FUJITSU

FUJITSU Software
Interstage AR Processing Server V1.1.1

Reference Guide

B1WS-1108-02ENZ0(00)
December 2014

Preface

Purpose of this document

This document is a reference guide to the JavaScript library, web API and commands provided by Interstage AR Processing Server.

Intended readers

This document is intended for system developers who develop business applications for Interstage AR Processing Server. Readers of this document are also assumed to have basic knowledge of:

- Interstage AR Processing Server Developer's Guide
- Basic knowledge of JavaScript
- Basic knowledge of databases
- Basic knowledge of file management

Structure of this document

This document is structured as follows:

Chapter 1 JavaScript libraries reference

Provides information about the JavaScript libraries provided by Interstage AR Processing Server.

Chapter 2 WebAPI (REST) reference

Provides information about the WebAPI provided by Interstage AR Processing Server.

Chapter 3 Command reference

Provides information about the commands provided by Interstage AR Processing Server.

Abbreviations

This manual uses the following abbreviations for the operating systems:

Official name	Abbreviation	
Microsoft Windows Server 2012 R2 Foundation	Windows Server 2012 R2	Windows
Microsoft Windows Server 2012 R2 Standard		
Microsoft Windows Server 2012 R2 Datacenter		
Microsoft Windows Server 2012 Foundation	Windows Server 2012	
Microsoft Windows Server 2012 Standard		
Microsoft Windows Server 2012 Datacenter		
Microsoft Windows Server 2008 R2 Standard	Windows Server 2008 R2	
Microsoft Windows Server 2008 R2 Enterprise		
Microsoft Windows Server 2008 R2 Datacenter		
Microsoft Windows Server 2008 R2 Foundation		
Windows 8.1	Windows 8.1	
Windows 8.1 Pro		
Windows 8.1 Enterprise		
Windows 8	Windows 8	
Windows 8 Pro		
Windows 8 Enterprise		
Red Hat Enterprise Linux 6 (for Intel64)	RHEL 6	Linux

Official name	Abbreviation	
Red Hat Enterprise Linux 7 (for Intel64)	RHEL 7	

Notations

In this manual, text that must be replaced by the user is denoted in *italicsWithMixedCase* (for example, *installDir*).

Trademarks

- Access, Excel, PowerPoint and Word are products of Microsoft Corporation in the United States.
- Adobe, Acrobat, Adobe Reader, Acrobat Reader, the Adobe logo, Adobe AIR, Flash and Flash Player are registered trademarks or trademarks of Adobe Systems Incorporated in the United States and other countries.
- Android is a registered trademark of Google Inc.
- Eclipse is an open platform for the development tool integration constructed by Eclipse Foundation, Inc. that is an open community of the development tool provider.
- Internet Explorer, the Microsoft Internet Explorer logo, Microsoft, Windows, Windows Server, Visual Studio and other names and product names of Microsoft products are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Interstage is a registered trademark of Fujitsu Limited.
- iOS is a trademark of Apple Inc.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other company names and/or product names appearing in this document may also be trademarks or registered trademarks of their respective companies.
- QuickTime and the QuickTime logo are trademarks of Apple Inc., registered in the United States and other countries.
- Xeon and Xeon Inside are trademarks of Intel Corporation in the United States and other countries.
- Other company names and product names used in this document are trademarks or registered trademarks of their respective owners.

Note that system names and product names in this document are not accompanied by trademark symbols such as (TM) or (R).

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Notice

- Information in this document may be subject to change without prior notice.
- No part of the contents of this document may be reproduced without the written permission of Fujitsu Limited.
- Fujitsu assumes no responsibility for infringement of any patent rights or other rights of third parties arising from use of information in the manual.

Issue date

December 2014

Copyright

Contents

Chapter 1 JavaScript libraries reference

The documentation for the JavaScript libraries bundled with Interstage AR Processing Server is provided in HTML format, and can be accessed via browser in the address below.

W *dvdMountPoint*\manual\jslib\index.html

L *dvdMountPoint*/manual/jslib/index.html

Note

- In Internet Explorer 10 and earlier, the library may not be displayed properly. To solve this issue, on the menu bar, click **Tools** > **F12 developer tools**, and disable the browser mode compatibility view.
- In Internet Explorer 10, examples may be displayed without line feeds. To solve this issue, on the menu bar, click **Tools** > **F12 developer tools**, and change the document mode to **IE9 standards**.

Chapter 2 WebAPI (REST) reference

This chapter describes the WebAPIs provided by Interstage AR Processing Server.

2.1 Data management WebAPI

This section describes the data management WebAPI provided by Interstage AR Processing Server. Data management involves tasks such as registering, editing and deleting information related to AR markers, scenarios, scenes and AR overlay definitions.

2.1.1 Data management overview

In Interstage AR Processing Server, it is necessary to collect and retrieve information such as AR scenarios, scenes, overlay definitions, and job information linked to the AR markers captured by the camera, as well as the permissions given to users operating the camera. However, it is difficult to collect and manage this information in a single structure, because each type of information requires a different data structure.

In Interstage AR Processing Server, all information is expressed using the value sets below. This enables the user to collect and manage information with different data structures under a single structure, and cross-search all management information.

- QType - Defines the information, and is analogous to a relational database "table".
- QAttribute - Defines attributes for the information, and is analogous to a relational database "column".
- QEntity - Defines entities for the information, and is analogous to a relational database "row".
- QValue - Defines values for information, attributes, and entities, and is analogous to a relational database "column value".

Interstage AR Processing Server provides a data management feature as a framework for managing, searching, and editing the kinds of data mentioned in this section.

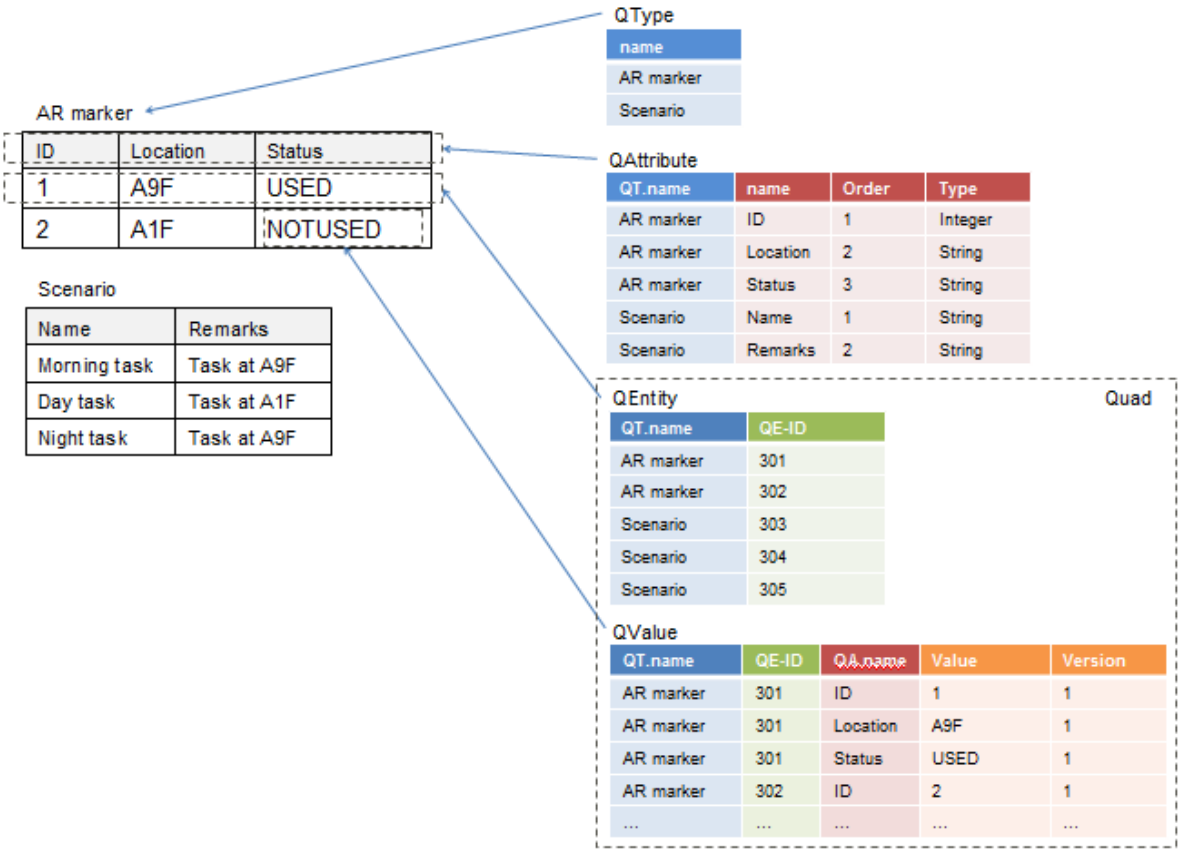
 See

Refer to the *Developer's Guide* for details on the data structures.

2.1.1.1 Data management diagram

This section describes the concept of data management in Interstage AR Processing Server.

Figure 2.1 Data management diagram



- QType
  Stores, as values, what is referred to in relational databases as a "table".
- QAttribute
  Stores, as values, what is referred to in relational databases as a "column". Its parent is QType, which is analogous to a "table".
- QEntity
  Stores, as values, what is referred to in relational databases as a "row". Its parent is QType, which is analogous to a "table".
- QValue
  Stores, as values, what is referred to in relational databases as a "column value". Its parents are QAttribute, which is analogous to a "column", and QEntity, which is analogous to a "row".
- Quad
  A logical table that extends QEntity. It can handle both QEntity and QValue arrays (the latter have QEntity as their parent).
  - QEntity including QValue arrays can be registered by a single request.
  - Quad can perform a cross-search of QType ("table"), QAttribute ("column"), and QValue ("column value") using a conditional expression that is string-based rather than ID-based.

  Example

  Records that have "A9F" for Location in the AR marker table and "Night task" for Name in the Scenario table are returned in the order of when State became "Used".

2.1.1.2 Data exclusion control

Interstage AR Processing Server performs optimistic concurrency control on data to handle multiple access requests simultaneously for the same data. For this reason, version (INTEGER type) is created for each table.

The basic flow of control is as follows:

1. When changing data, version information is acquired for determining whether the data has been changed by another user after acquiring the data.
2. The acquired version information is compared with the current version.
    1. If the version has not changed, the data is updated.
    2. If the version has changed, the change request is deemed invalid and an error is returned.

Interstage AR Processing Server uses ETag headers and If-Match headers to achieve optimistic concurrency control. It is therefore necessary to get the latest data from the server and update the version prior to updating/deleting the data. Furthermore, it is necessary to set the version in the If-Match header.

Because the version number is incremented each time the data is changed, an error occurs when the data is updated after INTEGER reaches its maximum value (2147483674).

2.1.2 API types

The following common APIs are individually provided in table units:

API type	Method	Provided for	URL, header/body of request/response			Example
Identify via primary key	GET	QEntity QValue	URL		http://.../(*resourceName*)/(*primaryKey*)	http://.../qtypes/Facility%20Management
			Request	Header	None	None
				Body	None	None
			Response	Header	ETag: "*version*"	ETag: "4"
				Body	JSON-formatted result resource	{name:"Facility management", version:4}
Search	GET	QType QAttribute QEntity QValue	URL		http://.../*resourceName*?key=*jsonFormattedValue*&...	http://.../qtypes?type=RECORDS&limitRange={start:1,end:10}&...
			Request	Header	None	None
				Body	None	None
			Response	Header	None	None

				Body	JSON-formatted search result object:
					class result {
					long unlimitedRecordCount;
					List<*tableResource*> records;
					}
						{unlimitedRecordCount:210,
						records:[
						{name:"Facility management", version:4},
						{name:"Parts management", version:8}]}

Register	POST	QEntity QValue	URL		http://.../*resourceName*	http://.../qtypes
			Request	Header	None	None
				Body	JSON-formatted resource	{name:"Facility management", version:-1}
			Response	Header	ETag: "*version*"	ETag: "1"
				Body	JSON-formatted registered resource	{name:"Facility management", version:1}
Delete	DELETE	QType QAttribute	URL		http://.../*resourceName*/*primaryKey*	http://.../qtypes/Facility%20management

API type	Method	Provided for	URL, header/body of request/response			Example
		QEntity QValue	Request	Header	If-Match: "*version*"	If-Match: "5"
				Body	None	None
			Response	Header	None	None
				Body	None	None

Point

- A "none" header means that data management WebAPI does not actively add a header. There are standard HTTP headers and [Content-Type:application/json][Accept:application/json], which indicate requests and responses in JSON format.
- The above response is used if the state is normal.

2.1.2.1 HTTP status codes returned by APIs and the return of exception objects

The HTTP status codes returned by APIs are determined by API type, as follows:

Code	Meaning (W3C)	Meaning (Product)	Identify ID (GET)	Search (GET)	Registration (POST)	Delete (DEL)
200	OK	Processed successfully	Y	Y		Y
201	Created	Created successfully			Y	
400	Bad Request	Invalid request	Y	Y	Y	Y
404	Not Found	Resource not found	Y			Y
412	Precondition Failed	Denied by optimistic concurrency control				Y
500	Internal Server Error	Other error	Y	Y	Y	Y

Point

- The above status codes are returned by Interstage AR Processing Server. There are cases where codes such as the following are returned by the web server/Java EE runtime environment, not by Interstage AR Processing Server:
  - Normal HTTP status codes such as "401 Unauthorized", etc.
  - 4xx and 5xx error status codes that indicate an invalid HTTP message format/path format/JSON format/value type.
- Exception information such as the following is returned in the response body for non-2xx status codes (refer to *Messages* for details).

Description	Example

JSON-formatted exception object.
 	
class exception {
String message;
}
 	
The message is the same as the output for the component name and subsequent content in message output format.
	{"message":"QDWEB: ERROR: 13201: QType does not exist."}

2.1.3 Notes

Note the following when using the data management WebAPI.

2.1.3.1 Constraints arising from web server/Java EE runtime environment specifications

The flow for processing HTTP requests is as follows: Web server to Java EE runtime environment to data management feature. Therefore, responses are returned by a front-end Web server or Java EE runtime environment if requests do not reach the data management feature, which may result in the following issues:

- When the HTTP message format, path format, JSON format or value type is deemed invalid, another component on the front end responds with error status code 400 or 500, and the response from Interstage AR Processing Server may not be returned.
- The HTTP status code may change due to features (such as HTTP server) placed between the smart device and Java EE runtime environment.
- If a value exceeding the maximum value for a parameter of a numeric type is specified, it may be converted into a negative value or null.
- The exception [{WHAT} must be not null."] may be returned if a non-null invalid value is referenced. In this situation, the value is not only null but the format may also be invalid, therefore you must address this as necessary.

2.1.3.2 Data send/receive constraints placed on APIs

Passing an extremely large definition to Interstage AR Processing Server may degrade server performance. Therefore, set a maximum value for the request body of APIs, not only to prevent attacks by a third party with malicious intent, but also to prevent misuse by authorized users.

2.1.3.3 Character-related specifications/constraints

- The character set is Unicode, and the encoding for requests/responses is UTF-8.

URL encoding

Although the example URLs used in this document are in plain text for ease of understanding, the actual URLs should originally be encoded in UTF-8.

- The maximum length of strings is calculated using UTF-16 (The characters that are not on plane 0 are expressed using surrogate pairs).
- String values cannot contain 0x00. Also, if character sequence errors (incorrect arrangement order of surrogate pairs, etc.) or non-characters (U+FFFE, U+FFFF, etc.) are specified, they may be dropped (become 0x3F, etc.) on the distribution path or data store.
- The sort order is Unicode code point order.

2.1.3.4 Record limitations

Interstage AR Processing Server has the following kinds of record limitations in place for each table.

Table name	Maximum number of records
QType	104
QAttribute	3,100
QEntity	1,100,000
QValue	31,000,000

2.1.3.5 Interstage AR Processing Server long data type

The Interstage AR Processing Server long type is equivalent to Java long. Interstage AR processing server may not be able to handle all long values, though, depending on the programming language used, and the application runtime environment.

Example

The long type value 12345678901234567890 stored in Interstage AR Processing Server is rounded to 12345678901234567000 in JavaScript.

Below are cases of long values used in Interstage AR Processing Server - address these on the client side:

- Qvalue
  If inserting long values such as the following, take into consideration the types of clients that will be used:
  - If QValue is Long
  - If searching for long Qvalues (specifying qvalueRanges and longValueRanges)
- IDs automatically assigned by Interstage AR Processing Server
  The IDs below are created in epoch time (microsecond units) - values of 100 years or more can be successfully handled as a number type using JavaScript.
  - QEntity or QValue, file management IDs
  - Searches using an ID
- Applicable record count for search API runtime (unlimitedRecordCount)
  Address this on the client side to handle record counts registered in Interstage AR Processing Server.

2.1.4 QType

2.1.4.1 Data structure

Attribute	Description	Value type	Constraints
name	Name of the data. This is analogous to a relational database table name.	String	Required, unique, between 1 and 30 characters
description	Description	String	Between 0 and 300 characters
version	Version for optimistic concurrency control. Assigned by the server.	Integer	Required

2.1.4.2 Identity via primary key API

Method/URL

```
GET http://server:port/arsvdm/qtypes/name
```

Request

Path parameter	Description
name	name

Response

Header/body	Description
ETag header	Version information
Body	JSON-formatted QType data

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

Request example

```
http://server:port/arsvdm/qtypes/computer001
```

2.1.4.3 Search API

Method/URL

```
GET http://server:port/arsvdm/qtypes?queryParameter...
```

Request

Query parameter	Description	Value type	Constraints, etc.
type	Type of data returned as a processing result - can have the following values: - RECORDS - Returns result array - COUNT - Returns number of results - RECORDANDCOUNT - Returns result array and total number of records	Enum	Default=RECORDS
limitRange	Range of records in the search result to return. The interval start must be greater than 0, the interval end must be greater than or equal to the interval start, and the interval cannot include more than 100 items. Example: {"start":1,"end":10}	Range<Integer>	Default=1 to 10, interval less than 100, cannot reverse the start/end values
nameRanges	name range array. Example: [{"start":"computer001","end":"computer999"}]	List<Range<String>>	Up to 100 characters, up to 5 arrays
sortOrders	Sort order array - can have the following values: - NAME - Returns name in ascending order - NAME_DESC - Returns name in descending order Example: ["NAME"]	List<Enum>	Default=none (undefined), up to 5 arrays

Response

Header/body	Description
Body	JSON-formatted search result object. Format for records is List<QType>.

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

Request example

```
http://server:port/arsvdm/qtypes?type=RECORDSANDCOUNT&limitRange={"start":1,"end":10}&
nameRanges=[{"start":"computer001","end":"computer999"}]&sortOrders=["NAME"]
```

2.1.5 QAttribute

2.1.5.1 Data structure

Attribute	Description	Value type	Constraints
qtypeName	Parent QType.name	String	Required

Attribute	Description	Value type	Constraints
name	Name of the data. This is analogous to a relational database column name.	String	Required, unique for same qtypeName, between 1 and 30 characters
description	Same description as QType	See left	See left
orderIndex	This is analogous to a relational database column sequence.	Integer	
qvalueType	This is analogous to a relational database column data type - can have the following values: STRING LONG FLOAT	Enum	Required, default=STRING
version	Same description as QType	See left	See left

2.1.5.2 Identity via primary key API

Method/URL

```
GET http://server:port/arsvdm/qattributes/qtypename/name
```

Request

Path parameter	Description
qtypename	qtypeName
name	name

Response

Header/body	Description
ETag header	Same description as QType
Body	JSON-formatted QAttribute data

Example

Request example

```
http://server:port/arsvdm/qattributes/computer001/Remark01
```

2.1.5.3 Search API

Method/URL

```
GET http://server:port/arsvdm/qattributes?queryParameter...
```

Request

Query parameter	Description	Value type	Constraints, etc.
type	Same description as QType	See left	See left
limitRange	Same description as QType	See left	See left
qtypeNameRanges	Same description as nameRanges of QType	See left	See left
nameRanges	name range array Example: [{"start":"Remark01","end":"Remark99"}]	List<Range<String>>	Up to 100 characters, up to 5 arrays

Query parameter	Description	Value type	Constraints, etc.
sortOrders	Sort order array - can have the following values: - QTYPENAME - Returns qtypeName in ascending order - QTYPENAME_DESC - Returns qtypeName in descending order - NAME - Returns name in ascending order NAME_DESC - Returns name in descending order - ORDERINDEX - Returns orderIndex in ascending order - ORDERINDEX_DESC - Returns orderIndex in descending order Example: ["ORDERINDEX","QTYPENAME","NAME"]	List<Enum>	Default=none (undefined), up to 5 arrays

Response

Header/body	Description
Body	Same description as QType. Format for records is List<QAttribute>.

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

Request example

```
http://server:port/arsvdm/qattributes?type=RECORDSANDCOUNT&limitRange={"start":1,"end":
10}&qtypeNameRanges=[{"start":"computer001","end":"computer999"}]&nameRanges=[{"start":"Remark01","e
nd":"Remark99"}]&sortOrders=["ORDERINDEX","QTYPENAME","NAME"]
```

2.1.6 QEntity

2.1.6.1 Data structure

Attribute	Description	Value type	Constraints
qtypeName	Same description as QAttribute	See left	See left
id	Data ID. Assigned by the server.	Long	Required, unique for same qtypeName
version	Same description as QType	See left	See left

2.1.6.2 Identity via primary key API

Method/URL

```
GET http://server:port/arsvdm/qentities/qtypename/id
```

Request

Path parameter	Description
qtypename	Same as QAttribute
id	ID

Response

Header/body	Description
ETag header	Same description as QType
Body	JSON-formatted QEntity data

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

```
http://server:port/arsvdm/qentities/computer001/922337203685477580
```

2.1.6.3 Search API

Method/URL

```
GET http://server:port/arsvdm/qentities?queryParameter...
```

Request

Query parameter	Description	Value type	Constraints, etc.
type	Same description as QType	See left	See left
limitRange	Same description as QType	See left	See left
qtypeNameRanges	Same description as QAttribute	See left	See left
idRanges	id range array Example:[{"start":100,"end":200}]	List<Range<Long>>	Up to 5 arrays
sortOrders	Sort order array - can have the following values: - QTYPENAME - Returns qtypeName in ascending order - QTYPENAME_DESC - Returns qtypeName in descending order - ID - Returns ID in ascending order - ID_DESC - Returns ID in descending order Example: ["QTYPENAME","ID"]	List<Enum>	Default=none (undefined), up to 5 arrays

Response

Header/body	Description
Body	Same description as QType. Format for records is List<QEntity>.

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

Request example

```
http://server:port/arsvdm/qentities?type=RECORDSANDCOUNT&limitRange={"start":1,"end":
10}&qtypeNameRanges=[{"start":"computer001","end":"computer999"}]&idRanges=[{"start":100,"end":200},
{"start":300,"end":400}]&sortOrders=["QTYPENAME","ID"]
```

2.1.6.4 Register API

Method/URL

```
POST http://server:port/arsvdm/qentities
```

Request

Header/body	Description
Body	JSON-formatted QEntity data

It is not necessary to set a value for ID/version. The server automatically assigns a new value when registration is performed.

Response

Header/body	Description
ETag header	Same description as QType
Body	JSON-formatted registered QEntity data

When performing client authoring concurrently, the maximum number of QEntity values that can be added to arpoiarmk_default per scenario is 100.

2.1.6.5 Delete API

Method/URL

```
DELETE http://server:port/arsvdm/qentities/qtypename/id
```

Request

Header/path/body	Description
If-Match header	Same description as QType
qtypename	Same description as QAttribute
id	ID

Response

Header/body	Description
Body	Same description as QType

When a QEntity is deleted, all QValues whose parent is QEntity are deleted.

2.1.7 QValue

2.1.7.1 Data structure

Attribute	Description	Value type	Constraints
qtypeName	Same description as QAttribute	See left	Required, unique for same qentityId/qattributeName

Attribute	Description	Value type	Constraints
qentityId	The parent QEntity.id	Long	Required, unique for same qtypeName/qattributeName
qattributeName	The parent QAttribute.name	String	Required, unique for same qtypeName/qentityId
stringValue	Value used if the parent QAttribute.qvalueType=STRING	String	Not null under the condition on the left. Between 0 and 1000 characters
longValue	Value used when the parent QAttribute.qvalueType=LONG	Long	Not null under the condition on the left.
floatValue	Value used when the parent QAttribute.qvalueType=FLOAT	Float	Not null under the condition on the left.
version	Same description as QType	See left	See left

When expressing a null value, it should not be registered as QValue.xxxValue=null, but instead the QValue itself should not be registered at all. That is, it is necessary to express this as if there is a QAttribute but no QValue.

2.1.7.2 Identity via primary key API

Method/URL

```
GET http://server:port/arsvdm/qvalues/qtypename/qentityid/qattributename
```

Request

Path parameter	Description
qtypename	Same description as QAttribute
qentityid	ID of qentity
qattributename	Name of qattribute

Response

Header/body	Description
ETag header	Same description as QType
Body	JSON-formatted QValue data

The above response is used if the state is normal. If an error occurs, exception information is output.

```
http://server:port/arsvdm/qvalues/Facility001/9223372036854775807/Remark01
```

2.1.7.3 Search API

Method/URL

```
GET http://server:port/arsvdm/qvalues?parameter...
```

Request

Query parameter	Description	Value type	Constraints, etc.
type	Same description as QType	See left	See left
limitRange	Same description as QType	See left	See left
qtypeNameRanges	Same description as QAttribute	See left	See left
qentityIdRanges	Same description as idRanges of QEntity	See left	See left
qattributeNameRanges	Same description as nameRanges of QAttribute	See left	See left
stringValueRanges	stringValue range array Example: [{"start":"KanagawaPref_a","end":"KanagawaPref_z"}]	List<Range<String>>	Up to 100 characters, up to 5 arrays
longValueRanges	longValue range array Example: [{"start":1,"end":100}]	List<Range<Long>>	Up to 5 arrays
floatValueRanges	floatValue range array Example: [{"start":0.1,"end":0.9}]	List<Range<Float>>	Up to 5 arrays
sortOrders	- QTYPENAME - Returns qtypeName in ascending order - QTYPENAME_DESC - Returns qtypeName in descending order - QENTITYID - Returns qentityId in ascending order - QENTITYID_DESC - Returns qentityId in descending order - QATTRIBUTENAME - Returns qattributeName in ascending order - QATTRIBUTENAME_DESC - Returns qattributeName in descending order Example: ["QTYPENAME","QENTITYID","QATTRIBUTENAME"]	List<Enum>	Default=none (undefined), up to 5 arrays

Response

Header/body	Description
Body	Same description as QType. Format for records is List<QValue>.

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

```
http://server:port/arsvdm/qvalues?type=RECORDSANDCOUNT&limitRange={"start":1,"end":
10}&qtypeNameRanges=[{"start":"conputer001","end":"computer999"}]&qattributeNameRanges=[{"start":"Re
mark01","end":"Remark99"}]&qentityIdRanges=[{"start":100,"end":200},{"start":300,"end":
400}]&stringValueRanges=[{"start":"KanagawaPref_a","end":"KanagawaPref_z"}]&sortOrders=["QTYPENAME",
"QENTITYID","QATTRIBUTENAME"]
```

2.1.7.4 Register API

Method/URL

```
POST http://server:port/arsvdm/qvalues
```

Request

Header/body	Description
Body	JSON-formatted QValue data

It is not necessary to set a value for version. The server automatically assigns a new value when registration is performed.

Response

Header/body	Description
ETag header	Same description as QType
Body	JSON-formatted registered QValue data

2.1.7.5 Delete API

Method/URL

```
DELETE http://server:port/arsvdm/qvalues/qtypename/qentityid/qattributename
```

Request

Header/path/body	Description
If-Match header	Same description as QEntity
qtypename	Same description as QEntity
qentityid	ID of qentity
qattributename	Name of qattribute

Response

Header/body	Description
Body	Same description as QType

2.1.8 Quad

2.1.8.1 Data structure

Attribute	Description	Value type	Constraints
qtypeName	Same description as QEntity.qtypeName	See left	See left
id	Same description as QEntity.id	See left	See left
qvalues	QValue arrays which have a QEntity as parent	List<Quad.QValue>	Up to 100 arrays
version	Same description as QEntity.version	See left	See left

2.1.8.2 Search API

Method/URL

```
GET http://server:port/arsvdm/quads?parameter...
```

Request

Query parameter	Description	Value type	Constraints, etc.
type	Same description as QType	See left	See left
limitRange	Same description as QType	See left	See left
qattributeOrderIndexRange	Range for retrieving the result array in QAttribute direction. Specify in the sequence for sorting QAttribute.orderIndex in ascending order. Must be greater than 0. Example: {"start":1,"end":10}	Range<Integer>	Default=1 to 10, interval less than 100, cannot reverse the start/end values
qtypeNameRanges	Same description as QEntity	See left	See left
whereExpressions	Filter condition array	List<Expression>	Up to 5 arrays
sortOrderExpressions	Sort condition array	List<Expression>	Up to 5 arrays

Expression structure

Attribute	Description	Value type	Constraints, etc.
qattributeNameRanges	Same description as QValue Example: [{"start":"column01","end":"column09"}] Note: If a range such as [{"start":"column01","end":"column99"}] is specified as the sort condition, any value from column 01 to column 99 is used for sorting, and it is unpredictable which column value will be used. Specify a single value in the sort condition, and do not allow a range or range array to be specified.	See left	Required (the whole expression is ignored if it does not contain any valid values), up to 100 characters, up to 1 array for sort condition, same values for start and end
qvalueType	QAttribute.qvalueType identified by QValue.qtypeName, .qentityId, and .qattributeName	See left	Required
qvalueRanges	Range of Quad.QValue.xxxValue	Conforms with Quad.QValue.xxxValue	Up to 100 characters (in the case of stringValue), up to 5 arrays
desc	Whether to sort in descending order. Effective only when there is a sort condition.	boolean	

Response

Header/body	Description
Body	Same description as QType. Format for records is List<Quad>.

The above response is used if the state is normal. If an error occurs, exception information is output.

Example

```
http://server:port/arsvdm/quads?type=RECORDSANDCOUNT&limitRange={"end":10,"start":
1}&qattributeOrderIndexRange={"end":10,"start":1}&whereExpressions=[{"qattributeNameRanges":
[{"end":"age99","start":"age01"}],"qvalueType":"LONG","qvalueRanges":[{"end":69,"start":
```

```
10}]}]&sortOrderExpressions=[{"desc":false,"qattributeNameRanges":
[{"end":"gender","start":"gender"}],"qvalueType":"STRING","qvalueRanges":
[{"end":"male","start":"male"}]}]
```

Point

Since Quads only exist logically, building QEntity and QValue arrays inside Quad is only to show it as a single entity. For example, note that version and ETag are not managed in Quad units, but only in QEntity.

2.1.8.3 Register API

Method/URL

```
POST http://server:port/arsvdm/quads
```

Request

Header/body	Description
Body	JSON-formatted Quad data

Do not set values in ID/version for Quad, nor in qentityId/version for a QValue array. New numbers are assigned and IDs are identified by the server when registration is performed.

Response

Header/body	Description
ETag header	Same description as QType
Body	JSON-formatted registered Quad data

Etag is synonymous with the ETag for QEntity, included in the Quad data. It is not the ETag for QValue, included in the Quad data.

2.2 File data management WebAPI

This section describes the file data management WebAPI provided by Interstage AR Processing Server.

2.2.1 Outline

A file data management WebAPI is provided by Interstage AR Processing Server for registering or searching for data from a PC or smart device via the web.

The file data management WebAPI handles the following types of data:

File data

Data that needs to be saved, such as large text data, images, video, etc., is stored in the file system as file data. Files are saved sequentially under the file management root directory.

Management IDs assigned by the server during registration are used for file names when saving files. The base directory is specified in the settings file.

File properties

The following attribute information is stored as properties in the database. The server acquires property information such as registration time.

- Management ID
- File name
- File mimetype
- Registration time

- Latest update time
- File size

2.2.1.1 Data exclusion control

Interstage AR Processing Server performs optimistic concurrency control on data to handle multiple access requests simultaneously for the same data. For this reason, version (INTEGER type) is created for each table.

The basic flow of control is as follows:

1. When changing data, version information is acquired for determining whether the data has been changed by another user after acquiring the data.
2. The acquired version information is compared with the current version.
    1. If the version has not changed, the data is updated.
    2. If the version has changed, the change request is deemed invalid and an error is returned.

Interstage AR Processing Server uses ETag headers and If-Match headers to achieve optimistic concurrency control. It is therefore necessary to get the latest data from the server and update the version prior to updating/deleting the data. Furthermore, it is necessary to set the version in the If-Match header.

Note

Because a new version is added each time the data is changed, an error occurs when the data is updated after INTEGER reaches its maximum value (2147483647).

2.2.1.2 HTTP status codes returned by APIs and the return of exception objects

The HTTP status codes returned by APIs are determined by API type, as follows. A message is inserted in the response body when returning non-2xx status codes.

Code	Meaning (W3C)	Meaning (Product)	Acquire (GET)	Registration (POST)	Delete (DEL)
200	OK	Processed successfully	Y		Y
201	Created	Created successfully		Y	
400	Bad Request	Invalid request	Y	Y	Y
404	Not Found	Resource not found	Y		Y
412	Precondition Failed	Denied by optimistic concurrency control			Y
500	Internal Server Error	Other error	Y	Y	Y

Point

Standard HTTP status codes such as "401 Unauthorized" are returned by the web server, not by Interstage AR Processing Server.

2.2.2 Notes

2.2.2.1 File management limitations

This section describes the file management specifications for Interstage AR Processing Server.

File name

File names of up to 100 characters, including the extension, can be registered.

File size

The recommended maximum file size for registration is up to 5 MB.

mimetype

The available file mimetypes for registration are as follows:

MIME type	Extension
application/pdf	.pdf
application/msword	.doc
application/vnd.openxmlformats-officedocument.wordprocessingml.document	.docx
application/vnd.ms-excel	.xls
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	.xlsx
application/vnd.ms-powerpoint	.ppt
application/vnd.openxmlformats-officedocument.presentationml.presentation	.pptx
application/zip	.zip
image/jpeg	.jpg .jpeg
image/png	.png
image/gif	.gif
video/3gpp	.3gp
video/mp4	.mp4
video/mpeg	.mpg .mpeg
video/quicktime	.mov
video/x-ms-wmv	.wmv

2.2.2.2 Note about using Internet Explorer (IE) to access files

If using IE to access files, there are cases in which the file type is determined from the content, and the mimetype specified by the server may be ignored when displaying the file. Also, in the case of files containing HTML or JavaScript code, there may be risks present from XSS, etc., due to the file being run regardless of the mimetype.

2.2.2.3 Constraints arising from web server/Java EE runtime environment specifications

The flow for processing HTTP requests is as follows: Web server to Java EE runtime environment to data management feature. Therefore, responses are returned by a front-end Web server or Java EE runtime environment if requests do not reach the data management feature, which may result in the following issues:

- When the HTTP message format, path format, JSON format or value type is deemed invalid, another component on the front end responds with error status code 400 or 500, and the response from Interstage AR Processing Server may not be returned.
- The HTTP status code may change due to features (HTTP server) placed between the smart device and Java EE runtime environment.
- If a value exceeding the maximum value for a parameter of a numeric type is specified, it may be converted into a negative value or null.

2.2.2.4 Data send/receive constraints placed on APIs

Passing an extremely large definition to Interstage AR Processing Server can run the risk of degraded server performance. Therefore, set a maximum value for the request body of APIs not only to prevent attacks by a third party with malicious intent, but also to prevent misuse by authorized users.

2.2.2.5 Character-related specifications/constraints

- The character set is Unicode, and the encoding for requests/responses is UTF-8.

- The maximum length of strings is calculated using UTF-16 (The characters that are not on plane 0 are expressed using surrogate pairs).
- String values cannot contain 0x00. Also, if character sequence errors (incorrect arrangement order of surrogate pairs, etc.) or non-characters (U+FFFE, U+FFFF, etc.) are specified, they may be dropped (become 0x3F, etc.) on the distribution path or data store.
- The sort order is Unicode code point order.

2.2.3 File properties

2.2.3.1 Data structure

Attribute	Description	Type	Limitations
id	ID. Assigned by the server.	long	Unique
url	URL for locating a file resource. Used only for API responses.	String	
name	File name. User-specified. Duplicate registrations allowed.	String	Between 1 and 100 characters Allowed characters: a-z, A-Z, 0-9, *.-_
mimetype	The mimetype of a file being registered.	String	Only mimetypes described in the settings file can be used
registtime	Registration time. Elapsed time since 1970/1/1 00:00:00:00, in milliseconds.	long	long.MIN to long.MAX
lastmodifiedtime	Latest update time. Elapsed time since 1970/1/1 00:00:00:00, in milliseconds.	long	long.MIN to long.MAX
size	File size in bytes.	long	0 to 2097152
version	Version of file. The version is incremented each time a property or the file itself is updated. Processing is not performed if the specified version is different when updating or deleting a file.	int	0 to int.MAX

2.2.3.2 Get file data via file name API

Method/URL

```
GET http//server:port/arsvfdm/file/id/name
```

Request

Parameter	Description
id	ID
name	File name

Response

Header/body	Description
ETag header	Version information
Content-Type header	A registered mimetype is set in this header.
Body	File data

Example

- Request example

```
http://foo.bar.com/arsvfdm/file/287265202174368/test.jpeg
```

- Response example

```
HTTP/1.1 200 OK
X-Content-Type-Options:  nosniff
ETag:  "1"
Content-Type:  image/jpeg
Transfer-Encoding:  chunked
-File data-
```

2.2.3.3 Get file properties via file name API

Method/URL

```
GET http//server:port/arsvfdm/prop/id/name
```

Request

Parameter	Description
id	ID
name	File name

Response

Header/body	Description
ETag header	Version information
Body	File data

Example

Example of a request

```
http://foo.bar.com/arsvfdm/prop/287265202174368/test.jpeg
```

Example of a response

```
{"id":287265202174368,"name":"test.jpeg","url":"http://foo.bar.com/arsvfdm/file/287265202174368/
test.jpeg","mimetype":"image/jpeg","registtime":1359805887457,"lastmodifiedtime":
1359805887457,"size":25130,"version":1}
```

2.2.3.4 Searching file properties API

Method/URL

```
GET http//server:port/arsvfdm/prop?queryParameter
```

Request

Query parameter	Description	Value type	Constraints, etc.
type	Type of data returned as a processing result - can have the following values:	Enum	Required, default=RECORDS

Query parameter	Description	Value type	Constraints, etc.
	RECORDS - Returns result array COUNT - Returns number of results RECORDANDCOUNT - Returns result array and total number of records		
limitRange	Range of records in the search result to return. The interval start must be greater than 0, the interval end must be greater than or equal to the interval start, and the interval cannot include more than 100 items. Example: {"start":1,"end":10}	Range<Integer>	Required, default=1 to 10, interval up to 100, cannot reverse start/end
idRanges	id range array Example: [{"start":100,"end":200},{"start":300,"end":400}]	List<Range<Long>>	Up to 10 arrays
nameRanges	name range array Example: [{"start":"computer001.jpg","end":"computer999.jpg"}]	List<Range<String>>	Up to 10 arrays
mimetypeRanges	mimetype range array Example: [{"start":"image/jpeg","end":"image/png"}]	List<Range<String>>	Up to 10 arrays
registtimeRanges	registTime range array Example: [{"start":1357532800000,"end": 1357532900000}]	List<Range<Long>>	Up to 10 arrays
lastmodifiedtimeRanges	Lastmodifiedtime range array Example: [{"start":1357532800000,"end": 1357532900000}]	List<Range<Long>>	Up to 10 arrays
sizeRanges	size range array - can have the following values: Example: [{"start":1024,"end": 10000}]	List<Range<Long>>	Up to 10 arrays
sortOrders	Sort order array - can have the following values: - ID - Returns ID in ascending order - ID_DESC - Returns ID in descending order - NAME - Returns name in ascending order - NAME_DESC - Returns name in descending order - MIMETYPE - Returns mimetype in ascending order - MIMETYPE_DESC - Returns mimetype in descending order - REGISTTIME - Returns registtime in ascending order - REGISTTIME_DESC - Returns registtime in descending order - LASTMODIFIEDTIME - Returns lastmodifiedtime in ascending order - LASTMODIFIEDTIME_DESC=lastmodifiedtime in descending order - SIZE - Returns size in ascending order	List<Enum>	default=ID, up to 10 arrays

Query parameter	Description	Value type	Constraints, etc.
	- SIZE_DESC - Returns in descending order Example: ["ID","NAME"]		

Response

Header/body	Description
Body	Format for records is List<FileProperties>.

Example

Request example

```
http://foo.bar.com/arfilemanager/prop?type=RECORDSANDCOUNT&limitRange={"start":1,"end":
10}&idRanges=[{"start":100,"end":200},{"start":300,"end":
400}]&nameRanges=[{"start":"test001.jpeg","end":"test999.jpeg"}]&sortOrders=["ID","NAME"]
```

2.2.3.5 Register API

Method/URL

```
POST http//server:port/arsvfdm/file
```

Request

The file name and mimetype are required items in the file properties. Other values are set by the server.

Path/header/body		Description
Content-type header		Specifies multipart/form-data
File part Body	Content-Disposition Header	form-data;name="file"
	Body	File data
Property part Body	Content-ype	application/json
	Content-Disposition	form-data;name="prop"
	Body	JSON-formatted file properties

Response

Header/body	Description
ETag header	Version information
Body	JSON-formatted file properties

Example

- Request example

```
POST /arsvfdm/file HTTP/1.1
Content-Type:  multipart/form-data; boundary=Boundary_1_1161595825_1359806732000
--Boundary_1_1161595825_1359806732000
Content-Type:  application/json
Content-Disposition:  form-data; name="prop "
{"id":"","name":"test.jpeg","url":"","mimetype":"image/
jpeg","registtime":"","lastmodifiedtime":"","size":"","version"::""}
--Boundary_1_1161595825_1359806732000
Content-Type:  image/jpeg
```

```
Content-Disposition:  form-data; filename="test.jpeg"; name="file
"-File data-
--Boundary_1_18036279_1357532897107--
```

In JSON, keys for properties other than name and mimetype can be omitted without problem, because a missing key is deemed to be a null value.

```
{"name":"test.png","mimetype":"image/png"}
```

- Example of a response

```
HTTP/1.1 201 Created
ETag:  "1"
Content-Type:  application/json
Transfer-Encoding:  chunked
{"id":288109881061070,"name":"test.png","url":"http://foo.bar.com/arsvfdm/file/288109881061070/
test.png","mimetype":"image/png","registtime":1359806732139,"lastmodifiedtime":
1359806732147,"size":83054,"version":1}
```

2.2.3.6 Delete API

Uses the ID or version to identify a registered file, and deletes the file properties or file data.

Point

Processing is not performed if the version registered in the properties is different from the specified version.

Method/URL

```
DELETE http//server:port/arsvfdm/file/id/name
```

Request

Path/header/body	Description
id	ID
name	File name
If-Match header	Version information

Response

Path/header/body	Description
Body	Null

Chapter 3 Command reference

This chapter describes the commands provided by Interstage AR Processing Server.

The table below lists the provided commands and their descriptions:

Command name	Overview
arsvadmin	Performs various operations. Performs operations on the processing server.
arsvbackup	Backs up resources. Backs up user resources stored in the processing server. Multipe generations of backup can be created - the directory name will contain the backup datetime.
arsvrestore	Restores resources. Restores user resources backed up on the processing server.

Note

Interstage AR Processing Server supports the following server configurations:

- Model A: Small-scale entry model used in an intranet environment (all-in-one minimum configuration)
- Model B: Small-scale entry model used in an Internet environment (web servers deployed in DMZ)
- Model C: Medium- to large-scale, high reliability, high availability model used in an intranet environment (multiple-server configuration)
- Model D: Medium- to large-scale, high reliability, high availability model used in an Internet environment (in addition to above, web servers are deployed in DMZ)

The operation management features provided by Interstage AR Processing Server only support models A and B. If using a system configuration that integrates multiple chassis such as models C and D, use your existing operation management application.

3.1 Command operation modes

Commands can be operated in the modes below:

1. Run commands directly from AR server
2. Connect to AR server remotely and run commands

The command operating method is different, depending on the command and how it is used.

W

Note

The command prompt must be started by a user with administrator privileges. To do that, from the context menu, select **Run as administrator**.

3.2 arsvadmin

Name

W *installDir*\bin\arsvadmin

L /opt/FJSVar/bin/arsvadmin.sh

Description

Performs operations for web containers, database services, and applications managed by the processing server.

Format

arsvadmin *subcommand* [*subcommandOption*] [*operand*]

Subcommand

The operation to be executed

(refer to "3.2.1 Subcommands" for details).

Subcommand options

Prefix the subcommand with two hyphens (for example, --configfile).

Specifying values

The method used to specify values is explained below.

Option type	Method for specifying values	Example
Option for specifying strings and numerals	Options must be separated from values by a whitespace character.	--name *componentName*

Specifying multiple options

Use a whitespace character to separate each option. An error is output if the same option is specified more than once.

Operand

String at the end of an argument, used to specify data such as the name of the object to be operated on.

Execution privileges

The user must have operating system administrator privileges.

Log output

The progress status of commands is output to a log file.

The log output format is as follows:

[MM/dd/yyyy_HH:mm:ss.sss] FSP_*productIdentifier_componentName*: *errorType*: *messageID*: *messageBody*

FSP_*productIdentifier_componentName above is output as a fixed string: FSP_INTS-AR/SV_ADCMD.

An example of the log output is shown below:

[01/26/2013_15:16:51:200] FSP_INTS-AR/SV_ADCMD: INFO: 10120: JavaEE Server stop successfully.

Output log file name: *installDir*\logs\arsvcmd.log.X

- Do not perform conflicting operations using arsvadmin.
- Do not perform the following operations, because this might cause the command to not end normally:
  - Using Ctrl+C to terminate arsvadmin
  - Using arsvrestore while arsvadmin is running

3.2.1 Subcommands

The table below lists the arsvadmin subcommands:

Category	Operation target	Operation	Subcommand name	Outline
Web container operation	Web container	Start	start-webcontainer	Starts a web container managed by the processing server.
		Stop	stop-webcontainer	Stops a web container managed by the processing server.

Category	Operation target	Operation	Subcommand name	Outline
		Show status	show-webcontainer	Displays the status of a web container managed by the processing server.
Database service operation	Database service	Start	start-db	Starts a database service managed by the processing server.
		Stop	stop-db	Stops a database service managed by the processing server.
Application	Application	Deploy	deploy	Deploys the specified application.
		Undeploy	undeploy	Undeploys the specified application.
		Enable	enable	Enables the specified application.
		Disable	disable	Disables the specified application.
		Show status	list-components	Displays the status of applications deployed to a web container.

3.2.2 Web container operations

This section describes subcommands related to the operation of web containers.

3.2.2.1 start-webcontainer subcommand

Format

arsvadmin start-webcontainer

Description

Starts a web container managed by the processing server.

Operand

This subcommand does not allow operands.

Return value

0: Ended normally

1: Ended in an error

Example

```
> arsvadmin start-webcontainer
```

```
./arsvadmin.sh start-webcontainer
```

Output

Output is sent to the standard output,

as in the example below:

```
Command start-webcntainer executed successfully.
```

Note

The web container may still be running immediately after the command has ended normally. Therefore, in order to avoid a connection error, wait for a few moments, then reconnect to the web container.

3.2.2.2 stop-webcontainer subcommand

Format

arsvadmin stop-webcontainer

Description

Stops a web container managed by the processing server.

Operand

This subcommand does not allow operands.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> arsvadmin stop-webcontainer
```

L
```
./arsvadmin.sh stop-webcontainer
```

Output

Output is sent to the standard output,

Example

as in the example below:

```
Command stop-webcntainer executed successfully.
```

3.2.2.3 show-webcontainer subcommand

Format

arsvadmin show-webcontainer

Description

Displays the status of a web container managed by the processing server.

Operand

This subcommand does not allow operands.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> arsvadmin show-webcontainer
```

L
```
./arsvadmin.sh show-webcontainer
```

Output

Output is sent to the standard output, and the following status values can be listed:

- Started - the web container is in the started state.
- Stopped - the web container is in the stopped state.
- Partially started - the web container is in the started state, but some features are stopped.

Note

If the web container is partially started, then restart is requred. Execute the stop-webcontainer subcommand and then the start-webcontainer subcommand.

Example

Output example:

```
Name                                        Status
------------------------------------------------
Interstage AR Web Container Service      started

Command show-webcontainer executed successfully.
```

3.2.3 Database service operations

This section describes subcommands related to the operation of database services.

3.2.3.1 start-db subcommand

Format

arsvadmin start-db

Description

Starts a database service managed by the processing server.

Operand

This subcommand does not allow operands.

Return value

0: Ended normally

1: Ended in an error

Example

```
> arsvadmin start-db
```

```
./arsvadmin.sh start-db
```

Output

Output is sent to the standard output,

as in the example below:

```
Command start-db executed successfully.
```

3.2.3.2 stop-db subcommand

Format

arsvadmin stop-db

Description

Stops a database service managed by the processing server.

Operand

This subcommand does not allow operands.

Operands will be ignored if specified.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> arsvadmin stop-db
```

L
```
./arsvadmin.sh stop-db
```

Output

Output is sent to the standard output,

as in the example below:

```
Command stop-db executed successfully.
```

3.2.4 Applications

This section describes subcommands related to the operation of user-created applications.

3.2.4.1 deploy subcommand

Format

arsvadmin deploy

[--contextroot *contextRoot*] [--name *componentName*] [--libraries *jarFile*[,*jarFile*]...] *filepath*

Description

Deploys modules of web applications (.war) created by the user.

Options

Option	Mandatory	Description
--contextroot *contextRoot*	No	Specifies the context root. Refer to the Notes below for details on the application name.
--name *componentName*	No	Specifies the application name. Refer to the Notes below for details on the application name. Applications named arsvdm, arsvfdm, or arsvdmc cannot be deployed.
--libraries *jarFile*[,*jarFile*]...	No	Specifies the library (JAR file) referenced by the application, using an absolute path. Separate files using a comma (,).

Operand

Operand	Mandatory	Description
filePath	Yes	Specifies the full name of the file to be deployed.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> installDir\bin\arsvadmin deploy C:\sample.war
```

L
```
/opt/FJSVar/bin/arsvadmin.sh deploy /sample.war
```

Output

Output is sent to the standard output,

as in the example below:

```
Command deploy executed successfully.
```

Notes on the --libraries option

- An existing JAR file must be specified.
- Enclose the path with double quotation marks if it contains whitespace characters.
- Store libraries in a location that does not include a comma in the path component, otherwise the comma might be interpreted as a path separator, and deployment might fail.

Notes on application file names

When specifying the --name option, note the following about the application name:

- It must be up to 50 characters long, and the following characters are allowed:
  - Alphanumeric characters and the following symbols: hyphen (-), underscore (_), period (.), semicolon (;).
  - It can start with an underscore (_) or an alphanumeric character.
- If the --name option is specified, then application name is the value specified for *componentName*, otherwise it is the value specified for *filePath* (without the extension).

Notes on executing multiple operations

Multiple deploy/undeploy operations cannot be performed simultaneously - if a new deploy/undeploy operation is performed before the preceding one ends, the new operation is placed on standby until the end of the preceding one.

3.2.4.2 undeploy subcommand

Format

arsvadmin undeploy *componentName*

Description

Undeploys modules of web applications (.war) created by the user.

Operand

Operand	Mandatory	Description
component_name	Yes	Application name.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> installDir\bin\arsvadmin undeploy sample
```

L
```
/opt/FJSVar/bin/arsvadmin.sh undeploy sample
```

Output

Output is sent to the standard output,

as in the example below:

```
Command undeploy executed successfully.
```

Note

Multiple deploy/undeploy operations cannot be performed simultaneously - if a new deploy/undeploy operation is performed before the preceding one ends, the new operation is placed on standby until the end of the preceding one.

3.2.4.3 enable subcommand

Format

arsvadmin enable *componentName*

Description

Enables the specified application.

Operand

Operand	Mandatory	Description
componentName	Yes	Application name. This operand cannot be specified more than once.

Return value

0: Ended normally

1: Ended in an error

Example

W

```
> installDir\bin\arsvadmin enable sample
```

L

```
/opt/FJSVar/bin/arsvadmin.sh enable sample
```

Output

Output is sent to the standard output,

as in the example below:

```
Command enable executed successfully.
```

3.2.4.4 disable subcommand

Format

arsvadmin disable *componentName*

Description

Application name.

Operand

Operand	Mandatory	Description
componentName	Yes	Application name. This operand cannot be specified more than once.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> installDir\bin\arsvadmin disable sample
```

L
```
/opt/FJSVar/bin/arsvadmin.sh disable sample
```

Output

Output is sent to the standard output,

as in the example below:

```
Command disable executed successfully.
```

3.2.4.5 list-components subcommand

Format

arsvadmin list-components

Description

Displays the status of applications deployed to a web container.

Operand

This subcommand does not allow operands.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> arsvadmin list-components
```

L
```
./arsvadmin.sh list-components
```

Output

Output is sent to the standard output.

Example

Output example:

```
NAME          TYPE   STATUS
arsvdm         <web>  enabled
arsvfdm        <web>  enabled
arsvdmc        <web>  enabled
_arsvsample   <web>  disabled

Command list-components executed successfully.
```

3.3 arsvbackup

Backs up user resources - the backup points can then be used for recovery if a database or file becomes corrupted. The user resources to be backed up are managed by the AR processing server.

The following resources are backed up:

- AR markers
- Scenarios
- Scenes
- AR overlay definitions
- User-defined tables
- Files
- Database users

Name

W *installDir*\bin\arsvbackup

L /opt/FJSVar/bin/arsvbackup.sh

Description

- Backs up the Interstage AR Processing Server user resources to the directory specified in the operation management UI definition file.
- A directory will be created under *backupDir* with the datetime of the backup, and two compressed files will be created under it:
  - *backupDir**yyyyMMddHHmmss*
    yyyyMMddHHmmss is the format used for the year (four digits), month (two digits), day (two digits), hour (two digits), minute (two digits), and second (two digits).
- The backupConfiguration.ini file will be created under *backupDir*, with the following information:
  - Product version of the backup data
  - Operating system information for the backup data

Example

For example, 12:34.56 on January 23, 2014 is represented as "20140123123456".

Point

If the datetime directory already exists, then files under that directory will be overwritten.

The actual behavior in this situation will be one of the following:

- Existing files with the same name will be renamed, then if backup ends normally, the renamed files will be deleted.
- Existing files with the same name will be renamed, then if backup ends in an error, the renamed files will be reverted to their original names.

Format

arsvbackup

Execution privileges

The user must have operating system administrator privileges.

Return value

0: Ended normally

1: Ended in an error

Example

W
```
> installDir\bin\arsvbackup
```

L
```
/opt/FJSVar/bin/arsvbackup.sh
```

If a backup is performed at 15:30.45 on January 14, 2015, the following files will be created:

- *backupDir*\20150114153045\arsvdb.zip
- *backupDir*\20150114153045\arsvfile.zip
- *backupDir*\20150114153045\backupConfiguration.ini

Output

Output is sent to the standard output, as in the example below:

```
Backup to backupDir\datetimeDir
Command arsvbackup executed successfully.
```

Log output

The progress status of commands is output to a log file.

Details such as the log output format are the same as for arsvadmin.

- Do not perform the following operations, because this might cause the command to not end normally:
  - Using Ctrl+C to terminate arsvbackup
  - Using arsvadmin while arsvbackup is running
  - Using arsvrestore while arsvbackup is running

- Execute the following commands to stop Interstage AR Processing Server before running a backup command:
  - arsvadmin stop-db
  - arsvadmin stop-webcontainer

  The following message is output if Interstage AR Processing Server has not stopped running:

```
The command cannot be executed because Web Container or Database Service is running.
Command arsvbackup failed.
```

- Ensure that there is sufficient available disk space before performing a backup.

3.4 arsvrestore

Restores Interstage AR Processing Server user resources.

Name

W
installDir\bin\arsvrestore

L
/opt/FJSVar/bin/arsvrestore.sh

Description

Restores Interstage AR Processing Server user resources.

Restores user resources backed up using arsvbackup. User resources saved before running this command will no longer be accessible after the command is run. The following user resources are restored:

- AR markers
- Scenarios
- Scenes
- AR overlay definitions
- User-defined tables
- Files
- Database users

Format

arsvrestore [--target *directoryName*]

Parameters

--target *directoryName*

Specifies the backup name, using the backup datetime.

The format of the backup name is *yyyyMMddHHmmss*,

which includes the year (four digits), month (two digits), day (two digits), hour (two digits), minute (two digits), and second (two digits).

- If omitted, the directory with the latest backup will be used as the target.
- The command will not be executed if the specified directory does not exist.

Example:

```
installDir\bin\arsvrestore --target 20150114153045
```

```
/opt/FJSVar/bin/arsvrestore.sh --target 20150114153045
```

In the example above, the arsvdb.zip and arsvfile.zip backup files contained in the 20130114153045 directory under the backup directory will be used as recovery data.

Execution privileges

The user must have operating system administrator privileges.

Return value

0: Ended normally

1: Ended in an error

Output

Output is sent to the standard output,

as in the example below:

```
Restore from backupDir\datetimeDir
Command arsvrestore executed successfully.
```

Log output

The progress status of commands is output to a log file.

Details such as the log output format are the same as for arsvadmin.

- Do not perform the following operations, because this might cause the command to not end normally:

- Using Ctrl+C to terminate arsvrestore
- Using arsvadmin while arsvrestore is running
- Using arsvbackup while arsvrestore is running

- Run the following commands to stop Interstage AR Processing Server before running a restore command:
  - arsvadmin stop-db
  - arsvadmin stop-webcontainer

The following message is output if Interstage AR Processing Server has not stopped running:

```
The command cannot be executed because Web Container or Database Service is running.
Command arsvrestore failed.
```

- Ensure that there is sufficient available disk space when performing recovery, so that the current user resources can be backed up.
- The backed up user resources cannot be restored to a different platform. Contact Fujitsu technical support if you need to migrate data between Windows and Linux.

3.5 arsvsetSSL

Specifies the certificate to be used as the web container certificate.

Name:

W

```
installDir\bin\arsvsetSSL
```

L

```
/opt/FJSVar/bin/arsvsetSSL.sh
```

Description:

Specifies the certificate to be used as the web container certificate. A certificate for which you specify information must have been registered in the keystore. You must restart the web container to make any changes take effect.

Format:

arsvsetSSL *alias*

Parameters

alias

Specify the alias of the certificate registered in the keystore.

Example:

```
arsvsetSSL newcert
```

In the example above, the certificate with the alias newcert will be assigned as the web container certificate.

Execution privileges:

The user must have operating system administrator privileges.

Return values

0: Ended normally

1: Ended in an error

Output:

Output is sent to the standard output as in the example below:

```
The command completed successfully.
```