledged results are processed and can be removed.

6.5 Partial Results and Result Content

Generating results may take time, and during that period *Servers* may already expose partial results, i.e., results that do contain some information but are not complete. For example, a vision system may inspect a produced part several times from different angles, and each inspection is a partial result. The meta data of the result indicate, when the result is only a partial result. The final result contains the aggregated data of the partial results.

The *ResultContent* or the *ResultEvaluation* and potentially in addition the *ResultFile* together provide the complete result. *Servers* may also provide a reduced result summarizing the overall result. For example, a vision system inspecting a produced part may provide in the result the raw image data of the product part in addition to the flag indicating if the quality of the produced part is sufficient. The *ReducedResultContent* may only contain the quality information without the raw image data. The optional *ReducedResultContent* is provided with the *ResultReadyEvents* and the *Variables* of the *Results* folder.

7 OPC UA ObjectTypes

7.1 ResultManagementType ObjectType Definition

7.1.1 Overview

The *ResultManagementType* provides mechanism to access results generated by the underlying system. Results can be managed in a local result store of the *Server*. *Methods* and *Objects* with *Variables* as well as *Events* and external file stores can be used to provide the results to the *Client*. The *ObjectType* is formally defined in Table 1.

Table 1 – ResultManagementType Definition

Attribute	**Value**				
BrowseName	ResultManagementType				
IsAbstract	False				
Description	Provides mechanism to access results generated by the underlying system.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:BaseObjectType defined in OPC 10000-5					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	
0:HasComponent	Method	GetResultById			O
0:HasComponent	Method	GetResultIdListFiltered			O
0:HasComponent	Method	ReleaseResultHandle			O
0:HasComponent	Method	GetLatestResult			O
0:HasComponent	Object	ResultTransfer		ResultTransferType	O
0:HasComponent	Object	Results		0:FolderType	O
0:HasComponent	Method	AcknowledgeResults			O
0:GeneratesEvent	ObjectType	ResultReadyEventType			
Conformance Units					
Machinery-Result Types					

The optional *ResultTransfer Object* is used to transfer the content of a result by a temporary file transfer. In order to be called by the *Client*, the *Client* first need to receive a *ResultId*.

The optional *Results Object* is used to organize available results of the system which the *Server* decides to expose in the *AddressSpace*. It may contain no result if no result is available, if the *Server* does not expose results in the *AddressSpace* at all or if no available result matches the criteria of the *Server* for exposure in the *AddressSpace*. It may contain multiple results if multiple results are available.

The components of the *ResultManagementType* have additional subcomponents which are defined in Table 2.

Table 2 – ResultManagementType Additional Subcomponents

Source Path	**Reference**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
Results	0:HasComponent	Variable	<ResultVariable>	ResultDataType	ResultType	OP

Each <ResultVariable> represents a result. If partial results are supported, the *Variable* value gets updated with partial results until the final result is provided as value.

The child *Nodes* of the *ResultManagementType* have additional *Attribute* values defined in Table 3.

Table 3 – ResultManagementType Attribute values for child Nodes

BrowsePath	Value Attribute	Description Attribute
0:DefaultInstanceBrowseName	ResultManagement	The default BrowseName for instances of the type.

7.1.2 GetResultById

The *Method GetResultById* is used to retrieve a result. Depending on the design of the system, the *Client* may be informed by *Events* of *ResultReadyEventType* that a new result is available. Then, the *Client* might fetch this result using the information provided by *Events* of *ResultReadyEventType* which is defined in Section 8.1.

Since the *ResultId* is supposed to be system-wide unique, this *Method* shall return only a single result. If the system supports partial results, only the latest partial result, or the final result, shall be returned.

Since there may be additional result content to be retrieved by temporary file transfer, the *Server* should keep result data available, resources permitting, until the *Client* releases the handle by calling *ReleaseResultHandle*. However, the *Client* cannot rely on the data to remain available until then.

The signature of this *Method* is specified below. Table 4 and Table 5 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
GetResultById (
   [in]    0:TrimmedString  resultId,
   [in]    0:Int32          timeout,
   [out]   0:Handle         resultHandle,
   [out]   ResultDataType   result,
   [out]   0:Int32          error)
```

Table 4 – GetResultById Method Arguments

Argument	Description
resultId	System-wide unique identifier for the result.
timeout	With this argument the Client can give a hint to the Server how long it will need access to the result data. A value > 0 indicates an estimated maximum time for processing the data in milliseconds. A value = 0 indicates that the Client will not need anything besides the data returned by the method call. A value < 0 indicates that the Client cannot give an estimate. The Client cannot rely on the data being available during the indicated time period. The argument is merely a hint allowing the Server to optimize its resource management.
resultHandle	The Server shall return to each Client requesting result data a system-wide unique handle identifying the result set / Client combination. This handle should be used by the Client to indicate to the Server that the result data is no longer needed, allowing the Server to optimize its resource handling. If the instance of ResultManagementType does not support the ReleaseResultHandle Method, the resultHandle should always be set to 0. If the error is set to a value other than 0, the resultHandle may be set to 0.
result	The result including metadata. May be set to Null, if error is set to a value other than 0.
error	0 – OK Values > 0 are reserved for errors defined by this and future standards. Values < 0 shall be used for application-specific errors.

Table 5 – GetResultById Method AddressSpace Definition

Attribute	Value				
BrowseName	GetResultById				
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

7.1.3 GetResultIdListFiltered

The *Method GetResultIdListFiltered* is used to get a list of results matching certain filter criteria. The signature of this *Method* is specified below. Table 6 and Table 8 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
GetResultIdListFiltered (
   [in]    0:ContentFilter          filter,
   [in]    0:RelativePath[]         orderedBy,
   [in]    0:UInt32                 maxResults,
   [in]    0:Int32                  timeout,
   [out]   0:Handle                 resultHandle,
   [out]   0:TrimmedString[]        resultIdList,
   [out]   0:Int32                  error)
```

Table 6 – GetResultIdListFiltered Method Arguments

Argument	Description
filter	Filter used to filter for specific results based on the meta data of the results. Valid BrowsePaths used in the filter can be built from the fields of the ResultReadyEventType, the ResultType VariableType or the ResultDataType or corresponding subtypes.
orderedBy	An array of RelativePaths identifying the ordering criteria for the results. If the array is null or empty, no ordering is executed. If several RelativePaths are provided, the first entry in the array is used as first ordering criteria, etc.
maxResults	Defines how many resultIds the Client wants to receive at most. If no maximum should be provided, it is set to 0.
timeout	With this argument the Client can give a hint to the Server how long it will need access to the result data. A value > 0 indicates an estimated maximum time for processing the data in milliseconds. A value = 0 indicates that the Client will not need anything besides the data returned by the method call. A value < 0 indicates that the Client cannot give an estimate. The Client cannot rely on the data being available during the indicated time period. The argument is merely a hint allowing the Server to optimize its resource management.
resultHandle	The Server shall return to each Client requesting result data a system-wide unique handle identifying the result set / Client combination. This handle has to be used by the Client to release the result set. If the instance of ResultManagementType does not support the ReleaseResultHandle Method, the resultHandle should always be set to 0. If the error is set to a value other than 0, the resultHandle may be set to 0.
resultIdList	List of resultIds of results matching the Filter.
error	0 – OK Values > 0 are reserved for errors defined by this and future standards. Values < 0 shall be used for application-specific errors.

Table 7 – GetResultIdListFiltered Method AddressSpace Definition

Attribute	Value				
BrowseName	GetResultIdListFiltered				
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

Table 8 – GetResultIdListFiltered Method AddressSpace Definition

Attribute	Value				
BrowseName	GetResultIdListFiltered				
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

7.1.4 ReleaseResultHandle

The *Method ReleaseResultHandle* is used to inform the *Server* that the *Client* has finished processing a given result set allowing the *Server* to free resources managing this result set.

The *Server* should keep the data of the result set available for the *Client* until the *ReleaseResultHandle Method* is called or until a timeout given by the *Client* has expired. However, the *Server* is free to release the data at any time, depending on its internal resource management, so the *Client* cannot rely on the data being available. *ReleaseResultHandle* is merely a hint allowing the *Server* to optimize its internal resource management. For timeout usage see the description in Section 6.4.

The signature of this *Method* is specified below. Table 9 and Table 10 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
ReleaseResultHandle (
   [in]    0:Handle          resultHandle,
   [out]   0:Int32           error)
```

Table 9 – ReleaseResultHandle Method Arguments

Argument	Description
resultHandle	Handle returned by *GetResultById* or *GetResultIdListFiltered*, identifying the result set/Client combination.
Error	0 – OK Values > 0 are reserved for errors defined by this and future standards. Values < 0 shall be used for application-specific errors.

Table 10 – ReleaseResultHandle Method AddressSpace Definition

Attribute	Value				
BrowseName	ReleaseResultHandle				
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

7.1.5 GetLatestResult

The *Method GetLatestResult* is used to retrieve the latest result created for the instance of *ResultManagementType*.

Since there may be additional result content to be retrieved by temporary file transfer, the *Server* should keep result data available, resources permitting, until the *Client* releases the handle by calling *ReleaseResultHandle*. However, the *Client* cannot rely on the data to remain available until then.

The signature of this *Method* is specified below. Table 11 and Table 12 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
GetLatestResult (
   [in]    0:Int32          timeout,
   [out]   0:Handle         resultHandle,
   [out]   ResultDataType   result,
   [out]   0:Int32          error)
```

Table 11 – GetLatestResult Method Arguments

Argument	Description
timeout	With this argument the Client can give a hint to the Server how long it will need access to the result data. A value > 0 indicates an estimated maximum time for processing the data in milliseconds. A value = 0 indicates that the Client will not need anything besides the data returned by the method call. A value < 0 indicates that the Client cannot give an estimate. The Client cannot rely on the data being available during the indicated time period. The argument is merely a hint allowing the Server to optimize its resource management.
resultHandle	The Server shall return to each Client requesting result data a system-wide unique handle identifying the result set / Client combination. This handle should be used by the Client to indicate to the Server that the result data is no longer needed, allowing the Server to optimize its resource handling. If the instance of ResultManagementType does not support the ReleaseResultHandle Method, the resultHandle should always be set to 0. If the error is set to a value other than 0, the resultHandle may be set to 0.
result	The result including metadata.
error	0 – OK Values > 0 are reserved for errors defined by this and future standards. Values < 0 shall be used for application-specific errors.

Table 12 – GetLatestResult Method AddressSpace Definition

Attribute	Value				
BrowseName	GetLatestResult				
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

7.1.6 AcknowledgeResults

The *Method AcknowledgeResults* is used to inform the *Server* that the *Client* has finished the processing of given results allowing the *Server* to free resources managing those results.

The *Server* may be configured to keep the results available for the *Client* until the *AcknowledgeResults Method* is called. However, the *Server* is free to release the results at any time, depending on its internal resource management, so the *Client* cannot rely on the data being available.

Note: It is recommended that this *Method* is used only by the *Client* that is responsible for exclusively storing the results.

The signature of this *Method* is specified below. Table 13 and Table 14 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
AcknowledgeResults (
   [in]    0:TrimmedString[]   resultIds,
   [out]   0:Int32[]           errorPerResultId,
   [out]   0:Int32             error
)
```

Table 13 – AcknowledgeResults Method Arguments

Argument	Description
resultIds	List of result identifiers to be acknowledged.
errorPerResultId	Shall be null or empty if error equals 0. Shall have the same length as resultIds if error is not equal 0. Indicates for each resultId in resultIds, if the acknowledge was successful. Per entry: 0 – OK Values > 0 are reserved for errors defined by this and future standards. Values < 0 shall be used for application-specific errors.
error	0 – OK Values > 0 are reserved for errors defined by this and future standards. Values < 0 shall be used for application-specific errors. Shall be not equal 0 if any resultId of resultIds was not successfully acknowledged. Shall be 0 if all resultIds where acknowledged successful.

Table 14 – AcknowledgeResults Method AddressSpace Definition

Attribute	Value				
BrowseName	AcknowledgeResults				
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

7.2 ResultTransferType ObjectType Definition

7.2.1 Overview

This *ObjectType* is a subtype of the 0:*TemporaryFileTransferType* defined in OPC 10000-20 and is used to transfer result data as a file.

The *ResultTransferType* overwrites the *Method GenerateFileForRead* to specify the concrete type of the generateOptions Argument of the *Method*. It does not specialize the *GenerateFileForWrite Method* of the base type as results are supposed to be only generated by the *Server*, not received.

The *ResultTransferType* is formally defined in Table 15.

Table 15 – ResultTransferType Definition

Attribute	Value				
BrowseName	ResultTransferType				
IsAbstract	False				
Description	Transfers result data as a file.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:TemporaryFileTransferType defined in OPC 10000-20					
HasComponent	Method	GenerateFileForRead			M
Conformance Units					
Machinery-Result Types					

7.2.2 GenerateFileForRead

The *Method GenerateFileForRead* is used to start the read file transaction. A successful call of this *Method* creates a temporary *FileType Object* with the file content and returns the *NodeId* of this *Object* and the file handle to access the *Object*. The signature of this *Method* is specified below. Table 16 and Table 17 specify the *Arguments* and *AddressSpace* representation, respectively.

Signature

```
GenerateFileForRead (
   [in]    BaseResultTransferOptionsDataType    generateOptions,
   [out]   0:NodeId                             fileNodeId,
   [out]   0:UInt32                             fileHandle,
   [out]   0:NodeId                             completionStateMachine)
```

Table 16 – GenerateFileForRead Method Arguments

Argument	Description
generateOptions	Options how to generate the file, including the resultId of the result the file belongs to.
fileNodeId	NodeId of the temporary file.
fileHandle	The FileHandle of the opened TransferFile. The FileHandle can be used to access the TransferFile methods Read and Close.
completionStateMachine	If the creation of the file is completed asynchronously, the parameter returns the NodeId of the corresponding FileTransferStateMachineType Object. If the creation of the file is already completed, the parameter is null. If a FileTransferStateMachineType object NodeId is returned, the Read Method of the file fails until the TransferState changed to ReadTransfer.

Table 17 – GenerateFileForRead Method AddressSpace Definition

Attribute	Value				
BrowseName	0:GenerateFileForRead				
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

8 OPC UA EventTypes

8.1 ResultReadyEventType ObjectType Definition

The *ResultReadyEventType* provides information of a complete or partial result. *Events* of this type are to be triggered when the *Server* has a complete or partial result available to the *Client*. It is formally defined in Table 18.

To enable access to a result, several *Properties* of the result, which are generated by the system, are supplied. These *Properties* can be used to query the results in the *ResultManagement Object*. A sufficiently small result can also be provided in the *ResultContent* component of the *Event*.

Table 18 – ResultReadyEventType Definition

Attribute	Value				
BrowseName	ResultReadyEventType				
IsAbstract	True				
Description	Provides information of a complete or partial result.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:BaseEventType defined in OPC 10000-5					
0:HasComponent	Variable	Result	ResultDataType	ResultType	M
Conformance Units					
Machinery-Result ResultEvents					

This *EventType* inherits all *Properties* of the *BaseEventType*.

The *Result Variable* of *ResultType VariableType* contains the subvariables as defined by the *ResultType*. *Clients* can select some of those individually when subscribing to *Events*. The *Result* represents a partial or complete result.

9 OPC UA VariableTypes

9.1 ResultType VariableType Definition

The *ResultType* is a subtype of *BaseDataVariableType*. It is used to expose the information of the *ResultDataType* in individual subvariables. They have the same meaning as defined by the *DataType* in section 10.5.

The *VariableType* is formally defined in Table 19.

Table 19 – ResultType Definition

Attribute	Value				
BrowseName	ResultType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	ResultDataType				
Description	Exposes the information of the ResultDataType in individual subvariables.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:BaseDataVariableType defined in OPC 10000-5					
0:HasStructuredComponent	Variable	ResultMetaData	ResultMetaDataType	0:BaseDataVariableType	M
0:HasStructuredComponent	Variable	ResultContent	0:BaseDataType[]	0:BaseDataVariableType	O
0:HasComponent	Variable	ReducedResultContent	0:BaseDataType[]	0:BaseDataVariableType	O
Conformance Units					
Machinery-Result Types					

The optional *ReducedResultContent Variable*, not defined in the *ResultDataType*, contains a reduced *ResultContent* summarizing the overall result (see 6.5).

The components of the ResultType have additional subcomponents which are defined in Table 20.

Table 20 – ResultType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
ResultMetaData	0:HasStructuredComponent	Variable	ResultId	0:TrimmedString	0:BaseDataVariableType	M
ResultMetaData	0:HasStructuredComponent	Variable	HasTransferableDataOnFile	0:Boolean	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	IsPartial	0:Boolean	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	IsSimulated	0:Boolean	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ResultState	0:Int32	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	StepId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	PartId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ExternalRecipeId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	InternalRecipeId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ProductId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ExternalConfigurationId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	InternalConfigurationId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	JobId	0:TrimmedString	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	CreationTime	0:UtcTime	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ProcessingTimes	ProcessingTimes DataType	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ResultUri	0:UriString[]	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ResultEvaluation	ResultEvaluation Enum	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ResultEvaluationDetails	0:LocalizedText	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	ResultEvaluationCode	0:Int64	0:BaseDataVariableType	O
ResultMetaData	0:HasStructuredComponent	Variable	FileFormat	0:String[]	0:BaseDataVariableType	O

10 OPC UA DataTypes

10.1 BaseResultTransferOptionsDataType

This abstract structure contains information on which *ResultFile* should be provided to be transferred to the *Client*. The structure contains the *ResultId* of the requested result. Companion Specifications and vendors shall either subtype this structured *DataType* to provide more information about the provided *ResultFile*, like the file format, or use an existing subtype like *ResultTransferOptionsDataType*. The structure is defined in Table 21.

Table 21 – BaseResultTransferOptionsDataType Structure

Name	Type	Description
BaseResultTransferOptionsDataType	structure	
ResultId	0:TrimmedString	The Id of the result to be transferred to the Clients

Its representation in the *AddressSpace* is defined in Table 22.

Table 22 – BaseResultTransferOptionsDataType Definition

Attribute	Value				
BrowseName	BaseResultTransferOptionsDataType				
IsAbstract	True				
Description	Abstract type containing information which file should be provided.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure defined in OPC 10000-5					
Conformance Units					
Machinery-Result Types					

10.2 ResultTransferOptionsDataType

This structure contains information on which *ResultFile* should be provided to be transferred to the *Client*. The structure is a concrete DataType that can be used directly if no additional information needs to be added. No additional fields have been added to the structure.

Its representation in the *AddressSpace* is defined in Table 23.

Table 23 – ResultTransferOptionsDataType Definition

Attribute	Value				
BrowseName	ResultTransferOptionsDataType				
IsAbstract	False				
Description	Contains information which file should be provided.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of BaseResultTransferOptionsDataType					
Conformance Units					
Machinery-Result Types					

10.3 ResultEvaluationEnum

This enumeration *ResultEvaluationEnum* indicates whether a result was in tolerance. The enumeration is defined in Table 24.

Table 24 – ResultEvaluationEnum Items

Name	Value	Description
Undefined	0	The evaluation of the result is unknown, for example because it failed
OK	1	The result is in tolerance
NotOK	2	The result is out of tolerance
NotDecidable	3	The decision is not possible due to measurement uncertainty.

Its representation in the *AddressSpace* is defined in Table 25.

Table 25 – ResultEvaluationEnum Definition

Attribute	Value				
BrowseName	ResultEvaluationEnum				
IsAbstract	False				
Description	Indicates whether a result was in tolerance				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	
Conformance Units					
Machinery-Result Types					

10.4 ProcessingTimesDataType

This structure contains measured times that were generated during the execution of a recipe. These measured values provide information about the duration required by the various sub-functions The structure is defined in Table 26.

Table 26 – ProcessingTimesDataType Structure

Name	Type	Description	Optional
ProcessingTimesDataType	structure		
StartTime	0:UtcTime	Contains the time when the system started execution of the recipe.	False
EndTime	0:UtcTime	Contains the time when the system finished (or stopped/aborted) execution of the recipe.	False
AcquisitionDuration	0:Duration	Time spent by the system acquiring data.	True
ProcessingDuration	0:Duration	Time spent by the system processing data.	True

Its representation in the *AddressSpace* is defined in Table 27.

Table 27 – ProcessingTimesDataType Definition

Attribute	Value				
BrowseName	ProcessingTimesDataType				
IsAbstract	False				
Description	Contains measured times that were generated during the execution of a recipe.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure defined in OPC 10000-5					
Conformance Units					
Machinery-Result Types					

10.5 ResultDataType

This structure contains fields that were created during the execution of a recipe. A result may be a total or a partial result, which is defined by the value of the property *IsPartial*. The structure of the *ResultContent* which is generated by the system is application-specific and not defined at this time. If no ResultContent is available, the array of ResultContent shall be empty.

The structure is defined in Table 28.

Table 28 – ResultDataType Structure

Name	Type	Description	Allow Subtypes
ResultDataType	structure		
ResultMetaData	ResultMetaDataType	Contains meta data describing the ResultContent	True
ResultContent	0:BaseDataType[]	Abstract data type to be subtyped from to hold result data created by the selected recipe.	False

Its representation in the *AddressSpace* is defined in Table 29.

Table 29 – ResultDataType Definition

Attribute	Value				
BrowseName	ResultDataType				
IsAbstract	False				
Description	Contains fields that were created during the execution of a recipe.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure defined in OPC 10000-5					
Conformance Units					
Machinery-Result Types					

10.6 ResultMetaDataType

This structure contains meta data of a result, describing the result. The structure is defined in Table 30.

Table 30 – ResultMetaDataType Structure

Name	Type	Description	Optional
ResultMetaDataType	structure		
ResultId	0:TrimmedString	System-wide unique identifier of the result, which is assigned by the system. This ID can be used for fetching exactly this result using the method GetResultById and it is identical to the ResultId of the ResultReadyEventType. If the system does not manage ResultIds, it should always be set to “NA”. If the system manages partial results, all partial results of a result shall use the same ResultId.	False
HasTransferableDataOnFile	0:Boolean	Indicates that additional data for this result can be retrieved by temporary file transfer. If not provided, it is assumed that no file is available.	True
IsPartial	0:Boolean	Indicates whether the result is the partial result of a total result. When not all samples are finished yet the result is 'partial'. If not provided, it is assumed to be a total result.	True
IsSimulated	0:Boolean	Indicates whether the result was created in simulation mode. Simulation mode implies that the result is only generated for testing purposes and not based on real production data. If not provided, it is assumed to not be simulated.	True
ResultState	0:Int32	ResultState provides information about the current state of the process or measurement creating a result. Applications may use negative values for application-specific states. All other values shall only be used as defined in the following: 0 – Undefined initial value 1 – Completed: Processing was carried out completely 2 – Processing: Processing has not been finished yet 3 – Aborted: Processing was stopped at some point before completion 4 – Failed: Processing failed in some way.	True
StepId	0:TrimmedString	Identifies the step which produced the result. Although the system-wide unique JobId would be sufficient to identify the job which the result belongs to, this makes for easier filtering without keeping track of JobIds. This specification does not define how the StepId is transmitted to the system. Typically, it is provided by the Client when starting an execution.	True
PartId	0:TrimmedString	Identifies the part used to produce the result. Although the system-wide unique JobId would be sufficient to identify the job which the result belongs to, this makes for easier filtering without keeping track of JobIds. This specification does not define how the PartId is transmitted to the system. Typically, it is provided by the Client when starting the job.	True

ExternalRecipeId	0:TrimmedString	External ID of the recipe in use which produced the result. The External ID is managed by the environment. This specification does not define how the ExternalRecipeId is transmitted to the system. Typically, it is provided by the Client.	True
InternalRecipeId	0:TrimmedString	Internal ID of the recipe in use which produced the result. This ID is system-wide unique and it is assigned by the system.	True
ProductId	0:TrimmedString	Identifies the product used to produce the result. This specification does not define how the productId is transmitted to the system. Typically, it is provided by the Client.	True
ExternalConfigurationId	0:TrimmedString	External ID of the *Configuration* in use while the result was produced. It is managed by the *Environment*. This specification does not define how the ExternalConfigurationId is transmitted to the system. Typically, it is provided by the Client.	True
InternalConfigurationId	0:TrimmedString	Internal ID of the *Configuration* in use while the result was produced. This ID is system-wide unique and it is assigned by the system.	True
JobId	0:TrimmedString	Identifies the job which produced the result. This ID is system-wide unique and it is assigned by the system.	True
CreationTime	0:UtcTime	CreationTime indicates the time when the result was created. Creation time on the measurement system (not the receive time of the Server). It is recommended to always provide the CreationTime.	True
ProcessingTimes	ProcessingTimesDataType	Collection of different processing times that were needed to create the result.	True
ResultUri	0:UriString[]	Path to the actual measured result, managed external to the Server.	True
ResultEvaluation	ResultEvaluationEnum	The ResultEvaluation indicates whether the result was in tolerance.	True
ResultEvaluationCode	0:Int64	Vendor-specific code describing more details on ResultEvaluation.	True
ResultEvaluationDetails	0:LocalizedText	The ResultEvaluationDetails provides high level status information in a user-friendly text. This can be left empty for successful operations.	True
FileFormat	0:String[]	The format in which the measurement results are available (e.g. QDAS, CSV, …) using the ResultTransfer Object. If multiple file formats are provided, the GenerateFileForRead of ResultTransfer should contain corresponding generateOptions, to select the file format. This specification does not define those generateOptions.	True

Its representation in the *AddressSpace* is defined in Table 31.

Table 31 – ResultMetaDataType Definition

Attribute	**Value**				
BrowseName	ResultMetaDataType				
IsAbstract	False				
Description	Meta data of a result, describing the result.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure defined in OPC 10000-5					
Conformance Units					
Machinery-Result Types					

11 Profiles and ConformanceUnits

11.1 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for Machinery – Result Transfer.

Table 32 – Conformance Units for Machinery – Result Transfer

Category	Title	Description
Server	Machinery-Result Types	Exposes the ResultManagementType, ResultTransferType, ResultType, Handle. TrimmedString, BaseResultTransferOptionsDataType, ResultTransferOptionsDataType, ResultEvaluationEnum, ProcessingTimesDataType, ResultDataType, and ResultMetaDataType and all their supertypes in the AddressSpace.
Server	Machinery-Result GetResultById	Supports at least one instance of the ResultManagementType supporting the GetResultById and the ReleaseResultHandle Methods.
Server	Machinery-Result GetResultsFiltered	Supports at least one instance of the ResultManagementType supporting the GetResultIdListFiltered, GetResultById and the ReleaseResultHandle Method.
Server	Machinery-Result GetLatestResult	Supports at least one instance of the ResultManagementType supporting the GetLatestResult Method.
Server	Machinery Result AcknowledgeResults	Supports at least one instance of the ResultManagementType supporting the AcknowledgeResults Method.
Server	Machinery-Result ResultVariables	Supports at least one instance of ResultManagementType supporting the Results folder having at least one ResultVariable.
Server	Machinery-Result ResultEvents	Exposes the ResultReadyEventType and all its supertypes in the AddressSpace. Supports at least one instance of ResultManagementType generating Events of ResultReadyEventType.
Server	Machinery-Result ResultFiles	Supports at least one instance of ResultManagementType supporting the ResultTransfer Object and providing access for at least some results using the file transfer.
Server	Machinery-Result PredefinedResultMetaData	For all results meta data exposed by the Server, either by events, methods or variables, the optional ExternalRecipeId, InternalRecipeId, JobId, ProductId, StepId and CreationTime shall be provided.
Client	Machinery-Result Client Simple Result Transfer	The Client is capable to receive results using the GetLatestResult method and can receive the additional result data of those results via the ResultTransfer Object.
Client	Machinery-Result Client Result Transfer	The Client is capable to receive results using any of the following mechanisms: GetResultById method, GetLatestResult, the Result Object with its Variables, the ResultReadyEvents, and if needed the ResultTransfer Object. Optionally it is capable to use the GetResultIdListFiltered method.
Client	Machinery Result Client AcknowledgeResults	The Client is capable to call the AcknowledgeResults method after receiving and processing results.

11.2 Profiles

11.2.1 Profile list

Table 33 lists all Profiles defined in this document and defines their URIs.

Table 33 – Profile URIs for Machinery – Result Transfer

Profile	URI
Machinery-Result Simple Result Transfer	http://opcfoundation.org/UA-Profile/Machinery/Result/Server/SimpleResultTransfer
Machinery-Result Result Transfer	http://opcfoundation.org/UA-Profile/Machinery/Result/Server/ResultTransfer
Machinery-Result Result Transfer Variables	http://opcfoundation.org/UA-Profile/Machinery/Result/Server/ResultTransferVariables
Machinery-Result Client Simple Result Transfer	http://opcfoundation.org/UA-Profile/Machinery/Result/Client/SimpleResultTransfer
Machinery-Result Client Result Transfer	http://opcfoundation.org/UA-Profile/Machinery/Result/Client/ResultTransfer

11.2.2 Server Facets

11.2.2.1 Overview

The following sections specify the *Facets* available for *Servers* that implement the Machinery – Result Transfer companion specification. Each section defines and describes a *Facet* or *Profile*.

11.2.2.2 Machinery-Result Simple Result Transfer Server Facet

Table 34 defines a *Facet* that describes a *Server* providing a simple result transfer by offering the *GetLatestResult Method*. This mechanism is limited to only provide the latest result, not accessing the previous results.

Table 34 – Machinery-Result Simple Result Transfer Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Machinery-Result	Machinery-Result GetLatestResult	M
Machinery-Result	Machinery-Result ResultFiles	O
Machinery-Result	Machinery-Result PredefinedResultMetaData	O
Machinery-Result	Machinery-Result Types	M
Address Space Model	0:Address Space Base	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Method Services	0:Method Call	M
Method Services	0:Method Call Complex	M

11.2.2.3 Machinery-Result Result Transfer Server Facet

Table 35 defines a *Facet* that describes a *Server* providing a result transfer mechanism by providing *ResultEvents* and the *GetResultById Method*. This mechanism can be used to allow accessing the latest and previous results.

Table 35 – Machinery-Result Result Transfer Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Machinery-Result	Machinery-Result GetResultById	M
Machinery-Result	Machinery-Result GetResultsFiltered	O
Machinery Result	Machinery Result AcknowledgeResults	O
Machinery-Result	Machinery-Result ResultEvents	M
Machinery-Result	Machinery-Result ResultFiles	O
Machinery-Result	Machinery-Result PredefinedResultMetaData	O
Machinery-Result	Machinery-Result Types	M
Address Space Model	0:Address Space Base	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Method Services	0:Method Call	M
Method Services	0:Method Call Complex	M
Profile	0:Standard Event Subscription Server Facet	M

11.2.2.4 Machinery-Result Result Transfer Variables Server Facet

Table 36 defines a *Facet* that describes a *Server* providing a result transfer mechanism by offering *ResultVariables*. This mechanism can be used to allow accessing the latest and previous results.

Table 36 – Machinery-Result Result Transfer Variables Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Machinery-Result	Machinery-Result ResultVariables	M
Machinery-Result	Machinery-Result ResultEvents	O
Machinery-Result	Machinery-Result ResultFiles	O
Machinery-Result	Machinery-Result PredefinedResultMetaData	O
Machinery-Result	Machinery-Result Types	M
Address Space Model	0:Address Space Base	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Method Services	0:Method Call	O
Method Services	0:Method Call Complex	O

11.2.3 Client Facets

11.2.3.1 Overview

The following tables specify the *Facets* available for *Clients* that implement the Machinery – Result Transfer companion specification.

11.2.3.2 Machinery-Result Client Simple Result Transfer Client Facet

Table 37 defines a *Facet* that describes the base characteristics for *Clients* using the *GetLatestResult Method* to receive results.

Table 37 – Machinery-Result Client Simple Result Transfer Client Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Machinery-Result	Machinery-Result Client Simple Result Transfer	M
Method Services	0:Method Client Call Complex	M
View Services	0:View Client Basic Browse	M
Address Space Model	0:Address Space Client Base	M

11.2.3.3 Machinery-Result Client Result Transfer Client Facet

Table 38 defines a *Facet* that describes the base characteristics for *Clients* supporting various ways to receive results.

Table 38 – Machinery-Result Client Result Transfer Client Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Machinery-Result	Machinery-Result Client Result Transfer	M
Machinery Result	Machinery Result Client AcknowledgeResults	O
Method Services	0:Method Client Call Complex	M
View Services	0:View Client Basic Browse	M
Address Space Model	0:Address Space Client Base	M
Profile	0:Event Subscriber Client Facet	
Profile	0:Attribute Read Client Facet	

12 Namespaces

12.1 Namespace Metadata

Table 39 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 39 – NamespaceMetadata Object for this Document

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/Machinery/Result/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/Machinery/Result/
NamespaceVersion	String	1.01.0
NamespacePublicationDate	DateTime	2025-07-01
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType []	0
StaticNumericNodeIdRange	NumericRange []	
StaticStringNodeIdPattern	String	

Note: The *IsNamespaceSubset Property* is set to False as the UANodeSet XML file contains the complete Namespace. *Servers* only exposing a subset of the Namespace need to change the value to True.

12.2 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 40 provides a list of mandatory and optional namespaces used in a Machinery – Result Transfer OPC UA *Server*.

Table 40 – Namespaces used in a Machinery – Result Transfer Server

NamespaceURI	Description
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.
Local Server URI	Namespace for nodes defined in the local *Server*. This namespace shall have namespace index 1.
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Machinery/Result/	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.
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.

Table 41 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 41 – Namespaces used in this document

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits

Annex A
(normative)

Machinery – Result Transfer Namespace and mappings

A.1 NodeSet and supplementary files for Machinery – Result Transfer Information Model

The Machinery – Result Transfer *Information Model* is identified by the following URI:

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

Documentation for the NamespaceUri can be found here.

The *NodeSet* associated with this version of specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/Result/&v=1.01.0&i=1

The *NodeSet* associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/Result/&i=1

Supplementary files for the Machinery – Result Transfer *Information Model* can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/Result/&v=1.01.0&i=2

The files associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/Result/&i=2

_____

Annex B
(informative)

Extending the Result Transfer

B.1 Overview

This specification defines concepts for result transfer that can be used as is. However, companion specifications or vendors may want to add additional information into the meta data of the result transfer or define explicitly the formats of the transferred results. This annex describes how to best extend the result transfer approach.

B.2 Extending Metadata

The metadata of the result is defined in the *ResultMetaDataType*, which is used in the *ResultDataType* and therefore also in the *ResultType VariableType*, which again is used in the *ResultReadyEventType*. The *ResultDataType* is also used in various methods.

If the metadata of the result is to be extended, the *ResultMetaDataType* has to be extended by creating a subtype. For using the *Methods* of the *ResultManagementType*, this is all that needs to be done. Those *Methods* returning instances of *ResultDataType*, now using the extended subtype in that structure. It is not allowed to change the signature of the *Methods* in a subtype of *ResultManagementType*, therefore, no subtype needs to be created.

In case, the *ResultType Variables* or the *ResultReadyEvents* are supported, additional subtypes should be created. It is not necessary to create a subtype of *ResultDataType*, as it is not possible to change the *DataType* of *ResultMetaData* in the *DataType*. Only instances of *ResultDataType* will use the subtype. But the *ResultType VariableType* is exposing the content of the *ResultMetaDataType* as subvariables. Therefore, a subtype of the *VariableType* should be created exposing those additional fields of the subtype of the *ResultMetaDataType* as subvariables. Since those should also be exposed as *Event* fields, a subtype of the *ResultReadyEventType* should be created using the subtype of the *VariableType*. In both cases, also a subtype of the *ResultManagementType* should be created, using the subtype of the VariableType and pointing to the subtype of the *ResultReadyEventType*.

In Table 42, this is summarized.

Table 42 – Required Subtypes for extending Result Metadata

Supported Concept	ResultMetaDataType	ResultDataType	ResultType	ResultReadyEventType	ResultManagementType
Result Methods	Yes	No	No	No	No
Result Variables	Yes	No	Yes	No	Yes
Result Events	Yes	No	Yes	Yes	Yes

In Figure 4, an example extending the metadata for all communication mechanisms is given.

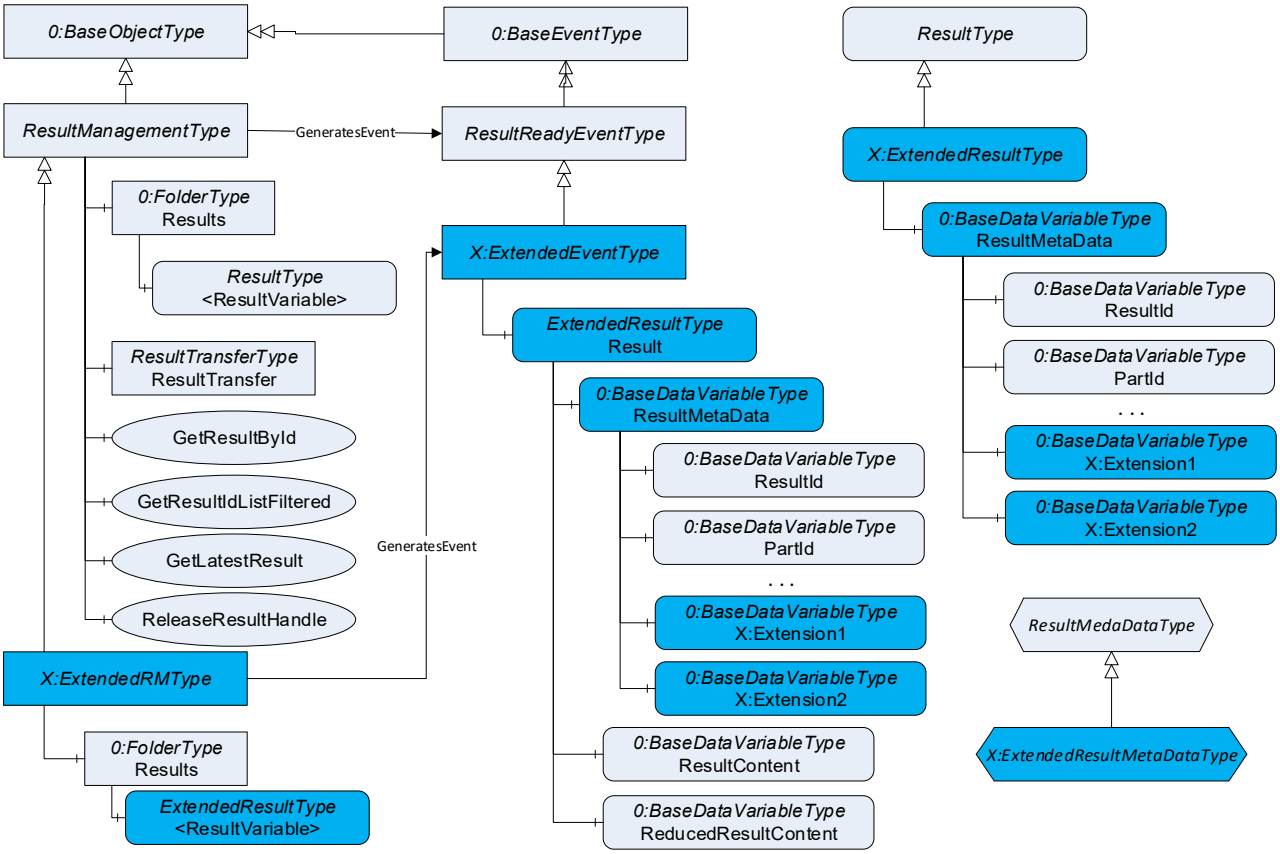

Figure 4 – Example of extending the metadata

B.3 Explicit Result Format

The result is returned in the *ResultContent* of the *ResultDataType*, as well as in the *ResultContent* and *ReducedResultContent* of the *ResultType VariableType*, and potentially also in the file transfer using the *ResultTransferType*.

B.3.1 Explicit Content Format

In order to expose the format and data structure of the *ResultContent*, it is not possible to subtype the *ResultDataType*, as the type of a structure field cannot be changed in a subtype. However, a subtype of the *ResultType VariableType* should be created, where the *ResultContent* and *ReducedResultContent* can be overridden and the *DataType* is changed to the specific *DataType*. In order to expose this for the *Event* subscriptions, a subtype of the *ResultReadyEventType* should be created as well, using the subtype of the *VariableType*.

That is, without a subtype of the *VariableType* there is no standardized mechanism to expose the expected *DataType* for *ResultContent* and *ReducedResultContent*. *Clients* need to receive an instance of *ResultDataType* in order to get the information about the used *DataType* in that instance.

B.3.2 Explicit File Format

If the file transfer is used to expose larger results, *Servers* may support various file formats, as exposed by the *FileFormat* field of the *ResultMetaDataType*. In order to allow access to those different file formats, a *Server* should subtype the *BaseResultTransferOptionsDataType* and add a field for selecting the file format (and potentially other fields defining the content of the file). *Servers* should subtype the *ResultTransferType* (and therefore also the *ResultManagementType*) to document which subtype of the *ResultTransferOptionsDataType* is allowed when calling the *Method* for the file transfer. The *BaseResultTransferOptionsDataType* is intentionally defined as an abstract *DataType*. The subtype *ResultTransferOptionsDataType* can be used as concrete *DataType*, if no additional fields are needed. This allows to override the *GenerateFileForRead* Method and change the *DataType* of the generateOptions in the signature to a concrete *DataType*. This would not be allowed, if the *BaseResultTransferOptionsDataType* would not be abstract.